Data-driven advice for applying machine learning to bioinformatics problems

Randal S. Olson[†*], William La Cava[†*], Zairah Mustahsan, Akshay Varik, and Jason H. Moore[†]

Institute for Biomedical Informatics, University of Pennsylvania
Philadelphia, PA 19104, USA
[†]*E-mails: rso@randalolson.com, lacava@upenn.edu and jhmoore@upenn.edu*

As the bioinformatics field grows, it must keep pace not only with new data but with new algorithms. Here we contribute a thorough analysis of 13 state-of-the-art, commonly used machine learning algorithms on a set of 165 publicly available classification problems in order to provide data-driven algorithm recommendations to current researchers. We present a number of statistical and visual comparisons of algorithm performance and quantify the effect of model selection and algorithm tuning for each algorithm and dataset. The analysis culminates in the recommendation of five algorithms with hyperparameters that maximize classifier performance across the tested problems, as well as general guidelines for applying machine learning to supervised classification problems.

Keywords: machine learning; data science; best practices; benchmarking; bioinformatics

1. Introduction

The bioinformatics field is increasingly relying on machine learning (ML) algorithms to conduct predictive analytics and gain greater insights into the complex biological processes of the human body.[1] For example, ML algorithms have been applied to great success in GWAS, and have proven effective at detecting patterns of epistasis within the human genome.[2] Recently, deep learning algorithms were used to detect cancer metastases on high-resolution pathology images[3] at levels comparable to human pathologists. These results, among others, indicate heavy interest in ML development and analysis for bioinformatics applications.

Owing to the development of open source ML packages and active research in the ML field, researchers can easily choose from dozens of ML algorithm implementations to build predictive models of complex data. Although having several readily-available ML algorithm implementations is advantageous to bioinformatics researchers seeking to move beyond simple statistics, many researchers experience "choice overload" and find difficulty in selecting the right ML algorithm for their problem at hand. As a result, some ML-oriented bioinformatics projects could be improved simply through the use of a better ML algorithm.

ML researchers are aware of the challenges that algorithm selection presents to ML practitioners. As a result, there have been some efforts to empirically assesses different algorithms across sets of problems, beginning in the mid 1990s with the StatLog project.[4] Early work in this field also emphasized bioinformatics applications.[5] More recently, Caruana *et al.*[6] and

[*] Contributed equally

Fernández-Delgado *et al.*[7] analyzed several supervised learning algorithms, coupled with some parameter tuning. The aforementioned literature often compared many algorithms but on relatively few example problems (between 4 and 12), with only [7] using upwards of 112 example problems. In the time since these assessments, researchers have moved towards standardized, open source implementations of ML algorithms (e.g. scikit-learn[8] and Weka[9]), and the number of publicly available datasets that can be used for comparison have skyrocketed, leading to the creation of decentralized, collaboration-based analyses such as the OpenML project.[10] However, the value of focused, reproducible ML experiments is still paramount. These observations motivated our work, in which we conduct a contemporary, open source, and thorough comparison of ML algorithms across a large set of publicly available problems, including several bioinformatics problems.

In this paper, we take a detailed look at 13 popular open source ML algorithms and analyze their performance across a set of 165 supervised classification problems in order to provide data-driven advice to practitioners who wish to apply ML to their datasets. A key part of this comparison is a full hyperparameter optimization of each algorithm. The results highlight the importance of selecting the right ML algorithm for each problem, which can improve prediction accuracy significantly on some problems. Further, we empirically quantify the effect of hyperparameter (i.e. algorithm parameter) tuning for each ML algorithm, demonstrating marked improvements in the predictive accuracy of nearly all ML algorithms. We show that the underlying behaviors of various ML algorithms cluster in terms of performance, as might be expected. Finally, based on the results of the experiments, we provide a refined set of recommendations for ML algorithms and parameters as a starting point for future researchers.

2. Methods

In this study, we compared 13 popular ML algorithms from scikit-learn,[8] a widely used ML library implemented in Python. Each algorithm and its hyperparameters are described in Table 1. The algorithms include Naïve Bayes algorithms, common linear classifiers, tree-based algorithms, distance-based classifiers, ensemble algorithms, and non-linear, kernel-based strategies. The goal was to represent the most common classes of algorithms used in literature, as well as recent state-of-the-art algorithms such as Gradient Tree Boosting.[11]

For each algorithm, the hyperparameters were tuned using a fixed grid search with 10-fold cross-validation. In our results, we compare the average balanced accuracy[12] over the 10 folds in order to account for class imbalance. We used expert knowledge about the reasonable hyperparameters to specify the ranges of values to tune for each algorithm. It is worth noting that we did not attempt to control for the *number* of total hyperparameter combinations budgeted to each algorithm. As a result, algorithms with more parameters have an advantage in the sense that they have more training attempts on each dataset. However, it is our goal to report as close to the best performance as possible for each algorithm on each dataset, and for this reason we chose to optimize each algorithm as thoroughly as possible.

