

Automated Statistical and Machine Learning Platform for Biology Research

Rimmo Loyi Lego¹, Samantha Gauthier², and Denver Jn. Baptiste³

¹ Department of Biomedical Engineering, Charles V. Schaefer, Jr. School of Engineering and Science, Stevens Institute of Technology, Hoboken, NJ 07030, USA ² Department of Computer Science, Charles V. Schaefer, Jr. School of Engineering and Science, Stevens Institute of Technology, Hoboken, NJ 07030, USA ³ Department of Biology, Charles V. Schaefer, Jr. School of Engineering and Science, Stevens Institute of Technology, Hoboken, NJ 07030, USA

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#)
- [Repository](#)
- [Archive](#)

Editor: [Open Journals](#)

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

The Automated Statistical and Machine Learning Platform for Biological Research (ASMLP-BR) software provides a platform that combines machine learning and statistical analysis for biology research. It is deployable as both a browser-based application and a standalone desktop software. Researchers can upload comma separated value (CSV) data files to train Random Forest classification making use of regression models and fully automated hyperparameter optimization. Our software performs comprehensive statistical tests through a unified interface requiring no programming expertise. The platform integrates data preprocessing, model training with version control, feature importance analysis, and interactive visualization, addressing the common workflow challenge of using multiple disconnected tools. Built with React 18.3 and TypeScript, it efficiently handles typical research datasets while allowing researchers to save and iteratively improve models through versioned training sessions. The complete implementation workflow from user interaction through model storage is illustrated herein.

Statement of Need

Biological and biomedical researchers routinely need to apply machine learning and statistics to experimental data, but existing tools create significant barriers. Powerful frameworks like scikit-learn (Pedregosa et al., 2011) and R (R Core Team, 2023) require programming expertise that many experimental scientists lack. Tools operate in isolation. Researchers must manually transfer data between separate programs for statistical testing, machine learning, and visualization, reducing efficiency and introducing errors (Baker, 2016).

Our software addresses analysis and computational limitations by providing both web-based and desktop applications that combine Random Forest classification (Breiman, 2001) with standard statistical tests (t-tests, ANOVA, correlation) in one interface. The dual deployment model offers flexibility: researchers can use the browser version or download the standalone desktop application for offline work and enhanced data privacy. Unlike Jupyter notebooks (Kluyver et al., 2016), it requires no coding knowledge. The limitations seen through visual tools like Orange (Demšar et al., 2013) are eliminated as the ASMLP-BR it includes comprehensive statistical testing alongside machine learning. The platform enables complete workflows to upload data, train models iteratively with version control, test hypotheses, and generate visualizations, all without switching applications or writing code.

Key Features and Implementation

The ASMLP-BR platform's modular interface organizes functionality into distinct tabs for data upload, model training, prediction, result visualization, and statistical analysis (Figure 1). This workflow-oriented design guides users through the complete analysis pipeline while maintaining access to all features.

