METHODOLOGY

Unsupervised encoding selection through ensemble pruning for biomedical classification

Sebastian Spänig, Alexander Michel and Dominik Heider*

*Correspondence:
dominik.heider@uni-marburg.de
Data Science in Biomedicine, Department of Mathematics and Computer Science, University of Marburg, Marburg, Germany
Full list of author information is available at the end of the article

Abstract

Background

Owing to the rising levels of multi-resistant pathogens, antimicrobial peptides, an alternative strategy to classic antibiotics, got more attention. A crucial part is thereby the costly identification and validation. With the ever-growing amount of annotated peptides, researchers employed artificial intelligence to circumvent the cumbersome, wet-lab-based identification and automate the detection of promising candidates. However, the prediction of a peptide's function is not limited to antimicrobial efficiency. To date, multiple studies successfully classified additional properties, e.g., antiviral or cell-penetrating effects. In this light, ensemble classifiers are employed to utilize the advantages of peptide encodings; hence, further improving the prediction. Although we recently presented a workflow to significantly diminish the initial encoding choice, an entire unsupervised encoding selection, considering various machine learning models, is still lacking.

Results

We developed a workflow, automatically selecting encodings and generating classifier ensembles by employing sophisticated pruning methods. We observed that the Pareto frontier pruning is a good method to create encoding ensembles for the datasets at hand. In addition, encodings combined with the Decision Tree classifier as the base model are often superior. However, our results also demonstrate that none of the ensemble building techniques is outstanding for all datasets.

Conclusion

The workflow conducts multiple pruning methods to evaluate ensemble classifiers composed from a wide range of peptide encodings and base models. Consequently, researchers can use the workflow for unsupervised encoding selection and ensemble creation. Ultimately, the extensible workflow can be used as a plugin for the PEPTIDE REACToR, further establishing it as a versatile tool in the domain.

Keywords: Biomedical classification; Antimicrobial peptides; Encodings; Machine Learning; Ensemble Learning

Background

Multi-resistant pathogens are a major threat for modern society [1]. In the last decades, a rising number of bacterial species developed mechanisms to elude efficiency to widely used antibiotics [1]. The importance of developing and implement-

ing alternative strategies is further underpinned by a recent study, which detected a certain baseline resistance in European freshwater lakes [2]. The study confirmed resistance specifically against four critical drug classes in human and veterinary health in freshwater, which is typically considered as a pathogen-free environment [2]. Moreover, already concerning levels of antibiotic resistance in Indian and Chinese lakes emphasize the requirement of alternative biocides [3, 4]. One promising approach to replace or even support common antibiotics refers to the deployment of peptides with antimicrobial efficiency [5]. However, identifying and validating active peptides requires intensive, hence, costly and time-consuming wet-lab work. Thus, in the pre-artificial intelligence (AI) era, the manual classification and verification of antimicrobial peptides (AMPs) engaged researchers. Although the *in vitro* confirmation of activity is still necessary, the application of AI, i.e., in particular machine learning (ML) algorithms, simplifies the identification process drastically and pushes specific AMPs to the second or third phase of clinical trials [6]. In addition, online databases provide access to thousands of annotated sequences and pave the way for AI application in peptide design and classification [7]. For instance, Chung *et al.* (2019) developed a method, which demonstrated good performance on classifying AMPs using a two-step approach, which first predicts efficiency, and afterward the precise target activity [8]. Another study employed a variational autoencoder to encode AMPs, mapped the probability of being active to a latent space, and predicted novel AMPs [9]. Fingerhut *et al.* (2020) introduced an algorithm to detect AMPs from genomic data [10]. For more information on computational approaches for AMP classification, we refer to the recent review of Aronica *et al.* (2021) [11].

However, the prediction of amino acid sequence features is not limited to AMPs. In the literature, one can find various applications, e.g., in oncology for predicting anticancer peptides [12], in pharmacology for the discovery and application of cell-penetrating peptides as transporters for molecules [13], or in immunotherapy, for classifying of pro- or antiinflammatory peptides [14, 15]. Other applications include antiviral peptides [16], or peptides with hemolytic [17] or neuro transmitting activity [18].

Unequivocally, the success of ML methods for the prediction of AMPs was enabled by the development and advances of peptide encodings. Encodings are algorithms mapping the amino acid sequences of different lengths to numerical vectors of an equal length, hence, fulfilling the requirement of many ML algorithms [19]. Moreover, peptides or proteins can be described by their primary structure, i.e., the amino acid sequence, and the aggregation in higher dimensions, denoted as the secondary or tertiary structure. Encodings derived from the primary structure are known as sequence-, and encodings describing a higher-order folding are structure-based encodings. To date, a large number of sequence- and structure-based encodings have been introduced and employed in various studies [19]. A significant amount of encodings has been recently acknowledged by another study, specifically benchmarking these by considering multiple biomedical applications [20]. It turned out that most encodings show acceptable performance, partly also beyond single biomedical domains [20]. In addition, Spänig *et al.* (2021) developed a workflow, which can dramatically reduce the number of initial encodings [20]. However, encoding selection is still challenging, and user-friendly approaches are required.

Furthermore, hyperparameter optimization is additionally aggravated by the model choice. Albeit Support Vector Machines (SVM) and Random Forests (RF) are widely employed in peptide classification [11], the variety of models used in a broad range of studies is large. For instance, Khatun *et al.* (2020) utilized several ML algorithms, including Naïve Bayes, AdaBoost, and a fusion-based ensemble for the prediction of proinflammatory peptides [21]. The fusion-based model outperformed the other ML models significantly for this task [21]. Plisson *et al.* (2020) employed Decision Trees (DT) and Gradient Boosting (GB), among others, to classify non-hemolytic peptides and demonstrated that the GB ensemble has superior performance [22]. In contrast, Timmons *et al.* (2020) used Artificial Neural Networks to characterize therapeutic peptides with hemolytic activity [23]. Singh *et al.* (2021) compared several base classifiers, e.g., Linear Discriminant Analysis and ensemble methods, e.g., GB and Extra Trees to detect AMPs [24]. They demonstrated that the GB performed best [24]. These studies clearly show that ensemble classifiers typically show superior performance than single classifiers, which is based on the fact that they can compensate for weaknesses of single encodings and base classifiers [25].

Recently, Chen *et al.* (2021) introduced a comprehensive tool, which allows less programming experienced researchers to simply select encodings and base or ensemble classifiers through a graphical user interface, allowing easy access to the underlying algorithms [26]. Nevertheless, the approach assumes that the user selects proper settings for the parameterized encodings, which has been previously shown to affect the classification process significantly [20]. Moreover, the encoding selection is independent of the classifier settings, meaning that the tool can set up the classifier automatically; however, the encoding selection is not part of it. Thus, it remains a challenge to pick good encodings and classifiers for a biomedical classification task at hand. To this end, we assessed unsupervised encoding selection and the performance and diversity of multiple ensemble methods. We added different overproduce-and-select techniques for ensemble pruning, facilitating an automatic ensemble generation. In addition, we utilized Decision Trees, Logistic Regression, and Naïve Bayes as base classifiers, owing to their prevalence in the field of biomedical classification due to their explainability [11, 19, 27].

Besides demonstrating the benefit of an unsupervised encoding selection, we also examined how the RF performs as a base and ensemble classifier, i.e., whether the RF, an ensemble method per se, is performance-wise already saturated or whether a subsequent fusion can improve the final predictions. Fusion of RFs has been shown in other studies to improve overall performance, e.g., for HIV tropism predictions [28, 29]. All in all, we complement our recent large-scale study on peptide encodings [20] with an automatic encoding selection and a performance analysis of multiple base and ensemble classifiers. Ultimately, the present research bridges the gap between many peptide encodings and available machine learning models.