The algorithms were compared on 165 supervised classification datasets from the Penn Machine Learning Benchmark (PMLB).[13] PMLB is a collection of publicly available classification problems that have been standardized to the same format and collected in a central location

with easy access via Python[a]. Although not limited to problems in biology and medicine, PMLB includes many biomedical classification problems, including tasks such as disease diagnosis, post-operative decision making, and exon boundary identification in DNA, among others. A sample of the biomedical classification tasks contained in PMLB is listed in Table 2.

Prior to evaluating each ML algorithm, we scaled the features of every dataset by subtracting the mean and scaling the features to unit variance. This scaling step was necessitated by some ML algorithms, such as the distance-based classifiers, which assume that the features of the datasets will be scaled appropriately beforehand.

The entire experimental design consisted of over 5.5 million ML algorithm and parameter evaluations in total, resulting in a rich set of data that is analyzed from several viewpoints in Section 3. As an additional contribution of this work, we have provided the complete code required both to conduct the algorithm and hyperparameter optimization study, as well as access to the analysis and results[b]. Doing so allows researchers to easily compare algorithm performance on the datasets that are most similar to their own, and to conduct further analysis pertaining to their research.

3. Results

In this section, we analyze the algorithm performance results through several lenses. First we compare the performance of each algorithm across all datasets in terms of best balanced accuracy in Section 3.1. We then look at the effect of hyperparameter tuning and model selection in Section 3.2. Finally, we analyze how algorithms cluster across the tested problems, and present a set of algorithms that maximize performance across the datasets in Section 3.3.

3.1. *Algorithm Performance*

As a simple bulk measure to compare the performance of the 13 ML algorithms, we plot the mean rankings of the algorithms across all datasets in Figure 1. Ranking is determined by the 10-fold CV balanced accuracy of each algorithm on a given dataset, with a lower ranking indicating higher accuracy. The rankings show the strength of ensemble-based tree algorithms in generating accurate models: The first, second, and fourth-ranked algorithms belong to this class of algorithms. The three worst-ranked algorithms also belong to the same class of Naïve Bayes algorithms.

In order to assess the statistical significance of the observed differences in algorithm performance across all problems, we use the non-parametric Friedman test.[14] The complete set of experiments indicate statistically significant differences according to this test ($p < 2.2e^{-16}$), and so we present a pairwise post-hoc analysis in Table 3. The post-hoc test underlines the impressive performance of Gradient Tree Boosting, which significantly outperforms every algorithm except Random Forest at the $p < 0.01$ level. At the other end of the spectrum, Multinomial NB is significantly outperformed by every algorithm except for Gaussian NB. These strong statistical results are interesting given the large set of problems and algorithms

[a]URL: `https://github.com/EpistasisLab/penn-ml-benchmarks`
[b]URL: `https://github.com/rhiever/sklearn-benchmarks`

Table 1. ML algorithms and hyperparameters tuned in the experiments.

