Supplementary Material

Machine Learning Approaches to Predict Alcohol Consumption from Biomarkers in the UK Biobank

Mohammed F. Hassan^{*1,2}, Amanda Elswick Gentry^{*1,2}, Elizabeth C. Prom-Wormley^{1,3}, Roseann E. Peterson^{4,1}, Bradley T. Webb^{5,1}

* Joint-First Authors

¹Virginia Institute for Psychiatric and Behavioral Genetics, Virginia Commonwealth University, Richmond, Virginia, USA;

²Department of Psychiatry, School of Medicine, Virginia Commonwealth University, Richmond, Virginia, USA;

³Department of Epidemiology, School of Public Health, Virginia Commonwealth University, Richmond, Virginia, USA;

⁴Department of Psychiatry and Behavioral Sciences, Institute for Genomics in Health, SUNY Downstate Health Sciences University, Brooklyn, New York, USA;

⁵GenOmics and Translational Research Center, RTI International, Research Triangle Park, North Carolina, USA

1. Supplementary Section S1: Definition of Drink Per Week in UK Biobank

Drinks Per Week (DPW) calculation in UK Biobank (UKB) data is quantified as the average number of alcoholic drinks consumed by a participant per week. UKB collects this data through self-reported questionnaires where participants are asked about their typical drinking habits, including the frequency and quantity of alcohol consumption. The details of the DPW calculation can be summarized as follows:

- **Questionnaire Data**: Participants are asked how often they drink alcohol and how many drinks they typically consume on each occasion. The questions usually cover different types of alcoholic beverages, such as beer, wine, spirits, etc.
- **Frequency of Drinking**: This data includes:
	- o Daily
	- o 3-4 times a week
	- o Once or twice a week
	- o 1-3 times a month
	- o Rarely
	- o Never
- **Quantity Per Occasion**: Participants indicate how many drinks they typically consume per drinking session, which can vary by drink type such as number of glasses of wine.

The formula used for the calculation of DPW is defined as

 $DPW = (Frequency of drinking) \times (Number of Drinks per Occasion)$ (1)

2. Supplementary Section S2: Review of machine learning methods

2.1. LASSO and Ridge Regressions

LASSO stands for Absolute Shrinkage and Selection Operator. It is an L1 regularization technique to estimate the relationship between independent variables and the dependent variable. For N samples, the response (dependent) variable (y_i) is approximated by a linear combination of independent variables [1]

$$
\bar{y}(x_i) = \beta_0 + \sum_{j=1}^{p} x_{ij} \beta_j.
$$
 (2)

The model is parameterized by the vector of regression weights $\beta_0, \beta_1, ..., \beta_n$ that are optimized during the fitting process and \bar{y}_i is the estimated dependent variable. To estimate the fitted line, the variable coefficient $\beta_0, \beta_1, ..., \beta_p$ should be optimized according to the following loss function

$$
arg_{\beta_{0}\beta_{j}} min \left\{ \frac{1}{N} \sum_{i=1}^{N} \left(y_{i} - \beta_{0} - \sum_{j=1}^{P} x_{ij} \beta_{j} \right)^{2} \right\}
$$
(3)

In LASSO, the loss function is modified by adding a regularized term to the loss function which is the sum of the absolute value of the coefficients as defined in the following objective function

$$
arg_{\beta_{0}\beta_{j}} min \left\{ \frac{1}{N} \sum_{i=1}^{N} \left(y_{i} - \beta_{0} - \sum_{j=1}^{P} x_{ij} \beta_{j} \right)^{2} \right\} + \lambda \sum_{j=1}^{P} |\beta_{j}| \tag{4}
$$

To get the best-fitted line, the cost function should be optimized by forcing the coefficient variables to zero and thus directly performing feature selection where λ is a tuning parameter that controls the strength of the penalty.