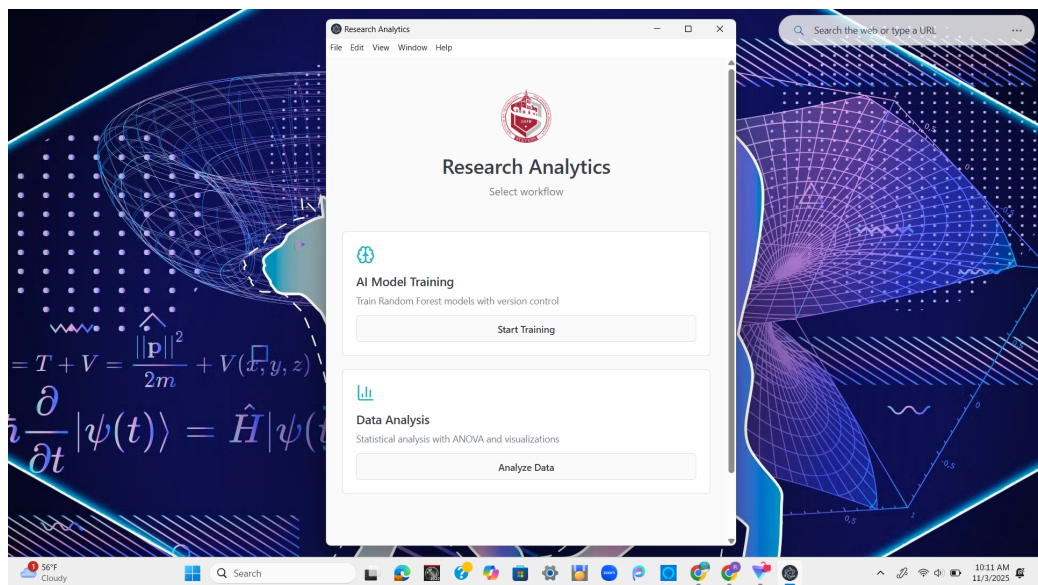


Figure 1: Interface dashboard showing the main analysis modules.

Architecture and Core Technologies

The application is built with React 18.3 and TypeScript, leveraging Vite for optimized production builds and Electron for desktop packaging. The implementation follows a modular component architecture that separates concerns across data processing, model training, statistical analysis, and visualization layers. Core dependencies include `ml-random-forest` (v2.1) for machine learning algorithms, `papaparse` (v5.5) for robust CSV parsing, and `recharts` (v2.15) for SVG-based interactive visualizations. All computation occurs client-side, eliminating server dependencies and ensuring data privacy. The desktop application packages the same codebase for Windows, macOS, and Linux platforms.

Data Upload and Preprocessing

The platform supports CSV file upload through drag-and-drop or file browser interfaces. Upon upload, the system performs automatic file structure detection and displays an interactive preview table showing the first 100 rows. Summary statistics (mean, median, standard deviation, quartiles, min/max) are computed for all numerical columns. Data validation identifies missing values, offering users options for row deletion or mean/median imputation. Preprocessing capabilities include z-score normalization, min-max scaling to [0,1], and automatic integer encoding of categorical variables. Column type detection distinguishes between numerical, categorical, and target variables, with manual override options.

Machine Learning Pipeline

The platform implements Random Forest classification (Breiman, 2001), widely used for chemical property prediction and QSAR modeling (Svetnik et al., 2003). Figure 2 illustrates the complete implementation workflow from initial data upload through final model storage,

66 showing how user interactions flow through data preprocessing, automated hyperparameter
67 optimization, model training, evaluation, and version management.