Algorithm	**Hyperparameters**
Gaussian Naïve Bayes (GNB)	No parameters.
Bernoulli Naïve Bayes (BNB)	**alpha**: Additive smoothing parameter. **binarize**: Threshold for binarizing the features. **fit_prior**: Whether or not to learn class prior probabilities.
Multinomial Naïve Bayes (MNB)	**alpha**: Additive smoothing parameter. **fit_prior**: Whether or not to learn class prior probabilities.
Logistic Regression (LR)	**C**: Regularization strength. **penalty**: Whether to use Lasso or Ridge regularization. **fit_intercept**: Whether or not the intercept of the linear classifier should be computed.
Stochastic Gradient Descent (SGD)	**loss**: Loss function to be optimized. **penalty**: Whether to use Lasso, Ridge, or ElasticNet regularization. **alpha**: Regularization strength. **learning_rate**: Shrinks the contribution of each successive training update. **fit_intercept**: Whether or not the intercept of the linear classifier should be computed. **l1_ratio**: Ratio of Lasso vs. Ridge reguarlization to use. Only used when the 'penalty' is ElasticNet. **eta0**: Initial learning rate. **power_t**: Exponent for inverse scaling of the learning rate.
Passive Aggressive Classifier (PAC)	**loss**: Loss function to be optimized. **C**: Maximum step size for regularization. **fit_intercept**: Whether or not the intercept of the linear classifier should be computed.
Support Vector Classifier (SVC)	**kernel**: 'linear', 'poly', 'sigmoid', or 'rbf'. **C**: Penalty parameter for regularization. **gamma**: Kernel coef. for 'rbf', 'poly' & 'sigmoid' kernels. **degree**: Degree for the 'poly' kernel. **coef0**: Independent term in the 'poly' and 'sigmoid' kernels.
K-Nearest Neighbor (KNN)	**n_neighbors**: Number of neighbors to use. **weights**: Function to weight the neighbors' votes.
Decision Tree (DT)	**min_weight_fraction_leaf**: The minimum number of (weighted) samples for a node to be considered a leaf. Controls the depth and complexity of the decision tree. **max_features**: Number of features to consider when computing the best node split. **criterion**: Function used to measure the quality of a split.
Random Forest (RF) & Extra Trees Classifier (ERF)	**n_estimators**: Number of decision trees in the ensemble. **min_weight_fraction_leaf**: The minimum number of (weighted) samples for a node to be considered a leaf. Controls the depth and complexity of the decision trees. **max_features**: Number of features to consider when computing the best node split. **criterion**: Function used to measure the quality of a split.
AdaBoost (AB)	**n_estimators**: Number of decision trees in the ensemble. **learning_rate**: Shrinks the contribution of each successive decision tree in the ensemble.
Gradient Tree Boosting (GTB)	**n_estimators**: Number of decision trees in the ensemble. **learning_rate**: Shrinks the contribution of each successive decision tree in the ensemble. **loss**: Loss function to be optimized via gradient boosting. **max_depth**: Maximum depth of the decision trees. Controls the complexity of the decision trees. **max_features**: Number of features to consider when computing the best node split.

compared here. Because the No Free Lunch theorem[15] guarantees that all algorithms perform the same on average over all possible classes of problems, the differentiated results imply that the problems in the PMLB belong to a related subset of classes. The initial PMLB study[13] also noted the similarity in properties of several publicly available datasets, which could lead

Table 2. A non-exhaustive sample of datasets included in the PMLB archive that pertain to biomedical classification.

Data Set	Classes	Samples	Dimensions	Description
allbp	3	3772	29	Diagnosis
allhyper	4	3771	29	Diagnosis
allhypo	3	3770	29	Diagnosis
ann-thyroid	3	7200	21	Diagnosis
biomed	2	209	8	Diagnosis
breast-cancer-wisconsin	2	569	30	Diagnosis
breast-cancer	2	286	9	Diagnosis
diabetes	2	768	8	Diagnosis
dna	3	3186	180	Locating exon boundaries
GMT 2w-20a-0.1n	2	1600	20	Simulated GWAS
GMT 2w-1000a-0.4n	2	1600	1000	Simulated GWAS
liver-disorder	2	345	6	Diagnosis
molecular-biology_promoters	2	106	58	Identify promoter sequences
postoperative-patient-data	2	88	8	Choose post-operative treatment

to inflated statistical significance. Nevertheless, it cannot be denied that the results are relevant to classification tasks encountered in real-world and biological contexts, since the vast majority of datasets used here are taken from those contexts.

Given these bulk results, it is tempting to recommend the top-ranked algorithm for all problems. However, this neglects the fact that the top-ranked algorithms may not outperform others for some problems. Furthermore, when simpler algorithms perform on par with a more complex one, it is often preferable to choose the simpler of the two. With this in mind, we investigate pair-wise "outperformance" by calculating the percentage of datasets for which one algorithm outperforms another, shown in Figure 2. One algorithm outperforms another on a dataset if it has at least a 1% higher 10-fold CV balanced accuracy, which represents a minimal threshold for improvement in predictive accuracy.

In terms of "outperformance," it is worth noting that no one ML algorithm performs best across all 165 datasets. For example, there are 9 datasets for which Multinomial NB performs as well as or better than Gradient Tree Boosting, despite being the overall worst- and best-ranked algorithms, respectively. Therefore, it is still important to consider different ML algorithms when applying ML to new datasets.

3.2. *Effect of Tuning and Model Selection*

Most ML algorithms contain several hyperparameters that can affect performance significantly (for example, the max tree depth of a decision tree classifier). Our experimental results allow us to measure the extent to which hyperparameter tuning via grid search improves each algorithm's performance compared to its baseline settings. We also measure the effect that model selection has on improving classifier performance.