Ridge regression [2] is considered as L2 regularization, the added regularized term to the loss function is the sum squares of the coefficients as defined in the following objective function

$$
arg_{\beta_{0}\beta_{j}} min \left\{ \frac{1}{N} \sum_{i=1}^{N} \left(y_{i} - \beta_{0} - \sum_{j=1}^{p} x_{ij} \beta_{j} \right)^{2} \right\} + \lambda \sum_{j=1}^{p} \beta_{j}^{2}
$$
 (5)

Although LASSO and Ridge regressions are built using similar structures, they have many key differences. LASSO can perform feature selection by shrinking some coefficients to zero, while Ridge regression does not. In terms of model complexity LASSO can create simpler, more interpretable models by selecting a subset of the features. Ridge regression typically includes all features but with smaller, more balanced coefficients. Regarding handling multicollinearity both methods handle multicollinearity, but Ridge is more effective when all predictors are relevant as it keeps all features. In terms of performance Ridge regression may perform better when dealing with highly correlated features, while LASSO may be more useful when dealing with highdimensional data with many irrelevant features.

Key Parameters for LASSO and Ridge [3]

- Lambda (λ): This is the regularization parameter. Higher values of λ lead to more regularization, which means that the coefficients can be reduced to zero, effectively performing feature selection and often determined using cross-validation.
	- o minimum λ for LASSO regression = 0.000256
	- o minimum λ for Ridge regression = 0.2558.
- Alpha: Determines the type of regularization
	- \circ alpha = 1: Lasso regression.
	- \circ alpha = 0: Ridge regression.
	- \circ 0 < alpha < 1: Elastic Net (combination of Lasso and Ridge).
- Family: The type of model being fit
	- o "Gaussian": Linear regression.
	- o "binomial": Logistic regression.
	- o "poisson", "multinomial.
		- \blacksquare Family = "Gaussian".
- Intercept: A boolean value indicating whether to calculate the intercept for the model. If set to False, no intercept will be used in the calculations.
	- \circ Intercept = True.
- Standardize: A boolean value, when set to True, standardizes the regressors before fitting the model. Standardization means that the mean will be 0, and the standard deviation will be 1.
	- \circ Standardize = True.
- Max Iter: The maximum number of iterations for the optimization algorithm. If the algorithm does not converge within this number of iterations, it will stop.
	- \circ Max Iter= 100.
- Tolerance: The tolerance for the optimization. It determines when the optimization should stop if the improvement in the cost function is below this threshold.
	- \circ Tolerance = 1e-7.

2.2. MBOOST, GBM, and XGBoost

MBOOST stands for model boosting [4], GBM for gradient boosting [5], and XGBOOST [6] for extreme boosting. These algorithms refer to different implementations of boosting algorithms used in machine learning for regression and classification tasks. In general, they build on similar structures by combining the outputs of several weak models to create a strong predictive model. The primary idea is to sequentially apply the weak models to adjust the weights of incorrectly classified instances and focus more on the difficult cases in subsequent iterations.

MBOOST is an R package designed for model-based boosting, offering flexible and interpretable boosting methods. MBOOST package supports a wide range of base learners and loss functions and is intended to provide interpretable models, making it easier to understand the impact of individual predictors. However, it may be slower and less efficient compared to other implementations, particularly for large datasets.

Key Parameters for MBOOST [4]

- Base Learners: Define the type of model used in each boosting iteration. Some common base learners and their defaults include:
	- \circ bbs (B-splines)
	- bols (Linear effects)
	- btree (Decision stumps/trees)
		- **•** base learners = "bols", number of base learners = 303 .
- Family (Loss Function): The loss function is specified using the family parameter. Common choices for regression include:
	- Gaussian ()
	- \circ Laplace ()
	- Huber ()
		- \blacksquare Family = Gaussian.
- Number of Boosting Iterations (mstop): This can be optimized using cross-validation.
	- \circ mstop = 1000.
- Learning Rate (nu): determines the step size at each boosting iteration. Smaller values lead to slower learning but can result in better generalization.
	- \circ nu = 0.1.

GBM is another R package that provides an implementation of gradient boosting. It is one of the earlier implementations and is widely used for a variety of predictive modeling tasks. It is suited for a range of tasks and includes a variety of options for tuning and regularization. GBM can handle various types of data and tasks. It is robust to overfitting with proper tuning and capable of capturing complex patterns.