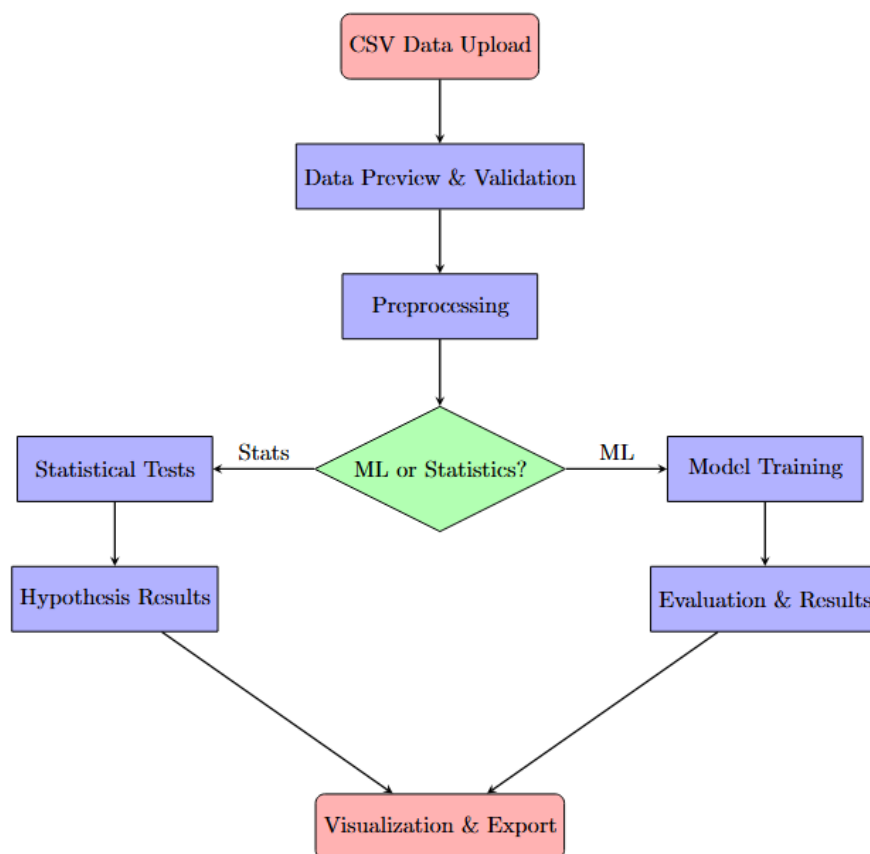


Figure 2: Implementation workflow from data upload through model storage.

68 Recognizing that most researchers lack expertise in hyperparameter tuning, the system automat-
69 ically optimizes Random Forest parameters based on dataset characteristics. The optimization
70 algorithm adjusts the number of trees (range: 10-500), maximum tree depth, and minimum
71 samples per split according to dataset size and feature dimensionality, eliminating the need for
72 manual configuration. Training executes asynchronously with real-time progress indicators to
73 maintain interface responsiveness.

74 The system performs stratified 80/20 train-test splitting to preserve class distribution, crucial for
75 imbalanced chemical datasets. Post-training, the interface displays comprehensive performance
76 metrics including accuracy, precision, recall, F1-score, and interactive confusion matrices.
77 Feature importance scores, computed via mean decrease in impurity, reveal which molecular
78 descriptors most influence classification, supporting interpretable model analysis.

79 Trained models persist in browser local storage or local file system (desktop version) with
80 comprehensive version control. Researchers can save multiple model versions, each tagged
81 with training timestamp, dataset characteristics, and performance metrics. This versioning
82 system enables iterative model refinement, wherein users can load previous versions, add new
83 training data, and create improved versions while maintaining the training history. Models
84 export as JSON files for deployment, sharing, or backup purposes.

Statistical Analysis Tools

The platform provides both parametric and non-parametric statistical tests for hypothesis testing and exploratory analysis. For comparing group means, Welch's t-test (Welch, 1947) handles unequal variances, while the Mann-Whitney U test offers a distribution-free alternative for non-normal data. One-way ANOVA enables multi-group comparisons. Correlation analysis includes Pearson's coefficient (Pearson, 1895) for linear relationships and Spearman's rank correlation for monotonic associations.

All statistical tests output comprehensive reports including p-values, effect sizes (Cohen's d, r), and 95% confidence intervals. The interface provides contextual guidance on assumption checking (normality, homoscedasticity) and appropriate test selection based on data characteristics. Visual diagnostics include Q-Q plots and residual plots for assumption validation.

User Interface Design

The interface employs tab-based navigation mirroring typical analysis workflows: Data Upload → Model Training → Prediction → Results → Statistical Analysis. Tabs remain disabled until prerequisite steps complete, preventing workflow errors. Form inputs include real-time validation with error messages and tooltip hints. The responsive design adapts to desktop and tablet viewports. Model management features include persistent storage (browser local storage with 5MB capacity or unlimited desktop file system), version control with timestamp metadata and performance tracking, and JSON import/export for model sharing and backup.

Acknowledgements

The authors acknowledge the Department of Biomedical Engineering, Department of Biology, and Department of Computer Science at Stevens Institute of Technology for institutional support. This work was supported by computational resources provided by Stevens Institute of Technology. Last but not the least, special thanks to shadcn for the free to use opensource, userinterface templates.