Figure 3 compares the performance of the tuned classifier to its default settings for each algorithm across all datasets. The results demonstrate why it is unwise to use default ML al-

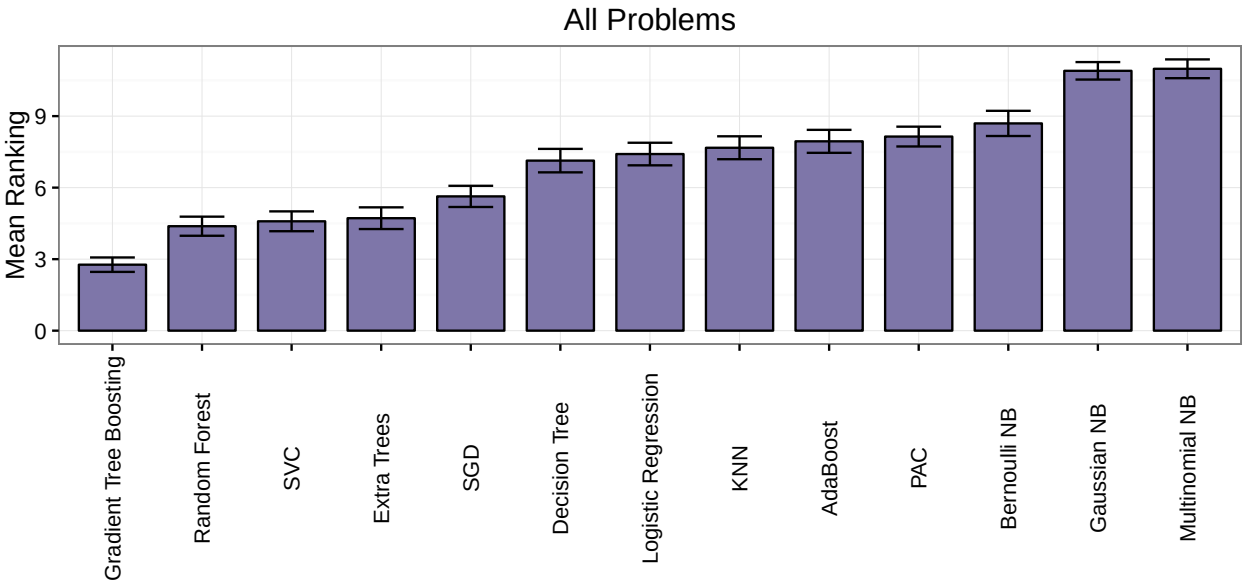

Fig. 1. Average ranking of the ML algorithms over all datasets. Error bars indicate the 95% confidence interval.

Table 3. Post-hoc Friedman test of algorithm rankings across all problems. Bold values indicate $p < 0.01$.

	GTB	RF	SVC	ERF	SGD	DT	LR	KNN	AB	PAC	BNB	GNB
RF	0.01	-	-	-	-	-	-	-	-	-	-	-
SVC	**0.001**	1	-	-	-	-	-	-	-	-	-	-
ERF	**0.0004**	1	1	-	-	-	-	-	-	-	-	-
SGD	**3e-10**	0.1	0.4	0.6	-	-	-	-	-	-	-	-
DT	**0**	**3e-09**	**1e-07**	**3e-07**	0.03	-	-	-	-	-	-	-
LR	**0**	**1e-11**	**1e-09**	**1e-07**	**0.003**	1	-	-	-	-	-	-
KNN	**0**	**1e-13**	**5e-12**	**7e-11**	**0.0002**	1	1	-	-	-	-	-
AB	**0**	**6e-15**	**4e-14**	**4e-13**	**3e-06**	0.8	1	1	-	-	-	-
PAC	**0**	**2e-16**	**3e-15**	**8e-15**	**2e-07**	0.5	0.9	1	1	-	-	-
BNB	**0**	**0**	**0**	**0**	**4e-10**	0.02	0.1	0.4	0.9	1	-	-
GNB	**0**	**0**	**0**	**0**	**0**	**0**	**2e-15**	**9e-13**	**1e-10**	**5e-09**	**2e-05**	-
MNB	**0**	**0**	**0**	**0**	**0**	**0**	**2e-15**	**7e-14**	**1e-11**	**4e-09**	**4e-06**	1

gorithm hyperparameters: tuning often improves an algorithm's accuracy by 3-5%, depending on the algorithm. In some cases, parameter tuning led to CV accuracy improvements of 50%.

Figure 4 shows the improvement in 10-fold CV accuracy attained both by model selection and hyperparameter optimization compared to the average performance on each dataset. The results demonstrate that selecting the best model and tuning it leads to approximately a 20% increase in accuracy, up to more than a 60% improvement for certain datasets. Thus, both selecting the right ML algorithm and tuning its parameters is vitally important for most problems.