Key Parameters for GBM [7]

- Distribution: Specifies the loss function used in the model. For regression tasks, common choices are:
	- "laplace": For regression with a Laplace loss function.
	- "gaussian" is typically used for continuous target variables in regression problems. \blacksquare Distribution = "gaussian"
- n.trees: The number of trees to fit (i.e., the number of boosting iterations). Typically chosen through cross-validation.
- \circ n.trees = 10000.
- interaction.depth: The maximum depth of each tree. It controls how complex the individual trees can be. Larger values lead to more complex trees.
	- \circ interaction.depth = 2
- shrinkage (or learning.rate): The learning rate or shrinkage factor. It controls the contribution of each tree. A smaller value makes the learning process more conservative and reduces the risk of overfitting.
	- \circ Shrinkage = 0.01.
- n.minobsinnode: The minimum number of observations required in a terminal node (leaf). This controls the size of the trees. Larger values can prevent overfitting.
	- \circ n.minobsinnode = 15.
- verbose: Whether to print progress messages during the fitting process.
	- \circ verbose = FALSE.
- bag.fraction: The fraction of the training data used for each iteration of boosting. This can help prevent overfitting by creating a more diverse set of trees.
	- \circ bag.fraction = 0.5.

XGBoost is an R package that provides a highly efficient and scalable implementation of the gradient boosting framework. It is designed for speed and performance, making it a popular choice for machine learning tasks, particularly those involving structured/tabular data. XGBoost is designed to be efficient in both computation and memory usage. It supports parallel processing, making it faster than many other implementations of gradient boosting. It includes L1 and L2 regularizations to prevent overfitting, which can improve the model's generalization capabilities. It can handle missing values within the dataset, using a sparse matrix format and a special handling strategy.

Key Parameters for XGBoost [8]

- General Parameters: These parameters control the overall behavior of the XGBoost model.
	- o booster: Type of booster to use.
		- Options: "gbtree", "gblinear", "dart"
			- Booster $=$ "gbtree"
	- o nthread: Number of threads to use for training.
		- \blacksquare nthread = 1 (single-threaded)
	- o verbosity: Controls the level of messages printed.
		- Options: 0 (silent), 1 (warnings), 2 (info), 3 (debug)
- Booster Parameters: Tree Booster (gbtree and dart)
	- o eta (or learning_rate): Step size shrinkage to prevent overfitting.
		- $eta = 0.01$
	- o max_depth: Maximum depth of a tree.
		- max depth = 15
	- o min_child_weight: Minimum sum of instance weight (hessian) needed in a child.
		- \blacksquare min child weight = 1
- o gamma: Minimum loss reduction required to make a further partition.
	- gamma $= 100$
- o subsample: Fraction of samples used to grow trees.
	- Subsample $= 1$
- o colsample_bytree: Fraction of features used per tree.
	- colsample bytree $= 1$
- o colsample_bylevel: Fraction of features used per level.
	- colsample bylevel $= 1.0$
- o colsample_bynode: Fraction of features used per split.
	- colsample bynode $= 1.0$
- o lambda (or reg_lambda): L2 regularization term on weights.
	- Lambda = 100
- o alpha (or reg_alpha): L1 regularization term on weights.
	- alpha = 100
- o tree_method: Tree construction algorithm.
	- § Options: "auto", "exact", "approx", "hist", "gpu_hist"
- Learning Task Parameters: These parameters are specific to regression tasks.
	- o objective: Specifies the learning task.
		- Common options for regression:
			- "reg:squarederror" (default): Mean squared error regression.
			- "reg:logistic": Logistic regression.
			- "reg:pseudohubererror": Pseudo Huber regression.
		- Other advanced options include quantile and Poisson regression.
	- o eval_metric: Evaluation metrics for validation data.
		- Options:
			- "rmse": Root Mean Squared Error (default for regression).
			- "mae": Mean Absolute Error.
			- "mape": Mean Absolute Percentage Error.
			- "poisson-nloglik": Negative log-likelihood for Poisson regression.
			- "gamma-nloglik": Negative log-likelihood for gamma regression.
			- "logloss": Log loss.
	- o base score: Initial prediction score (global bias).
		- base $score = 7.524918$

3. Supplementary Section S3: ML Model Evaluation and Explanation.

3.1. Model Evaluation

In order to compare the performance of machine learning models, we need to select an appropriate tool for a given task (regression or classification), which is considered essential to assess the model's performance and make informed decisions.