Results

We developed an end-to-end workflow, which automatically generates and assesses classifier ensembles using different pruning methods and a variety of encoded datasets from multiple biomedical domains (see Table 4). Data scientists can easily

Table 1 The table shows the performance comparison (including RF) of classifier ensembles derived from different pruning methods and the single best classifier. Numbers refer to the mean performance of a 100-fold Monte Carlo cross-validation. Standard deviation (SD) is added in brackets. Mean and SD are rounded to 2 decimal places. The top base/ensemble classifier combination is always used (see Fig. 2). Classifier ensembles are significantly better than the single best classifiers. In particular, except for one case, the Pareto frontier pruning (pfront) generates the best ensembles. Significance levels are as follows: ** $p \leq 0.001$, * $p \leq 0.01$, and . $p \leq 0.05$.

	best	chull	mvo	pfront	rand	rand_single_best	single_best
acp_mlacp	0.73 (±0.06)	0.73 (±0.06)	0.7 (±0.07)	0.74** (±0.06)	0.69 (±0.06)	0.68 (±0.07)	0.69 (±0.07)
aip_antiinflam	0.5** (±0.04)	0.5 (±0.04)	0.45 (±0.04)	0.5 (±0.04)	0.48 (±0.04)	0.47 (±0.04)	0.47 (±0.04)
amp_antibp2	0.88 (±0.02)	0.89 (±0.02)	0.88 (±0.02)	0.9** (±0.02)	0.87 (±0.02)	0.84 (±0.02)	0.87 (±0.03)
atb_antitbp	0.75 (±0.07)	0.76 (±0.08)	0.72 (±0.08)	0.79** (±0.07)	0.7 (±0.06)	0.66 (±0.07)	0.68 (±0.07)
avp_amppred	0.79 (±0.03)	0.8 (±0.03)	0.77 (±0.02)	0.81** (±0.03)	0.79 (±0.03)	0.76 (±0.03)	0.76 (±0.03)
cpp_mlcpp	0.77 (±0.03)	0.78 (±0.03)	0.78 (±0.03)	0.79** (±0.03)	0.76 (±0.03)	0.74 (±0.03)	0.75 (±0.03)
hem_hemopi	0.88 (±0.03)	0.89 (±0.03)	0.87 (±0.03)	0.89** (±0.03)	0.88 (±0.03)	0.86 (±0.03)	0.87 (±0.03)
isp_il10pred	0.59 (±0.05)	0.59 (±0.05)	0.6 (±0.06)	0.6** (±0.05)	0.57 (±0.05)	0.58 (±0.04)	0.58 (±0.04)
nep_neuropipred	0.79 (±0.03)	0.81 (±0.02)	0.81 (±0.04)	0.81** (±0.03)	0.81 (±0.03)	0.76 (±0.03)	0.78 (±0.03)
pip_pipel	0.5 (±0.04)	0.52 (±0.04)	0.5 (±0.05)	0.53** (±0.04)	0.47 (±0.04)	0.41 (±0.04)	0.49 (±0.03)

Table 2 The table shows the performance comparison (excluding RF) of classifier ensembles derived from different pruning methods and the single best classifier. See Table 1 for more details.

	best	chull	mvo	pfront	rand	rand_single_best	single_best
acp_mlacp	0.72 (±0.06)	0.73 (±0.06)	0.69 (±0.04)	0.74** (±0.06)	0.69 (±0.06)	0.66 (±0.07)	0.67 (±0.07)
aip_antiinflam	0.47 (±0.04)	0.48 (±0.04)	0.44 (±0.05)	0.48** (±0.04)	0.41 (±0.04)	0.36 (±0.04)	0.44 (±0.04)
amp_antibp2	0.88 (±0.02)	0.88 (±0.02)	0.87 (±0.02)	0.89** (±0.02)	0.86 (±0.02)	0.84 (±0.02)	0.87 (±0.03)
atb_antitbp	0.73 (±0.06)	0.76 (±0.08)	0.67 (±0.04)	0.79** (±0.07)	0.68 (±0.07)	0.65 (±0.08)	0.68 (±0.07)
avp_amppred	0.76 (±0.04)	0.77 (±0.04)	0.73 (±0.02)	0.81** (±0.03)	0.74 (±0.04)	0.7 (±0.04)	0.73 (±0.03)
cpp_mlcpp	0.74 (±0.03)	0.75 (±0.03)	0.73 (±0.02)	0.78** (±0.03)	0.74 (±0.03)	0.71 (±0.03)	0.71 (±0.03)
hem_hemopi	0.87 (±0.03)	0.89 (±0.03)	0.87 (±0.03)	0.89** (±0.03)	0.86 (±0.03)	0.86 (±0.03)	0.86 (±0.03)
isp_il10pred	0.59 (±0.05)	0.57 (±0.05)	0.59 (±0.08)	0.6** (±0.05)	0.57 (±0.05)	0.58 (±0.04)	0.58 (±0.04)
nep_neuropipred	0.79 (±0.03)	0.79 (±0.03)	0.79 (±0.02)	0.8** (±0.03)	0.74 (±0.03)	0.65 (±0.04)	0.78 (±0.03)
pip_pipel	0.48 (±0.04)	0.45 (±0.04)	0.47 (±0.05)	0.48** (±0.04)	0.45 (±0.03)	0.38 (±0.03)	0.38 (±0.03)

extend the workflow with different base and ensemble classifiers, pruning methods, encodings, and datasets. The results can be reviewed using the provided data visualizations, and the performance is further revised using multiple statistics. We demonstrate that the Pareto frontier pruning is a valuable technique to generate efficient classifier ensembles. However, the utilized base classifiers show comparable performance, with the Decision Tree classifier being the model of choice for most datasets. We address the results in more detail in the following. We use the example of the avp_amppred dataset throughout the manuscript. The results for the remaining datasets can be found in the supplement. Moreover, the code is publicly available at https://github.com/spaenigs/ensemble-performance. Note that the workflow produces interactive versions of all charts.

Pruning methods

All pruning methods generate ensembles, i.e., combined encodings, superior to the single best classifier, i.e., individual encodings (see Tables 1 and 2). In the case of the Pareto frontier (pfront) pruning, which is predominantly ranked among the best pruning methods, we observe a significant ($p \leq 0.001$) performance improvement compared to the single best classifier. We also observed that the pfront pruning generates larger ensembles than the convex hull (chull) pruning, which can be visually verified in Fig. 1 (red line). Notably, including the Random Forest (RF) classifier (see Table 1, pfront) does not, or very slightly, affect the ensemble performance without RF (see Table 2), although the single best classifier performance is better with the RF included (see Table 1). Consequently, the RF increases the overall performance of the ensembles generated by the best encodings pruning. Finally, the multi-verse optimization (MVO) suffers from high computational demand, i.e., a long pruning time, and in general, an inferior performance compared to the other techniques.

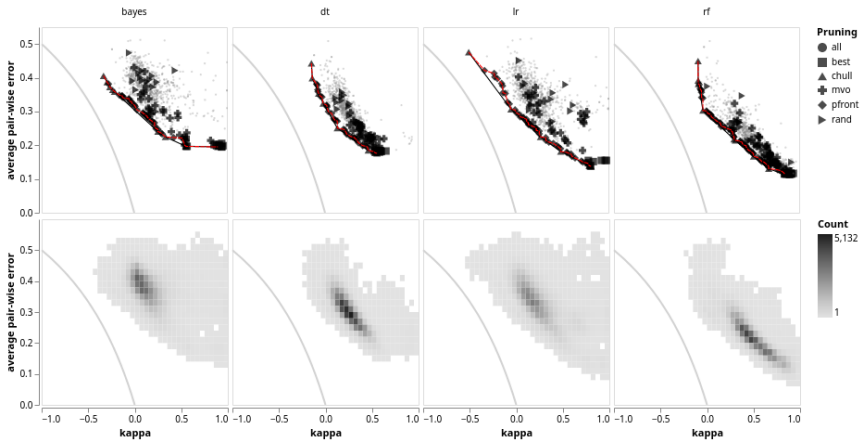


Figure 1 The kappa-error diagram depicts pairs of base classifiers (gray dots) using the kappa diversity on the x-axis and the average error on the y-axis (top row). Each classifier pair denotes two particular encodings. The black line indicates the convex hull, and the red line the Pareto frontier. Ensemble types are represented as symbols. The top row shows the example of the first fold. The bottom row groups the results of all folds in a 2D histogram. The darker the color, the more classifier pairs are binned in one group. The gray line in all panels depicts the theoretical boundary (refer to the method section for more details). The plot shows the example of the avp_amppred dataset.