3.3. *Algorithm Coverage*

Given that several of the 13 algorithms studied here have similar underlying methodologies, we would expect their performance across problems to align with the underlying assumptions

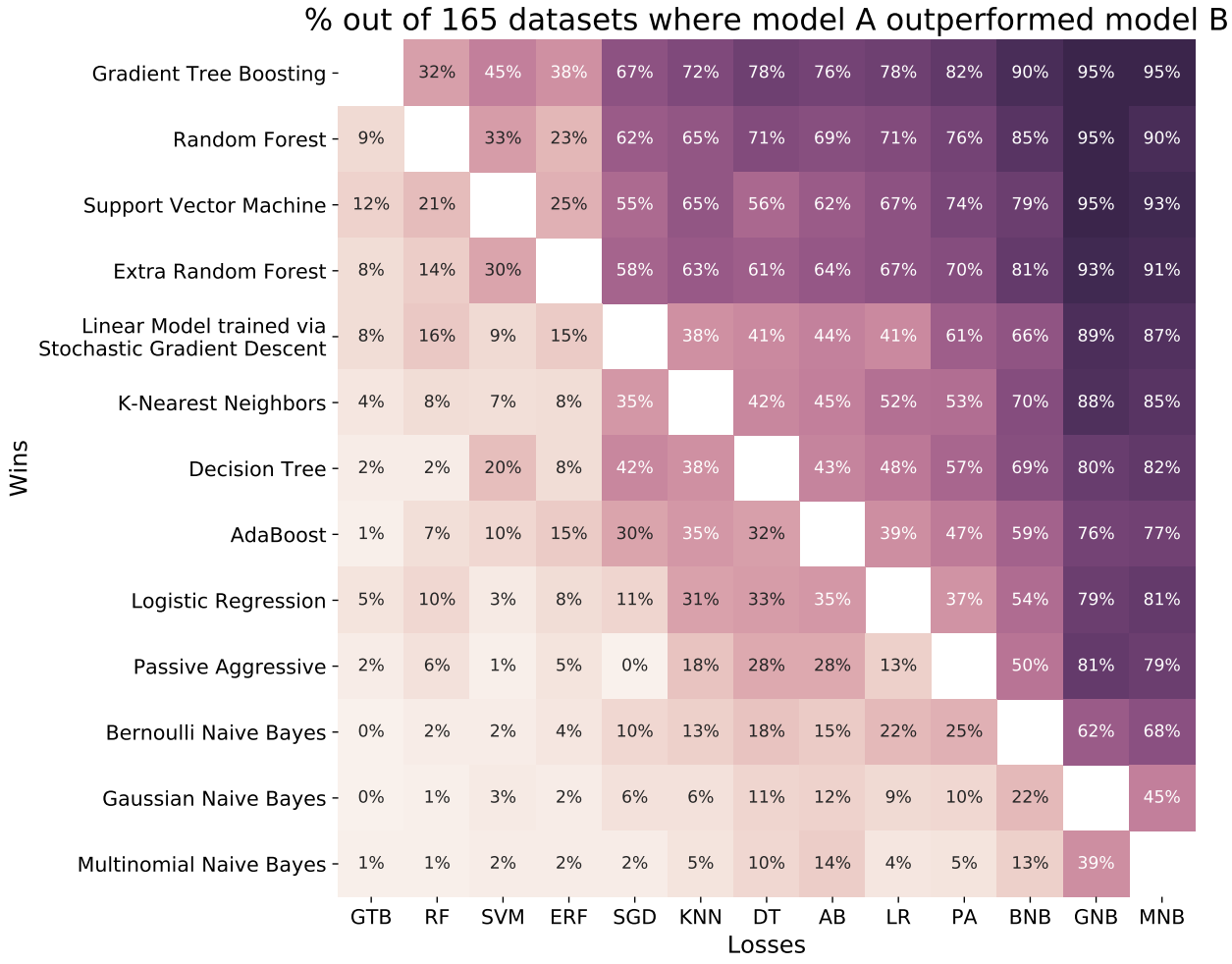

Fig. 2. Heat map showing the percentage out of 165 datasets a given algorithm outperforms another algorithm in terms of best accuracy on a problem. The algorithms are ordered from top to bottom based on their overall performance on all problems. Two algorithms are considered to have the same performance on a problem if they achieved an accuracy within 1% of each other.

that the modeling techniques have in common. One way to assess whether this holds is to cluster the performance of different algorithms across all datasets. We perform hierarchical agglomerative clustering on the 10-fold CV balanced accuracy results, which leads to the clusters shown in Figure 5. Indeed, we find that algorithms with similar underlying assumptions or methodologies cluster in terms of their performance across the datasets. For example, the Naïve Bayes algorithms (i.e., Multinomial, Gaussian, and Bernoulli) perform most similarly to each other, and the linear algorithms (i.e., passive aggressive and logistic regression) also cluster. The ensemble algorithms of Extra Trees and Random Forests, which both use ensembles of decision trees, also cluster. Support Vector Machines and Gradient Tree Boosting appear to be quite different algorithms, but given that both are able to capture nonlinear interactions between variables, it is less surprising that they cluster as well.