For regression tasks [9], there are several evaluation metrics and techniques commonly used to assess the quality of regression models. In this work, we used two methods; the first is the Mean Absolute Error (MAE), which measures the average absolute difference between predicted and actual values. It is less sensitive to outliers compared to some other metrics. MAE is defined as

$$
MAE = \left(\frac{1}{n}\right) \times \sum |Y_i - \overline{Y}_i| \tag{6}
$$

where *n* is the number of data points. Y_i and \overline{Y}_i are actual, and predicted phenotypes, respectively. The second method called Adjusted R-squared is a statistical metric used in regression analysis to assess the goodness of fit of a regression model. It is an improvement over the regular R-squared and takes into account the number of predictor variables in the model. Rsquared measures the proportion of the variance in the dependent variable that is explained by the independent variables in a regression model. However, as more independent variables are added to the model, R-squared tends to increase, even if those additional variables do not significantly improve the model's predictive power. Adjusted R-squared addresses this issue by penalizing the inclusion of unnecessary variables in the model. It is calculated using the formula:

$$
adjusted R_Squared = 1 - \left[\left(1 - R_Squared \right) \times \frac{(n-1)}{(n-p-1)} \right] \tag{7}
$$

where R-squared is the regular coefficient of determination. n is the number of observations (data points), and p is the number of predictor variables in the model.

For classification tasks [10], different ML metrics are used for evaluation. All these metrics are derived from the confusion matrix. A confusion matrix is a table used to evaluate the performance of a classification algorithm. It summarizes the predicted results of a model by comparing the actual (true) labels with the predicted labels for a set of data. It is especially useful in binary and multiclass classification problems.

The confusion matrix is defined as follows:

- True Positive (TP) : The number of instances correctly predicted as positive.
- True Negative (TN) : The number of instances correctly predicted as negative.
- False Positive (FP) : The number of instances incorrectly predicted as positive (also called a "Type I error").
- False Negative (FN) : The number of instances incorrectly predicted as negative (also called a "Type II error").

The following are definitions of the metrics used throughout this work

● Accuracy: Measures overall correctness.

$$
Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}.
$$

● Positive Predictive Value (Precision): Measures the proportion of correct positive predictions.

Postive Predictive Value = $\frac{TP}{(TP + FP)}$.

● Sensitivity (Recall or True Positive Rate): Measures the proportion of actual positives that are correctly identified

$$
Sensitivity = \frac{TP}{(TP + FN)}.
$$

● Specificity (True Negative Rate): Measures the proportion of actual negatives that are correctly identified.

$$
Specificity = \frac{IN}{(TN + FP)}
$$

 \bullet F1 score: A harmonic mean of precision and recall, useful when there is an imbalance

.

between positive and negative samples.
 $F1 score = \frac{2(Precision \times Recall)}{(Precision + Recall)}$.

● Negative Predictive Value (NPV): Measures the probability that individuals who test negative for a condition truly do not have the condition.

Negative Predictive Value $=$ $\frac{TN}{(TN + FN)}$.

3.2. Model Hyperparameter Tuning

All ML algorithms mentioned above need tuning [11], often referred to as hyperparameter tuning, which is the process of optimizing the hyperparameters during training of a machine learning model to achieve better performance on a specific task or dataset. Properly tuning hyperparameters can significantly improve the generalization performance of a machine-learning model. The tuned parameters for each ML model depend on its internal structure and are learned using different methods. In this work, we used the grid search to find the best hyperparameter values. A grid search is used to exhaustively search through all possible combinations of hyperparameters within the defined search space. It's simple but can be computationally expensive. In this work, the hyperparameters used in the training five ML methods are defined in the Supplementary Section S2.