Ensemble classifiers

The ensemble performance mainly depends on the pruning and the choice of the base classifiers; hence, the collection of individual encodings. Thus, the performance differences among the single best (single_best) and best random (rand) pruning are insignificant, which is in contrast to the remaining methods (see Fig. 2). Furthermore, no significant difference can be observed for ensembles with the same base classifiers, e.g., the RF or Decision Tree (DT). Thus, the fusion method impacts the overall performance slightly. However, various base classifiers result in significantly different ensembles, i.e., employing, for instance, the RF, generates significantly different ensembles compared to the application of other base classifiers (see Fig. 2).

Moreover, it is noticeable that the Naïve Bayes (NB) and the Logistic Regression (LR) classifiers result in ensembles with higher variance (see Fig. 1). In contrast, the area covered by RF and DT models is more compact. Therefore, the variables, i.e., diversity and the pairwise error, are revised by a multivariate analysis of variance (MANOVA), which revealed a significant difference ($p < 0.001$). A separate examination of the variables utilizing variance analysis (ANOVA) followed by a post-hoc analysis using Tukey's HSD, demonstrates that all variables are significantly different ($p < 0.001$). Finally, we conducted an ANOVA on the particular area values, which disproves the initial observation, i.e., all areas are significantly different ($p < 0.001$). However, considering the average values for all datasets, the DT and RF are commonly ranked as the base classifiers with low variance (see Table 3).

Single classifiers

In general, the performance of the base classifiers, i.e., single encodings, is lower compared to the classifier ensembles (see Fig. 3). We also observed that the ran-

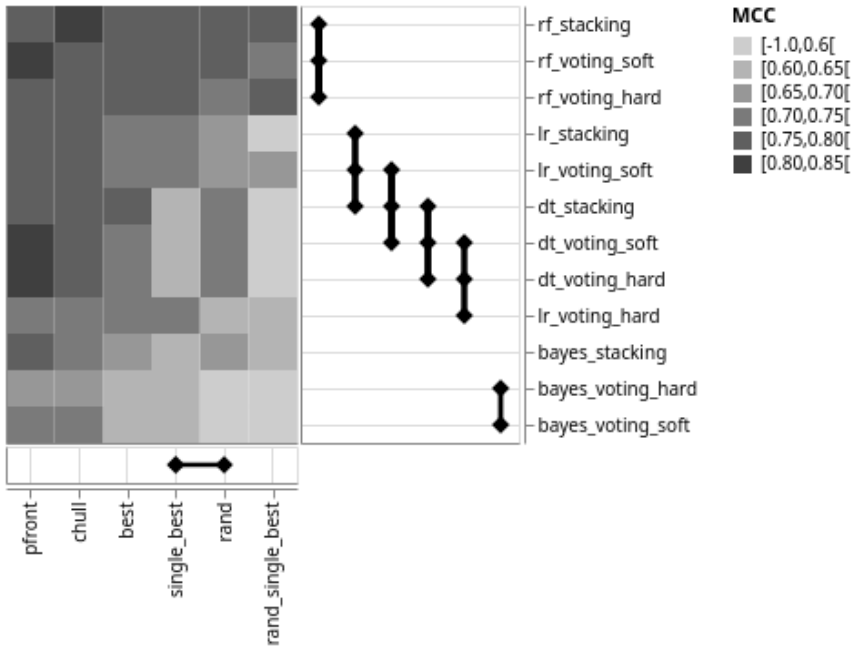


Figure 2 The XCD chart shows the difference between pruning methods (x-axis) and ensembles (y-axis). Entities connected with a bold line are not significant different. The higher the performance, the darker the color. The plot shows the example of the avp_amppred dataset.

domly selected model (rand_single_best) is inferior to the best model (single_best). In addition, we noticed that the RF is relatively saturated, i.e., using the RF as a single classifier and as a base model for ensembles does not have a significant effect on performance improvement. The low-performance variance is in line with the observation that weak models benefit most from ensemble learning; however, RFs are ensemble models [30, 31]. In contrast, the performance of other single classifiers revealed more distinct differences to the ensembles (see Fig. 3).

Data visualization

We leveraged two standard visualization techniques, which we adapted and extended for our particular application. First, we enhanced the kappa-error diagram [25] for

Table 3 The table lists the average area (±SD) covered by the base classifiers across the 100-fold Monte Carlo cross-validation. The lowest area per dataset is highlighted in bold. The DT classifier has the lowest area for most of the datasets, i.e., the predictions are more stable. Refer to Fig. 1 (bottom) for the example showing the avp_amppred dataset.

	bayes	dt	lr	rf
acp_mlacp	3.15 (±0.073)	2.6 (±0.08)	2.66 (±0.046)	**2.55** (±0.081)
aip_antiinflam	2.75 (±0.066)	2.27 (±0.045)	2.41 (±0.033)	**2.14** (±0.045)
amp_antibp2	3.01 (±0.077)	**2.5** (±0.056)	3.07 (±0.122)	2.63 (±0.061)
atb_antitbp	3.18 (±0.124)	2.82 (±0.059)	3.16 (±0.094)	**2.73** (±0.069)
avp_amppred	2.96 (±0.054)	**2.38** (±0.05)	3.15 (±0.081)	2.42 (±0.054)
cpp_mlcpp-complete	2.9 (±0.086)	**2.37** (±0.073)	2.48 (±0.049)	2.42 (±0.079)
hem_hemopi	3.2 (±0.06)	**2.74** (±0.135)	3.07 (±0.076)	2.79 (±0.122)
isp_il10pred	2.96 (±0.059)	2.45 (±0.046)	**2.38** (±0.031)	2.39 (±0.047)
nep_neuropipred	3.23 (±0.132)	**2.61** (±0.081)	3.18 (±0.309)	2.68 (±0.079)
pip_pipel	3.18 (±0.053)	**2.19** (±0.034)	2.23 (±0.024)	2.3 (±0.069)

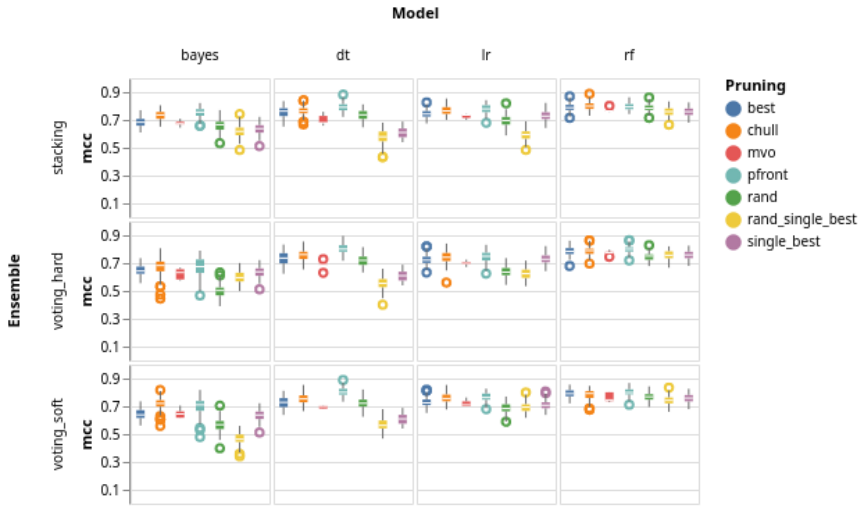


Figure 3 The box plot shows the Matthews correlation coefficients (MCC) distribution using a 100-fold Monte Carlo cross-validation. The columns include the base, i.e., diverse encodings, and the rows depict the ensemble classifiers. Colors refer to the pruning method. Note that the distributions of the best single (single_best) and best random single models (rand_single_best) are independent of the ensemble type. Variations within rand_single_best are due to the random selection. Moreover, MVO pruning (mvo) has only been conducted on the first five folds (see Discussion). The plot shows the example of the avp_amppred dataset.

the presentation of multiple folds, i.e., 100 in the current study, by aggregating the cross-validation results into a two-dimensional histogram (see Fig. 1). The color code allows the viewer to spot the peak at one glance. Hence, the tendency of ensembles to use a specific base classifier. Moreover, considering the distribution of the variables, one can make conclusions about the robustness.