We present a list of five recommended algorithms and parameter settings in Table 4.

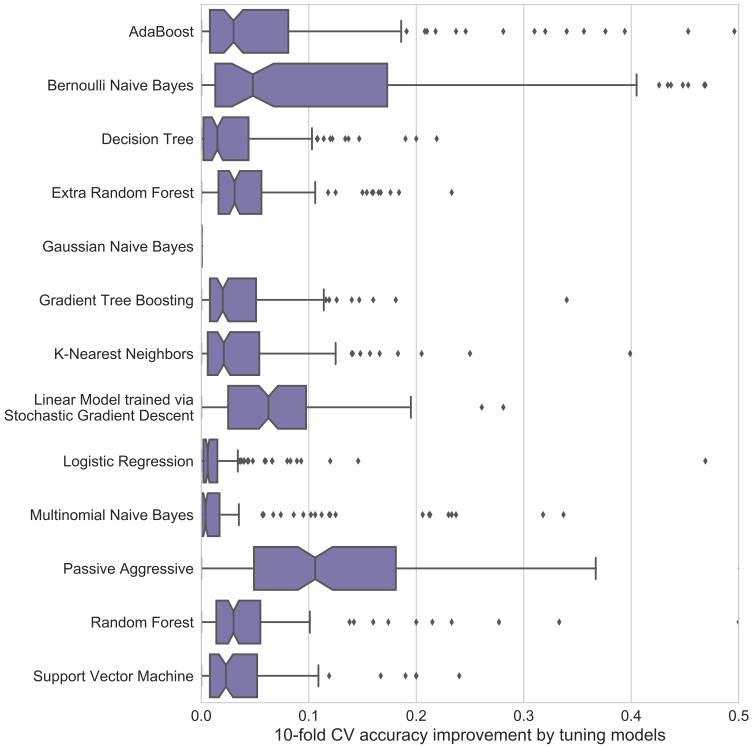

Fig. 3. Improvement in 10-fold CV accuracy by tuning each ML algorithm's parameters instead of using the default parameters from scikit-learn.

The five algorithms and parameters here are those that maximize the coverage of the 165 benchmark datasets, meaning that they perform within 1% of the best 10-fold CV balanced accuracy obtained on the maximum number of datasets in the experiment. For the datasets in PMLB, these five algorithms and associated parameters cover 106 out of 165 datasets to within 1% balanced accuracy. Notably, 163 out of 165 datasets can be covered by tuning the parameters of the five listed algorithms. Based on the available evidence, these recommended algorithms should be a good starting point for achieving reasonable predictive accuracy on a new dataset.

4. Discussion and Conclusions

We have empirically assessed 13 supervised classification algorithms on a set of 165 supervised classification datasets in order to provide a contemporary set of recommendations to bioinformaticians who wish to apply ML algorithms to their data. The analysis demonstrates the strength of state-of-the-art, tree-based ensemble algorithms, while also showing the problem-dependent nature of ML algorithm performance. In addition, the analysis shows that selecting the right ML algorithm and thoroughly tuning its parameters can lead to a significant improvement in predictive accuracy on most problems, and is there a critical step in every ML application. We have made the full set of experiments and results available online to encourage bioinformaticians to easily gather information most pertinent to their area of study.

Even with a large set of results, it is difficult to recommend specific algorithms or parameter