References

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., & others. (2016). TensorFlow: A system for large-scale machine learning. *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 265–283.
- Baker, M. (2016). 1,500 scientists lift the lid on reproducibility. *Nature*, 533, 452–454. <https://doi.org/10.1038/533452a>
- Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13, 281–305.
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. <https://doi.org/10.1023/A:1010933404324>
- Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297. <https://doi.org/10.1007/BF00994018>
- Darnag, R., Minaoui, B., Glorennec, P. Y., Fakri, A., Zahrae, O., & Mouchid, M. (2010). QSAR studies of HEPT derivatives using support vector machines and neural networks. *QSAR & Combinatorial Science*, 29(5), 567–577. <https://doi.org/10.1002/qsar.200960055>
- Demšar, J., Curk, T., Erjavec, A., Gorup, Č., Hočevár, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., & others. (2013). Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14, 2349–2353.

- 129 Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals*
130 *of Statistics*, 29(5), 1189–1232. <https://doi.org/10.1214/aos/1013203451>
- 131 Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning:*
132 *Data mining, inference, and prediction* (2nd ed.). Springer. [https://doi.org/10.1007/](https://doi.org/10.1007/978-0-387-84858-7)
133 [978-0-387-84858-7](https://doi.org/10.1007/978-0-387-84858-7)
- 134 Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K.,
135 Hamrick, J., Grout, J., Corlay, S., & others. (2016). Jupyter notebooks—a publishing
136 format for reproducible computational workflows. In F. Loizides & B. Schmidt (Eds.),
137 *Positioning and power in academic publishing: Players, agents and agendas* (pp. 87–90).
138 IOS Press. <https://doi.org/10.3233/978-1-61499-649-1-87>
- 139 Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables
140 is stochastically larger than the other. *Annals of Mathematical Statistics*, 18(1), 50–60.
141 <https://doi.org/10.1214/aoms/1177730491>
- 142 Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. MIT Press. ISBN: 978-
143 0262018029
- 144 Pearson, K. (1895). Notes on regression and inheritance in the case of two parents. *Proceedings*
145 *of the Royal Society of London*, 58, 240–242.
- 146 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M.,
147 Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D.,
148 Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in python.
149 *Journal of Machine Learning Research*, 12, 2825–2830.
- 150 Perkel, J. M. (2021). Ten simple rules for writing and sharing computational analyses in jupyter
151 notebooks. *PLOS Computational Biology*, 17(7), e1008993. [https://doi.org/10.1371/](https://doi.org/10.1371/journal.pcbi.1008993)
152 [journal.pcbi.1008993](https://doi.org/10.1371/journal.pcbi.1008993)
- 153 R Core Team. (2023). *R: A language and environment for statistical computing*. R Foundation
154 for Statistical Computing. <https://www.R-project.org/>
- 155 Spearman, C. (1904). The proof and measurement of association between two things. *American*
156 *Journal of Psychology*, 15(1), 72–101. <https://doi.org/10.2307/1412159>
- 157 Svetnik, V., Liaw, A., Tong, C., Culberson, J. C., Sheridan, R. P., & Feuston, B. P. (2003).
158 Random forest: A classification and regression tool for compound classification and QSAR
159 modeling. *Journal of Chemical Information and Computer Sciences*, 43(6), 1947–1958.
160 <https://doi.org/10.1021/ci034160g>
- 161 Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine*
162 *Learning Research*, 9, 2579–2605.
- 163 Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D.,
164 Burovski, E., Peterson, P., Weckesser, W., Bright, J., & others. (2020). SciPy 1.0:
165 Fundamental algorithms for scientific computing in python. *Nature Methods*, 17, 261–272.
166 <https://doi.org/10.1038/s41592-019-0686-2>
- 167 Welch, B. L. (1947). The generalization of student's problem when several different population
168 variances are involved. *Biometrika*, 34(1-2), 28–35. [https://doi.org/10.1093/biomet/34.](https://doi.org/10.1093/biomet/34.1-2.28)
169 [1-2.28](https://doi.org/10.1093/biomet/34.1-2.28)