Second, we extended the critical difference (CD) chart [32] with a categorical heatmap displaying the actual performance. The extension enables viewers to statistically compare classifiers and review the individual encoding performance, i.e., Matthews correlation coefficient in the present case, at one glance. In addition, the thickness of the vertical and horizontal rules is directly related to the critical difference, i.e., the thicker the rule, the closer the classifiers to the critical difference. Thus, the rule thickness provides an additional visual channel to access the CD.

Discussion

We developed a workflow for unsupervised encoding selection and performance assessment of multiple ensembles and base classifiers. Thus, we implemented and compared several algorithms to facilitate ensemble pruning, including convex hull, Pareto frontier pruning, and multi-verse optimization (MVO). Our results demonstrate that the crucial factors are the base classifiers and the individual encodings. The ensemble technique was not relevant, i.e., we could not observe performance variations using one of hard or soft voting or stacking. In general, applying the Decision Tree (DT) as a base classifier yielded good performance across all datasets. The Pareto frontier pruning selected suitable encodings throughout the experiments.

However, since we used one encoding per base classifier, we restricted the employed ensemble methods, i.e., majority voting, averaging, and stacking, which do

not modify the base classifiers. These ensemble types are in contrast to others, e.g., boosting, where weights are adapted for misclassified training instances in base classifiers [33]. More research is necessary to investigate how performance and more sophisticated ensemble methods are associated. The employed ensemble types are also the reason for the kappa-error point cloud shape solely depending on the base classifiers. Consequently, computing the kappa-error diagram for all ensemble methods was unnecessary. Our encoding/classifier approach is also contrary to other studies, e.g., [12], [14], or [16], which concatenated several encoded datasets to one final dataset (hybrid model) and applied feature selection before training. In the present study, we solely scaled the datasets to standardize the feature range; nevertheless, used the encoded datasets largely unprocessed, potentially affecting the final performance.

As mentioned above, we employed several methods for ensemble pruning comprising best single and random encodings for reference. In general, utilizing the Pareto frontier pruning generates good ensembles; however, requiring the calculation of the Cartesian product of all base classifiers; thus, encodings. Although only the (lower) triangular matrix is necessary, the computation is still CPU-intensive. Furthermore, considering the performance gain compared to the single best encodings, the diversity contribution is only small, but more research is required in this direction [34]. The results of the MVO also acknowledge the impact of diversity. One can observe that the MVO generates inferior ensembles (see Fig. 3).

Regarding Fig. 1, which depicts preferable classifier pairs towards the lower-left corner, one can readily recognize the inferiority of the MVO. The classifier pairs are distributed across the kappa-error area, i.e., the MVO screens the entire solution space and adds weak classifiers to the final ensemble. Nevertheless, since we limited the maximum number of generations to 15, we cannot rule out that more generations would yield better results. Moreover, due to high resource consumption, we limited the MVO to 5 folds, which might hamper comparison.

Moreover, the Random Forest (RF) deployment as a single classifier reveals good performance, which is expected since it is already an ensemble algorithm per se. With this respect, the other base classifiers are less accurate (see Fig. 3). However, it could be demonstrated that RFs as base classifiers, i.e., using different encoded datasets per model, slightly improves the performance. This further highlights the importance of different encodings, hence the projection of different biological aspects, for the classification process.

The implemented methods demonstrate usability on a broad range of datasets from various biomedical domains. With this respect, we incorporated the MVO owing to its excellent and promising performance on several benchmark datasets [35]. The comprehensive Monte Carlo cross-validation copes with the variance, ultimately increasing the robustness of the results. In addition, the Pareto frontier and convex hull pruning consider simultaneously the performance and the diversity of encodings and base classifiers; hence, compensating their strength and weaknesses and revealing their potential not only for ensembles [36], but also in particular for biomedical classification. Our proposed extension to the critical difference chart allows the viewer at one glance to grasp significant, i.e., critical, performance differences of encodings, models, and pruning methods jointly with the actual performance (see Fig. 2).

Conclusions

In summary, we employed two overproduce-and-select methods, namely Pareto frontier and convex hull pruning, as well as the multi-verse optimizer for exhaustively searching the encoding/base classifier space. We employed Logistic Regression, Decision Trees, Naïve Bayes, and Random Forest as base models and majority vote, averaging, and stacked generalization for the fusion. The experiments and visualizations enable the comparison of the respective components; however, further research is necessary to examine other ensemble classifiers, e.g., boosting. All in all, we propose an extensible workflow for automated encoding selection through diverse ensemble pruning methods. Researchers can utilize our workflow to augment the recently published PEPTIDE REACToR [20] with an unsupervised encoding selection, ultimately easing the access for non-technical users.

Methods

We developed a high-throughput workflow using Snakemake v6.5.1 [37], Python v3.9.1, and R v4.1.0. For the machine learning algorithms, we employed scikit-learn v0.24.2 [38]. The peptide datasets are taken from the PEPTIDE REACToR [20]. Finally, only encoded datasets with the final sequence- and structure-based encodings were used for the subsequent analyses.

Note that there are two approaches to harness multiple encodings in a single model, namely the fusion and the hybrid model [21]. Fusion models train one encoding per base classifier and fuse the output for the final prediction. Contrary, hybrid models use the concatenated features of multiple encodings for single model training. The concatenation approach is particularly problematic for entropy-based models such as DT or RF due to the bias in variable selection. Thus, in the present study, we implemented the fusion design, i.e., each ensemble consists of an arbitrary amount of base classifiers using one particular encoding, respectively. Finally, the employed datasets from a wide range of biomedical domains ensure broad applicability and the robustness of our results.

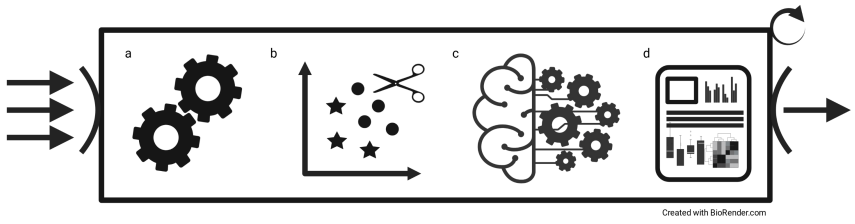


Figure 4 Overview of the workflow. (**a**) For each fold of the MCCV, the preprocessing is conducted, i.e., the indices of the train/test splits are determined, and the data is scaled. (**b**) The pruning methods, e.g., Pareto frontier and MVO, select the current fold's encodings and the number of base classifiers. (**c**) Different ensembles with various base classifiers are trained and validated on the test data. (**d**) The results are collected, statistically validated, and illustrated. The workflow accepts an arbitrary number of datasets as input (arrows). Refer to the method section for more details.

The workflow conducts the following steps. First, indices are determined to ensure equal samples for the comprehensive cross-validation, and the indices for all folds are calculated. Second, we standardized the encoded datasets using a min-max

Table 4 Employed datasets in this study. The function refers to the positive class, i.e., sequences of class + possess the respective function. The stated MCC refers to the performance reported in the original study. See the references or [20] for more details.

Name	Function	MCC	Size (+,-)	Ref.
acp_mlacp	Anti-cancer	0.698	581 (185,396)	[12]
aip_antiinflam	Anti-inflammatory	0.45	2124 (863,1261)	[14]
amp_antibp2	Anti-microbial	0.84	1975 (981,994)	[40]
atb_antitbp	Anti-tubercular	0.52	492 (246,246)	[41]
avp_amppred	Anti-viral	0.8	1476 (738,738)	[16]
cpp_mlcpp	Cell-penetrating	0.793	1901 (737,1164)	[13]
hem_hemopi	Hemolytic	0.52	1013 (522,461)	[17]
isp_il10pred	Immunosuppressive	0.59	1242 (394,848)	[42]
nep_neuropipred	Neuropeptides	0.67	1750 (875,875)	[18]
pip_pipel	Pro-inflammatory	0.454	3228 (833,2395)	[15]

normalization between 0 and 1. Afterward, we trained and assessed models for all encoded datasets and ensemble types using a 100-fold Monte Carlo cross-validation. We selected the best single and the random best encoding per dataset to compare the results to single encodings. Finally, we statistically assessed and visualized the results (see Fig. 4). Significant steps are described in more detail below. We will use the following definitions throughout the manuscript: the original unprocessed dataset is denoted as the dataset. One dataset can be encoded in manifold ways, which we refer to as encoded datasets. Encodings specify particular encoding algorithms.