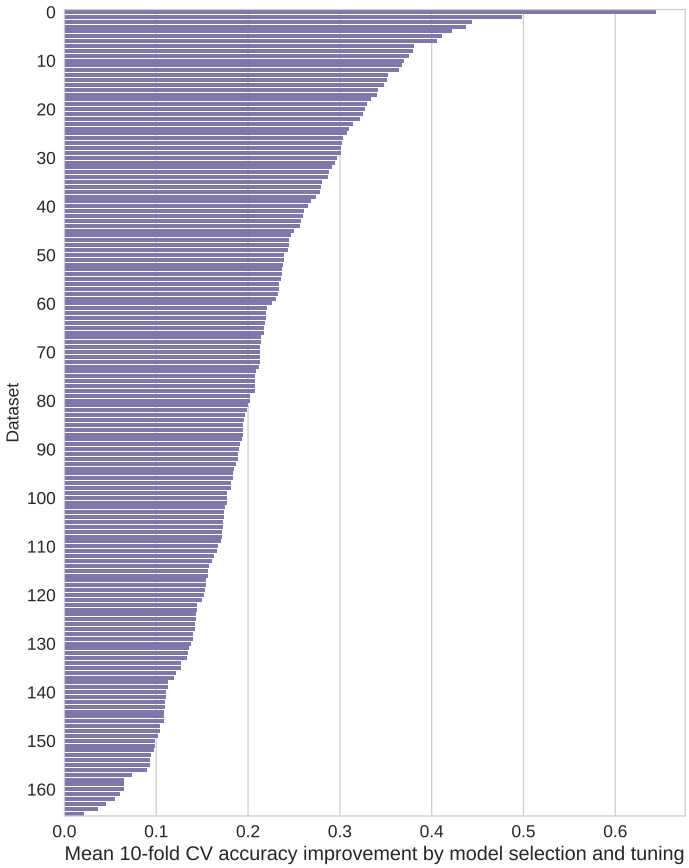

Fig. 4. Improvement in 10-fold CV accuracy by model selection and tuning, relative to the average performance on each dataset.

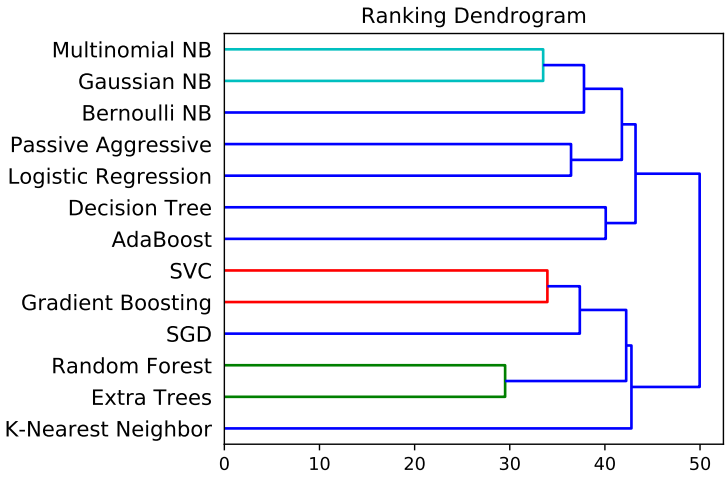

Fig. 5. Hierarchical clustering of ML algorithms by accuracy rankings across datasets.

Table 4. Five ML algorithms and parameters that maximize coverage of the 165 benchmark datasets. These algorithm and parameter names correspond to their scikit-learn implementations.

Algorithm	Parameters	Datasets Covered
GradientBoostingClassifier	loss="deviance" learning_rate=0.1 n_estimators=500 max_depth=3 max_features="log2"	51
RandomForestClassifier	n_estimators=500 max_features=0.25 criterion="entropy"	19
SVC	C=0.01 gamma=0.1 kernel="poly" degree=3 coef0=10.0	16
ExtraTreesClassifier	n_estimators=1000 max_features="log2" criterion="entropy"	12
LogisticRegression	C=1.5 penalty="l1" fit_intercept=True	8

settings with a strong amount of generality. As a starting point, we provided recommendations for 5 different ML algorithms and parameters based on their collective coverage of the 165 datasets from PMLB. However, it is important to note that these algorithms and parameters will not work best on all supervised classification problems, and they should only be used as starting points. For a more nuanced approach, the similarity of the dataset on which ML is to be applied to datasets in PMLB could be quantified, and the set of algorithms that performed best on those similar datasets could be used. In lieu of detailed problem information, one could also use automated ML tools[16,17] and AI-driven ML platforms[18] to perform model selection and parameter tuning automatically.

Of course, some bioinformaticians may value properties of ML algorithms aside from predictive accuracy. For example, ML algorithms are often used as a "microscope" to model and better understand the complex biological systems from which the data was sampled. In this use case, bioinformaticians may value the interpretability of the ML model, in which case black box predictive models that cannot be interpreted are of little use.[19] Although the logistic regression and decision tree algorithms are often outperformed by tree-based ensemble algorithms in terms of predictive accuracy (Figure 2), linear models and shallow decision trees often provide a useful trade-off between predictive accuracy and interpretability. Furthermore, methods such as LIME[19] show promise for explaining why complex, black box models make individual predictions, which can also be useful for model interpretation.