3.3. Model Interpretability

Machine learning interpretability is crucial when applied to the field of biology to extract meaningful insight [12]. By understanding the overarching process by which the input variables generate why a model makes a certain prediction, researchers can identify relevant phenotypes,

molecules, genes, proteins, or pathways that are involved in a particular biological phenomenon. In this work, we used feature importance, a concept commonly used in machine learning and data analysis to understand the relative significance of input variables (blood biomarkers) in predicting a given phenotype. It helps in identifying which markers or covariate variables have the most influence on the model's predictions. Various techniques can determine feature importance depending on a given machine learning algorithm. In models that use L1 regularization (e.g., LASSO), features that have non-zero coefficients after regularization are considered important, while features with zero coefficients are considered unimportant. Boosting algorithms (e.g., MBOOST, GBM, XGBOOST) use decision trees as base models. In decision trees, features that split the data into classes effectively are considered more important. Information gain or Gini impurity is commonly used to evaluate the quality of a split. Features that lead to nodes with a substantial reduction in impurity are considered more important.

SHapley Additive exPlanations (SHAP) [13] is another method that is used both in local and global interpretably of ML. SHAP is a framework for interpreting the output of machine learning models. It is based on cooperative game theory and the concept of Shapley values, which were introduced by Lloyd Shapley. The Shapley value provides a way to fairly distribute rewards among a group of contributors. In the context of machine learning interpretability, SHAP values aim to attribute the contribution of each feature to the prediction made by a model. These values provide insights into the importance of different features in determining the output of a model for a particular instance. The mathematical formula for SHAP analysis can be summarized as follows

$$
g(x) = \mu_0 + \sum_{i=1}^{N} \mu_i(g, x),
$$
\n(8)

where

- $g(x)$ is the model's prediction for an instance x.
- μ_0 is the base value or the average model output over the training dataset.
- $\mu_i(f, x)$ is the SHAP value for the feature *i*.
- *is the number of features.*

The SHAP value for the feature i is calculated as follows

$$
\mu_i(g, x) = \sum_{S \subseteq M \setminus \{i\}} \frac{|S|! \times (|M| - |S| - 1)!}{|M|!} \times [g(x_s \cup \{i\} - g(x_s))],
$$
\n(9)

where

- *is the set of all features.*
- S is a subset of features that does not include feature *i*.
- $|S|$ is the number of elements in the subset *S*.
- $g(x_s)$ is the model prediction for the subset *S* of features.

• $g(x, \cup \{i\})$ is the model prediction for the subset S of features plus feature *i*.

4. Supplementary Table T1: UKB Variables

The UKB is a large-scale biomedical research resource with data from \sim 500,000 volunteer participants aged between 40-69 years recruited in 2006-2010 from across the UK and contains a rich array of measures including lifestyle and health surveys, EHR derived diagnoses, blood chemistry, genetics, metabolomics, proteomics, imaging, and accelerometer data. With their consent, participants provided detailed information about their lifestyle, physical measures and had blood, urine, and saliva samples collected and stored for future analysis. In this work, the UKB blood biomarkers that are used for the phenotypic predictions consist of 249 Nuclear Magnetic Resonance (NMR) biomarkers from plasma, 31 blood count measures, 30 blood biochemistry measures, 25 infectious disease blood measures, and 3 urine assay measures summing up to a total of 338 initial predictors. In all analyses, age, sex, and statin use were considered as covariates. The inclusion of statin use was necessary due to the significant impact of statins on lipid measures, which are relevant to many predictors. Statin use was derived from the UKB field 20003 surveying prescription drug use. A binary outcome was produced based on the reported use of any HMG-CoA reductase inhibitors including Atorvastatin, Cerivastatin, Fluvastatin, Lovastatin, Mevastatin, Pitavastatin, Pravastatin, Rosuvastatin, Simvastatin, Eptastatin, and Velastatin.

5. Supplementary Table T2: Details of UKB Variables

Details of the UKB variable used for training ML methods that contain FieldID, variable name, number of participants, and variable measuring units. More details are available in the UKB online platform.