Note that we used Matthews correlation coefficient (MCC) throughout the study to handle the imbalance in the datasets [39]:

$$\mathrm{MCC} = \frac{a \times d - c \times b}{\sqrt{(a+c)(a+b)(d+c)(d+b)}}. \tag{1}$$

a is the number of true positives, d is the number of true negatives, b is the number of false negatives, and c is the number of false positives.

Datasets

For a comprehensive analysis on peptide encodings, Spänig *et al.* (2021) gathered a variety of datasets from multiple biomedical domains [20]. We specifically selected datasets with low to medium classification performance from this collection, i.e., a reported MCC of 0.63 ± 0.15 on the independent test set; additionally, covering diverse biomedical applications. Moreover, we excluded datasets for which accurate models have been published to investigate the potential effects of different classifiers and ensembles. We limited our study to ten datasets to cope with the computational complexity. The dataset size ranges from 492 to 3,228 sequences with an average of $1,580.8 \pm 812.1$ sequences. The datasets comprise 15,782 sequences with a mean length of 21.17 ± 13.23 amino acids. 6,404 sequences belong to the positive and 9,378 to the negative class. The average sequence length is 22.47 ± 15.88 and 20.29 ± 10.97, respectively. Duplicated sequences have been removed. Refer to Table 4 for more details.

Monte Carlo cross-validation

We applied the Monte Carlo cross-validation (MCCV) [43]. The MCCV improves the generalization and diminishes the variance of the results, i.e., results are more robust, hence comparable. In addition, we ensured that the n-th fold is identical across all experiments leading to improved comparability across all base classifiers and ensembles. Each fold is composed of one split using 80 % of the data for model training and another utilizing the remaining 20 % for testing. In contrast to k-fold cross-validation, MCCV follows a sampling with replacement strategy, i.e., splits can contain identical samples multiple times. However, duplicate samples do not occur in the train, and the test split [43].

Base classifiers

We used the following base classifiers for our experiments: Naïve Bayes, Logistic Regression, Decision Tree, and Random Forest. Each classifier will be briefly described hereinafter. We used the implementations provided by the scikit-learn library [38].

Naïve Bayes

The Naïve Bayes (NB) classifier (naively) assumes conditional independence of the feature vectors and applies the Bayes theorem for prediction [25]. Model training is enabled via a probability density function (PDF) and the prior probability of a given class. For simplicity, we assume a Gaussian distribution of the features. Hence, we applied the Gaussian NB using

$$p(x|y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{2}$$

as the PDF, whereby σ denotes the standard deviation and μ the mean of features x given a class y [44].

Logistic Regression

The binary Logistic Regression (LR) is another probability-based classifier, i.e., it derives the probability of a class y given a feature vector x [45]. The LR predicts probabilities between 0 and 1 using the logistic function denoted as

$$p(x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}} \tag{3}$$

and the maximum likelihood function to estimate the coefficients β, i.e., to train the model [45].

Decision Tree

The Decision Tree (DT) classifier, precisely the CART (Classification And Regression Trees) implementation, is a tree-based model, i.e., a tree structure is generated during training [46]. Each node is based on the most discriminating feature [25]. New splits are created based on the impurity of the remaining data, i.e., if a split is pure enough, a leaf node is added. Otherwise, intermediate nodes are created [25].

For prediction, the tree is traced until a leaf node, which states the final class. In particular, we used the Gini impurity, denoted as

$$i(t) = 1 - \sum_j P_j^2, \tag{4}$$

where $j \in \{0, 1\}$ for binary classification and P is the probability of class j at a node t [25].

Random Forest

The Random Forest (RF) classifier is an ensemble learning technique, which trains multiple DTs on random samples, i.e., bagging, of the input data [47]. For the final classification, the majority vote of the trees is used [47]. Note that we use the RF as a base learner, which allows comparing the performance with DTs and the actual ensembles techniques in general (see below).

Classifier ensembles

To combine the output specifically of the base classifiers introduced above, we employed the following ensemble methods: majority vote (hard voting), averaging (soft voting), and stacked generalization (stacking). In the present study, each base classifier is trained on one encoded dataset, meaning if for one dataset n encodings are selected, the size of one ensemble is n. We adapted the implementations of the scikit-learn library [38], such that not only one dataset but several encoded datasets can be used for training. For instance, if one passes n encoded datasets, the ensemble consists of n base classifiers trained on one particular encoded dataset, respectively.

Majority voting The majority voting ensemble (hard voting) combines the output by ultimately assigning the class, which has been predicted by the majority of the single base classifiers. We employed the customized version of scikit-learn's VotingClassifier class with hard voting enabled.

Averaging The averaging method (soft voting) computes the means of the predicted class probabilities per base classifier. The maximum value determines the final class. We used the adjusted VotingClassifier with voting set to soft.

Stacked generalization The stacking approach utilizes the output of the base classifiers to train a meta-model, i.e., the predicted class probabilities of the base classifiers are used as features [48]. We adapted the StackingClassifier from the scikit-learn package and employed Logistic Regression as the meta-model.

Ensemble pruning

Selecting the correct number of base classifiers in an ensemble is challenging. Thus, Kuncheva (2014) suggests several approaches to determine the ensemble size [25]. For instance, sequential forward selection, adding one classifier successively, in case the additional model improves the ensemble performance [25]. However, in the

present case, we are dealing with potentially hundreds of encoded datasets, for which this particular technique is not practical. To this end, we used two selection methods, namely convex hull and Pareto frontier pruning, circumventing the limitations mentioned above [25].

Moreover, we implemented the multi-verse optimization algorithm as an automatic encoding selection technique [49]. Finally, we employed best and random encodings selection as a baseline reference. The pruning methods are described more precisely in the following.

Kappa-error diagram

The kappa-error diagram, introduced by Margineantu and Dietterich (1997), is the basis for the convex hull and Pareto frontier pruning [50]. The graph represents pairs of classifiers by their average error and diversity, as shown in Fig. 1. The diversity measures the agreement of classifier outputs, i.e., the better the agreement of the classifier predictions, the less the diversity [25]. Specifically, the kappa diversity is denoted as

$$\kappa = \frac{2(ad - bc)}{(a+b)(b+d) + (a+c)(c+d)}. \tag{5}$$

The κ statistic ranges from -1 to 1, whereby $\kappa = 1$ denotes perfect agreement, $\kappa = 0$ random, and $\kappa < 1$ worse than random consensus [50]. The error is calculated using

$$e = 1 - \frac{a+d}{a+b+c+d}, \tag{6}$$

with the subtrahend being the accuracy. However, Kuncheva (2013) pointed out that diversity concerning the average error can not be arbitrarily low [36]. In fact, desirable classifier pairs approximate the lower-left corner (see Fig. 1), i.e., approximating a theoretical boundary, which is defined in Eq. 7 [36].

$$\kappa_{min} = \begin{cases} 1 - \frac{1}{1-e}, & \text{if } 0 < e \leq 0.5 \\ 1 - \frac{1}{e}, & \text{if } 0.5 < e < 1 \end{cases} \tag{7}$$

Note that the classifier pairs are composed using the lower triangular matrix of the Cartesian product. Afterward, the pruning methods select a subset of pairs, also likely include duplicated base classifiers. Thus, all pruning methods ensure that the final ensemble only uses unique classifiers. Hence, base classifiers are trained on individual encoded datasets.

Convex hull

The kappa-error diagram depicts a set of points, i.e., pairs of base classifiers, in a two-dimensional space. The kappa diversity is the first, and the pairwise average

error is the second dimension. We employed the Quickhull algorithm to calculate the convex hull [51]. Hence, the smallest convex set that contains the classifier pairs [51]. Thus, no further classifier pairs exist beyond the convex hull. We utilized the implementation of the Quickhull algorithm provided by the SciPy package in the ConvexHull module [52].