There are several opportunities to extend the analysis in this paper in future work. A natural extension should be made to regression, which is used several biomedical applications such

as quantitative trait genetics. In addition, these experiments do not take into account feature preprocessing, feature construction, and and feature selection, although it has been shown that learning better data representations can significantly improve ML performance.[20] We plan to extend this work to analyze the ability of various feature preprocessing, construction, and selection strategies to improve model performance. In addition, the experimental results contain rich information about the performance of different learning algorithms as a function of the datasets. In future work, we will take a deeper look into the properties of datasets that influence the performance of specific algorithms. By relating these dataset properties to specific areas of bioinformatics, we may be able to generate tailored recommendations for ML algorithms that work best for specific applications.

5. Acknowledgments

We thank Dr. Andreas C. Müller for his valuable input during the development of this project, as well as the Penn Medicine Academic Computing Services for the use of their computing resources. This work was supported by NIH grants P30-ES013508, AI116794, DK112217 and LM012601, as well as the Warren Center for Network and Data Science at the University of Pennsylvania.

References

1. H. Bhaskar, D. C. Hoyle and S. Singh, *Computers in Biology and Medicine* **36**, 1104 (2006), Intelligent Technologies in Medicine and Bioinformatics.
2. B. A. McKinney, D. M. Reif, M. D. Ritchie and J. H. Moore, *Applied Bioinformatics* **5**, 77 (2006).
3. Y. Liu, K. Gadepalli, M. Norouzi, G. E. Dahl, T. Kohlberger, A. Boyko, S. Venugopalan, A. Timofeev, P. Q. Nelson, G. S. Corrado, J. D. Hipp, L. Peng and M. C. Stumpe, Detecting cancer metastases on gigapixel pathology images arXiv e-print. https://arxiv.org/abs/1703.02442, (2017).
4. R. D. King, C. Feng and A. Sutherland, *Applied Artificial Intelligence an International Journal* **9**, 289 (1995).
5. A. C. Tan and D. Gilbert, An empirical comparison of supervised machine learning techniques in bioinformatics, in *Proceedings of the First Asia-Pacific Bioinformatics Conference on Bioinformatics 2003-Volume 19*, 2003.
6. R. Caruana and A. Niculescu-Mizil, An empirical comparison of supervised learning algorithms, in *Proceedings of the 23rd International Conference on Machine learning*, 2006.
7. M. Fernández-Delgado, E. Cernadas, S. Barro and D. Amorim, *Journal of Machine Learning Research* **15**, 3133 (2014).
8. 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, *Journal of Machine Learning Research* **12**, 2825 (2011).
9. E. Frank, M. Hall, L. Trigg, G. Holmes and I. H. Witten, *Bioinformatics* **20**, 2479 (2004).
10. J. Vanschoren, J. N. Van Rijn, B. Bischl and L. Torgo, *ACM SIGKDD Explorations Newsletter* **15**, 49 (2014).
11. T. J. Hastie, R. J. Tibshirani and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Springer, New York, NY, USA, 2009).
12. D. R. Velez *et al.*, *Genetic Epidemiology* **31**, 306 (2007).
13. R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz and J. H. Moore, PMLB:

A Large Benchmark Suite for Machine Learning Evaluation and Comparison arXiv e-print. https://arxiv.org/abs/1703.00512, (2017).
14. J. Demšar, *Journal of Machine Learning Research* **7**, 1 (2006).
15. D. H. Wolpert and W. G. Macready, *IEEE Transactions on Evolutionary Computation* **1**, 67 (1997).
16. R. S. Olson, N. Bartley, R. J. Urbanowicz and J. H. Moore, Evaluation of a tree-based pipeline optimization tool for automating data science, in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, 2016.
17. M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum and F. Hutter, Efficient and robust automated machine learning, in *Advances in Neural Information Processing Systems 28*, eds. C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett (Curran Associates, Inc., 2015) pp. 2962–2970.
18. R. S. Olson, M. Sipper, W. La Cava, S. Tartarone, S. Vitale, J. H. Fu, Weixuan Holmes and J. H. Moore, A system for accessible artificial intelligence arXiv e-print. https://arxiv.org/abs/1705.00594, (2017).
19. M. T. Ribeiro, S. Singh and C. Guestrin, "Why Should I Trust You?": Explaining the Predictions of Any Classifier, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16 (ACM, New York, NY, USA, 2016).
20. W. La Cava and J. H. Moore, Ensemble representation learning: an analysis of fitness and survival for wrapper-based genetic programming methods, in *Proceedings of the Genetic and Evolutionary Computation Conference 2017*, 2017.