Since we are only interested in the partial convex hull, that is, pairs approaching the theoretical boundary defined in Eq. 7 and depicted in Fig. 1, we adapted the *pareto_n* algorithm from Kuncheva (2014), which returns only classifier pairs fulfilling the criteria [25].

Pareto frontier

The Pareto optimality describes the compromise of multiple properties towards optimizing a single objective [53]. For instance, a pair of classifiers is Pareto optimal if improving the diversity is impossible without simultaneously impairing the average pairwise error. Analog to the partial convex hull introduced earlier, Pareto optimal classifier pairs approach the theoretical boundary as stated in Eq. 7, ultimately defining the Pareto frontier. Again, we used the *pareto_n* algorithm adapted from Kuncheva (2014) to obtain all classifier pairs determining the Pareto frontier (see Fig. 1).

Multi-verse optimization

The multi-verse optimization (MVO) algorithm is inspired by the alternative cosmological model stating that several big bangs created multiple, parallel existing universes, which are connected by black and white holes and wormholes [35]. In terms of an optimization algorithm, black and white holes are used to explore the search space and wormholes to refine solutions [35]. Moreover, the inflation rate, i.e., the fitness, of universes is used for the emergence of new holes; thus, to cope with local minima [49]. For more details, refer to Mirjalili *et al.* (2016) and Al-Madi *et al.* (2019) [35, 49]. We implemented the binary MVO following [49] using Python. Each solution candidate is represented as a binary vector, where each position denotes the path to an encoded dataset, that is, the i-th bit set means that the i-th encoding is included in the final ensemble (see Fig. 1). We examined different generations, i.e., 100, 80, 50, 25, and 15. However, we observed that performance depends mainly on the initialization and count of the universes. Specifically, the performance gain from the 15th generation is minor but requires much time. Thus, we set the optimization to a maximum of 15 generations with 32 universes each. Due to its resource intensity, we executed the MVO only for the first five folds (see section Monte Carlo cross-validation).

Best encodings

A further pruning method uses only the best classifier pairs. In particular, based on the kappa-error diagram, the algorithm selects 15 classifier pairs with the lowest pairwise average error (see Fig. 1).

Random encodings

The last pruning method selects 15 random classifier pairs from the kappa-error diagram. Note that the selection is only performed one time. That is, the pairs are the same across all folds.

Statistics

We examined the areas covered by the respective base classifiers (see Fig. 1). To this end, we calculated the area for each fold. The area is described by multiple variables, i.e., the kappa diversity and the average pairwise error. Thus, we applied the multivariate analysis of variance (MANOVA) to verify if the areas differ significantly. If this is the case, we subsequently employed an analysis of variance (ANOVA) to investigate the effect of the diversity and the average error separated. For post-hoc assessment, Tukey's HSD has been applied. We used the tests provided by the R standard library. α was set to 0.05, i.e., p values ≤ 0.05 are considered as significant.

In addition, we employed the Friedman test with the Iman and Davenport correction for the statistical comparison of multiple single and ensemble classifiers [54]. In the case at least one model is significantly different, we used the Nemenyi test for post-hoc analysis [54]. Refer also to Spänig *et al.* (2021) for more details [20]. The tests were provided by the scmamp R package v0.2.55 [32].

Finally, we examined if the best ensemble has a significant improvement over the best single classifier using Student's t-test for repeated measures, i.e., paired samples. Again, α was defined as 0.05.

Data visualization

All plots are realized using Altair v4.1.0 [55] and described in more detail hereinafter.

Kappa-error diagram

The kappa-error diagram, suggested by Margineantu and Dietterich (1997) [50], shows the result of a single split in the top row and a two-dimensional histogram aggregating all folds in the bottom row (see Fig. 1). The columns show the base classifiers. Note that the kappa-error shape depends only on the base classifiers (see Discussion). The top row also visualizes the partial convex hull (black line) and the Pareto frontier (red line). Symbols refer to the pruning method. Each dot is a classifier pair trained on two encoded datasets. Note that we display only 1000 dots per panel (top row). Moreover, we set the bin size to 40 for the binned heatmap with darker colors depicting more values (bottom row).

XCD chart

The extended critical difference (XCD) chart (Fig. 2) is based on the critical difference chart introduced by Calvo and Santafé (2016) [32]. Classifier groups not surpassing the critical difference (CD) are connected with black lines. The line thickness depicts the actual CD, meaning groups associated with thicker lines are closer to CD. The XCD charts present two classifier groups. The x-axis includes pruning types, and the y-axis the actual ensembles and the corresponding base classifier. The main area contains a categorical heatmap showing Matthews correlation coefficient (MCC) in 0.05 steps. The darker, the higher the MCC. The MCC is the median MCC of the respective group combination and corresponds to the median from Fig. 3. Note that for the computation of the CD, we concatenated the MCCs of all cross-validation runs, e.g., 12 * 100 MCCs for pfront, and 6 * 100 MCCs for bayes_voting_soft.

Funding
This work was financially supported by the BMWi in the project MoDiPro-ISOB (16KN0742325). This work was also supported by the BMBF-funded de.NBI Cloud within the German Network for Bioinformatics Infrastructure (de.NBI) (031A532B, 031A533A, 031A533B, 031A534A, 031A535A, 031A537A, 031A537B, 031A537C, 031A537D, 031A538A).

Availability of data and materials
The source code can be found at https://github.com/spaenigs/ensemble-performance. All datasets are available at https://github.com/spaenigs/peptidereactor.

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Consent for publication
Not applicable.

Authors' contributions
SS and DH developed the concept. SS designed and performed the experiments and analyzed the data. SS and DH interpreted the results. AM implemented the MVO algorithm. SS wrote the manuscript. DH supervised the study and revised the manuscript. All authors read and approved the final manuscript.

Author details
Data Science in Biomedicine, Department of Mathematics and Computer Science, University of Marburg, Marburg, Germany.

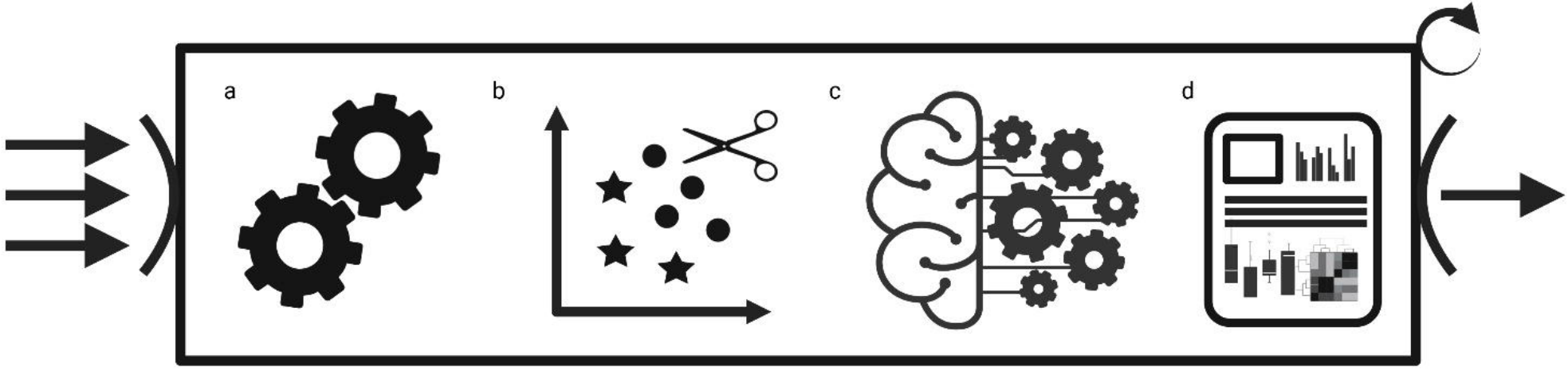
References

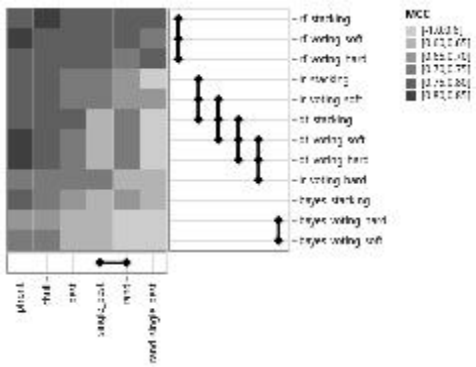
1. Holmes, A.H., Moore, L.S.P., Sundsfjord, A., Steinbakk, M., Regmi, S., Karkey, A., Guerin, P.J., Piddock, L.J.V.: Understanding the mechanisms and drivers of antimicrobial resistance. The Lancet **387**(10014), 176–187 (2016). doi:10.1016/S0140-6736(15)00473-0
2. Spänig, S., Eick, L., Nuy, J.K., Beisser, D., Ip, M., Heider, D., Boenigk, J.: A multi-omics study on quantifying antimicrobial resistance in European freshwater lakes. Environment International **157**, 106821 (2021). doi:10.1016/j.envint.2021.106821
3. Kakkar, M., Walia, K., Vong, S., Chatterjee, P., Sharma, A.: Antibiotic resistance and its containment in India. BMJ (Online) **358**, 25–30 (2017). doi:10.1136/bmj.j2687
4. Qu, J., Huang, Y., Lv, X.: Crisis of antimicrobial resistance in China: Now and the future. Frontiers in Microbiology **10**(SEP) (2019). doi:10.3389/fmicb.2019.02240
5. Lazzaro, B.P., Zasloff, M., Rolff, J.: Antimicrobial peptides: Application informed by evolution. Science **368**(6490) (2020). doi:10.1126/science.aau5480
6. Magana, M., Pushpanathan, M., Santos, A.L., Leanse, L., Fernandez, M., Ioannidis, A., Giulianotti, M.A., Apidianakis, Y., Bradfute, S., Ferguson, A.L., Cherkasov, A., Seleem, M.N., Pinilla, C., de la Fuente-Nunez, C., Lazaridis, T., Dai, T., Houghten, R.A., Hancock, R.E.W., Tegos, G.P.: The value of antimicrobial peptides in the age of resistance. The Lancet Infectious Diseases **20**(9), 216–230 (2020). doi:10.1016/S1473-3099(20)30327-3
7. Waghu, F.H., Idicula-Thomas, S.: Collection of antimicrobial peptides database and its derivatives: Applications and beyond. Protein Science **29**(1), 36–42 (2020). doi:10.1002/pro.3714
8. Chung, C.R., Kuo, T.R., Wu, L.C., Lee, T.Y., Horng, J.T.: Characterization and identification of antimicrobial peptides with different functional activities. Briefings in Bioinformatics **21**(3), 1098–1114 (2020). doi:10.1093/bib/bbz043
9. Dean, S.N., Walper, S.A.: Variational autoencoder for generation of antimicrobial peptides. ACS Omega **5**(33), 20746–20754 (2020). doi:10.1021/acsomega.0c00442
10. Fingerhut, L.C.H.W., Miller, D.J., Strugnell, J.M., Daly, N.L., Cooke, I.R.: ampir: an R package for fast genome-wide prediction of antimicrobial peptides. Bioinformatics **36**(21), 5262–5263 (2020). doi:10.1093/bioinformatics/btaa653/5873588
11. Aronica, P.G.A., Reid, L.M., Desai, N., Li, J., Fox, S.J., Yadahalli, S., Essex, J.W., Verma, C.S.: Computational Methods and Tools in Antimicrobial Peptide Research. Journal of Chemical Information and Modeling, 1–00175 (2021). doi:10.1021/acs.jcim.1c00175
12. Manavalan, B., Basith, S., Shin, T.H., Choi, S., Kim, M.O., Lee, G.: MLACP: machine-learning-based prediction of anticancer peptides. Oncotarget **8**(44), 77121–77136 (2017)
13. Manavalan, B., Subramaniyam, S., Shin, T.H., Kim, M.O., Lee, G.: Machine-Learning-Based Prediction of Cell-Penetrating Peptides and Their Uptake Efficiency with Improved Accuracy. Journal of Proteome Research **17**(8), 2715–2726 (2018). doi:10.1021/acs.jproteome.8b00148
14. Gupta, S., Sharma, A.K., Shastri, V., Madhu, M.K., Sharma, V.K.: Prediction of anti-inflammatory proteins/peptides: An insilico approach. Journal of Translational Medicine **15**(1) (2017). doi:10.1186/s12967-016-1103-6
15. Manavalan, B., Shin, T.H., Kim, M.O., Lee, G.: PIP-EL: A New Ensemble Learning Method for Improved Proinflammatory Peptide Predictions. Frontiers in immunology **9**, 1783 (2018). doi:10.3389/fimmu.2018.01783
16. Meher, P.K., Sahu, T.K., Saini, V., Rao, A.R.: Predicting antimicrobial peptides with improved accuracy by incorporating the compositional, physico-chemical and structural features into Chou's general PseAAC. Scientific Reports **7** (2017). doi:10.1038/srep42362

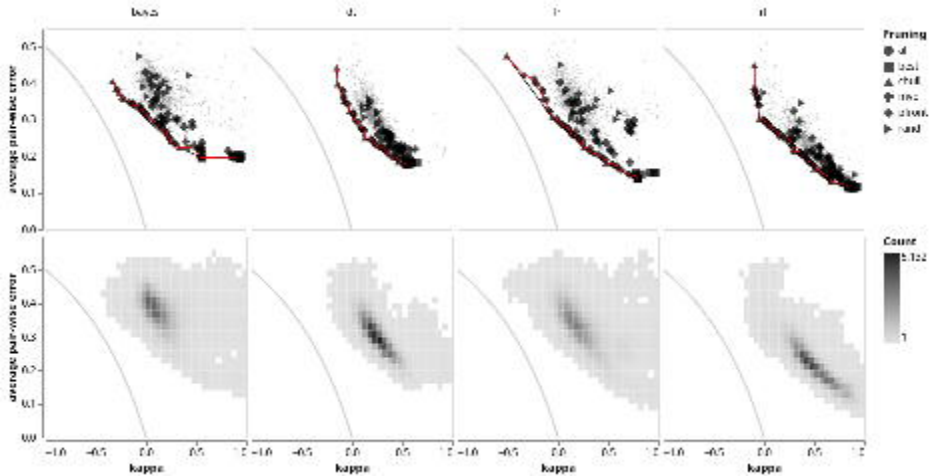
17. Chaudhary, K., Kumar, R., Singh, S., Tuknait, A., Gautam, A., Mathur, D., Anand, P., Varshney, G.C., Raghava, G.P.S.: A web server and mobile app for computing hemolytic potency of peptides. Scientific Reports **6** (2016). doi:10.1038/srep22843
18. Agrawal, P., Kumar, S., Singh, A., Raghava, G.P.S., Singh, I.K.: NeuroPIpred: a tool to predict, design and scan insect neuropeptides. Scientific Reports **9**(1) (2019). doi:10.1038/s41598-019-41538-x
19. Spänig, S., Heider, D.: Encodings and models for antimicrobial peptide classification for multi-resistant pathogens. BioData Mining **12**(1), 1–29 (2019). doi:10.1186/s13040-019-0196-x
20. Spänig, S., Mohsen, S., Hattab, G., Hauschild, A.-C., Heider, D.: A large-scale comparative study on peptide encodings for biomedical classification. NAR Genomics and Bioinformatics **3**(2) (2021). doi:10.1093/nargab/lqab039
21. Khatun, M.S., Hasan, M.M., Shoombuatong, W., Kurata, H.: ProIn-Fuse: improved and robust prediction of proinflammatory peptides by fusing of multiple feature representations. Journal of Computer-Aided Molecular Design **34**(12), 1229–1236 (2020). doi:10.1007/s10822-020-00343-9
22. Plisson, F., Ramírez-Sánchez, O., Martínez-Hernández, C.: Machine learning-guided discovery and design of non-hemolytic peptides. Scientific Reports **10**(1) (2020). doi:10.1038/s41598-020-73644-6
23. Timmons, P.B., Hewage, C.M.: HAPPENN is a novel tool for hemolytic activity prediction for therapeutic peptides which employs neural networks. Scientific Reports **10**(1) (2020). doi:10.1038/s41598-020-67701-3
24. Singh, O., Hsu, W.-L., Su, E.C.-Y.: Co-AMPpred for in silico-aided predictions of antimicrobial peptides by integrating composition-based features. BMC Bioinformatics **22**(1), 389 (2021). doi:10.1186/s12859-021-04305-2
25. Kuncheva, L.I.: Combining Pattern Classifiers, (2014). doi:10.1002/9781118914564
26. Chen, Z., Zhao, P., Li, C., Li, F., Xiang, D., Chen, Y.-Z., Akutsu, T., Daly, R.J., Webb, G.I., Zhao, Q., Kurgan, L., Song, J.: iLearnPlus: a comprehensive and automated machine-learning platform for nucleic acid and protein sequence analysis, prediction and visualization. Nucleic Acids Research (2021). doi:10.1093/nar/gkab122
27. Schwarz, J., Heider, D.: GUESS: projecting machine learning scores to well-calibrated probability estimates for clinical decision-making. Bioinformatics **35**(14), 2458–2465 (2019). doi:10.1093/bioinformatics/bty984
28. Heider, D., Dybowski, J.N., Wilms, C., Hoffmann, D.: A simple structure-based model for the prediction of HIV-1 co-receptor tropism. BioData Mining **7**(1) (2014)
29. Löchel, H.F., Riemenschneider, M., Frishman, D., Heider, D.: SCOTCH: subtype a coreceptor tropism classification in HIV-1. Bioinformatics **34**(15), 2575–2580 (2018)
30. Kuncheva, L.I., Jain, L.C.: Designing classifier fusion systems by genetic algorithms. In: IEEE Transactions on Evolutionary Computation, vol. 4 (2000)
31. Touw, W.G., Bayjanov, J.R., Overmars, L., Backus, L., Boekhorst, J., Wels, M., van Hijum, A.F.T.S.: Data mining in the life science swith random forest: A walk in the park or lost in the jungle? Briefings in Bioinformatics **14**, 315–326 (2013). doi:10.1093/bib/bbs034
32. Calvo, B., Santafé, G.: scmamp: Statistical Comparison of Multiple Algorithms in Multiple Problems. The R Journal **8**(1), 248–256 (2016)
33. Zhu, J., Zou, H., Rosset, S., Hastie, T.: Multi-class AdaBoost. Statistics and Its Interface **2**, 349–360 (2009)
34. Kuncheva, L.I.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning **51**, 181–207 (2003)
35. Mirjalili, S., Mirjalili, S.M., Hatamlou, A.: Multi-Verse Optimizer: a nature-inspired algorithm for global optimization. Neural Computing and Applications **27**(2), 495–513 (2016). doi:10.1007/s00521-015-1870-7
36. Kuncheva, L.I.: A bound on kappa-error diagrams for analysis of classifier ensembles. IEEE Transactions on Knowledge and Data Engineering **25**(3), 494–501 (2013). doi:10.1109/TKDE.2011.234
37. Köster, J., Rahmann, S.: Snakemake-a scalable bioinformatics workflow engine. Bioinformatics **28**(19), 2520–2522 (2012). doi:10.1093/bioinformatics/bts480
38. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passps, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2015). doi:10.1145/2786984.2786995
39. Chicco, D., Jurman, G.: The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics **21**(1) (2020). doi:10.1186/s12864-019-6413-7
40. Su, X., Xu, J., Yin, Y., Quan, X., Zhang, H.: Antimicrobial peptide identification using multi-scale convolutional network. BMC Bioinformatics **20**(1) (2019). doi:10.1186/s12859-019-3327-y
41. Usmani, S.S., Bhalla, S., Raghava, G.P.S.: Prediction of antitubercular peptides from sequence information using ensemble classifier and hybrid features. Frontiers in Pharmacology **9**(AUG), 1–11 (2018). doi:10.3389/fphar.2018.00954
42. Nagpal, G., Usmani, S.S., Dhanda, S.K., Kaur, H., Singh, S., Sharma, M., Raghava, G.P.S.: Computer-aided designing of immunosuppressive peptides based on IL-10 inducing potential. Scientific Reports **7** (2017). doi:10.1038/srep42851
43. Xu, Q.-S., Liang, Y.-Z.: Monte Carlo cross validation. Chemometrics and Intelligent Laboratory Systems **56**, 1–11 (2000)
44. Ren, J., Lee, S.D., Chen, X., Kao, B., Cheng, R., Cheung, D.: Naive bayes classification of uncertain data. In: Proceedings - IEEE International Conference on Data Mining, ICDM, pp. 944–949 (2009). doi:10.1109/ICDM.2009.90
45. James, G., Witten, D., Hastie, T., Tibshirani, R.: An Introduction to Statistical Learning, 8th edn. Springer, New York (2017)
46. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification And Regression Trees. Routledge, London (1984). doi:10.1201/9781315139470
47. Breiman, L.: Random Forests. Machine Learning **45**, 5–32 (2001)
48. Wolpert, D.H.: Stacked Generalization. Neural Networks **5**, 241–259 (1992)
49. Al-Madi, N., Faris, H., Mirjalili, S.: Binary multi-verse optimization algorithm for global optimization and

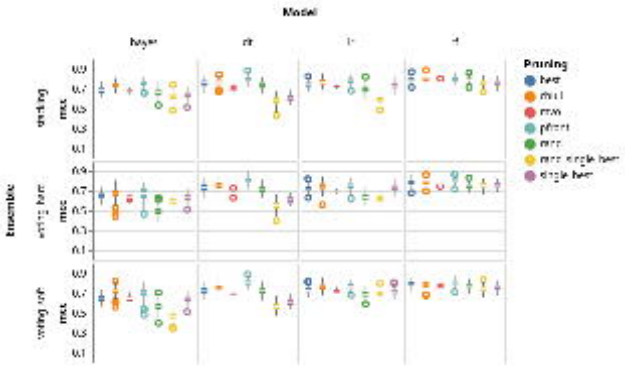
discrete problems. International Journal of Machine Learning and Cybernetics **10**(12), 3445–3465 (2019). doi:10.1007/s13042-019-00931-8

50. Margineantu, D.D., Dietterich, T.G.: Pruning Adaptive Boosting. In: ICML (1997)
51. Barber, C.B., Dobkin, D.P., Huhdanpaa, H.: The Quickhull Algorithm for Convex Hull. ACM Transactions on Mathematical Software **22**(4), 469–483 (1996)
52. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods **17**, 261–272 (2020). doi:10.1038/s41592-019-0686-2
53. Messac, A., Ismail-Yahaya, A., Mattson, C.A.: The normalized normal constraint method for generating the Pareto frontier. Structural and Multidisciplinary Optimization **25**(2), 86–98 (2003). doi:10.1007/s00158-002-0276-1
54. Santafe, G., Inza, I., Lozano, J.A.: Dealing with the evaluation of supervised classification algorithms. Artificial Intelligence Review **44**(4), 467–508 (2015). doi:10.1007/s10462-015-9433-y
55. VanderPlas, J., Granger, B., Heer, J., Moritz, D., Wongsuphasawat, K., Satyanarayan, A., Lees, E., Timofeev, I., Welsh, B., Sievert, S.: Altair: Interactive Statistical Visualizations for Python. Journal of Open Source Software **3**(32), 1057 (2018). doi:10.21105/joss.01057


a
b
c
d
Created with BioRender.com


NCC
if standing
if working soft
if working hard
in standing
in working soft
st standing
st working soft
st working hard
in working hard
hayes standing
hayes working hard
hayes working soft


average pairwise error
kappa
Pruning
Count
5,122
1


Model
bayes
rf
l
rl
Ensemble
stacking
mse
0.9
0.7
0.5
0.3
0.1
voting hard
voting soft
Pruning
best
rfrl
rmse
pfirst
rand
rand_single_best
single_best