6. Supplementary Table T3: Data filtering and splitting

Data filtering involves multiple tasks such as labeling, handling missing values, encoding categorical variables, scaling, and normalizing numerical variables. The UK Biobank is one of the largest and most comprehensive population-based biobanks in the world, collecting a wide range of health-related data from hundreds of thousands of participants. Given the volume and complexity of UKB data, careful data treatment is required to guarantee satisfactory results. Characterizing patterns of missingness, and standardizing data to a common format and units,

especially when data come from diverse sources, are all essential before feeding features into the ML pipeline to achieve meaningful results.

The starting sample size consists of 359,980 independent European UKB individuals. The final sample size and predictor variables were determined for each outcome separately using a fourstep filtering process. First, subjects with the missing outcome of interest were dropped. Second, subjects missing NMR metabolite data were dropped, as this assay represents the primary predictors of interest. Third, predictor variables missing in 10% or more of the remaining subjects were dropped. Finally, any subjects with missingness in the remaining predictors were dropped.

Data splitting divides the dataset into multiple subsets for training, validation, and testing purposes. First, the entire dataset is split into a test and a training set, typically along a 20/80 (test/train) split. The training set is used to estimate, assess, and hyper-tune the ML model's parameters, while the test set is completely withheld to validate the final ML model. Additional data splitting within the training set is necessary to estimate the optimal ML model for each implementation called a validation set. Within the training set, data splitting is accomplished by K-Fold Cross-Validation (CV,) a widely used technique in ML and statistical modeling for assessing predictive model performance and generalizability. The typical value for K-Fold Cross-Validation is 5. The sample sizes for all data filtering and splitting stages are shown in the table below.

7. Supplementary Table T4: Adjusted R2 for the predicted height, BMI, DPW, BF%, and MDDsx.

The training process is organized into three steps in order of increasing model complexity and prediction accuracy. First, a linear regression model is trained using only age, sex, and statin use as predictors. Second, a linear regression model is trained on age, sex, and statin use, in addition to four blood biochemistry assay metrics of liver function: alkaline phosphatase (AlkPhos), alanine aminotransferase (ALT), aspartate aminotransferase (AST), and gammaglutamyltransferase (GGT). The third and final models were trained using five ML frameworks to include all blood-based biomarkers, in addition to age, sex, and statin use.

8. Supplementary Table T5: MAE for the predicted height, BMI, DPW, BF%, and MDDsx. Estimate MAE for three ML models. First, a linear regression model is trained using only age, sex, and statin use as predictors. Second, a linear regression model is trained on age, sex, and statin use, in addition to four blood biochemistry assay metrics of liver function: alkaline phosphatase (AlkPhos), alanine aminotransferase (ALT), aspartate aminotransferase (AST), and gamma-glutamyltransferase (GGT). The third and final models were trained using five ML frameworks to include all blood-based biomarkers, in addition to age, sex, and statin use.

9. Supplementary Table T6: Estimated heritability

LDSC estimated heritabilities for observed and predicted DPW including labeling of seven sets, sample size, intercept, and heritability with standard error.

10. Supplementary Table T7: Estimated genetic correlations

LDSC estimated genetic correlations for observed and predicted DPW including labeling of 21 sets, genetic correlation with standard error, and p-value. The sets are generated by a pairwise combination of sets defined in supplementary Table T6.

11. Supplementary Table T8: Confusion matrices for DPW>8

Confusion matrices for predicted DPW>8 for females across five ML models.

12. Supplementary Table T9: Confusion matrices for DPW>15

Confusion matrices for predicted DPW>15 for males across five ML models.

13. Supplementary Table T10: Evaluation of ML metrics for DPW>8

ML Metrics evaluation including Accuracy, F1 score, Sensitivity (Recall), Specificity, Positive Predictive Value (PPV) (Precision), and Negative Predictive Value (NPV) for females and predicted DPW>8 across five ML models.

14. Supplementary Table T11: Evaluation of ML metrics for DPW>15

ML Metrics evaluation including Accuracy, Recall, Precision, F1 score, Sensitivity (Recall), Specificity, Positive Predictive Value (PPV) (Precision), and Negative Predictive Value (NPV) for males and predicted DPW>15 across five ML models.

15. Supplementary Table T12: A full list of all SHAP values of the blood-based biomarkers used to train ML models.

A full list of SHAP values for all the variables ranked in descending order from the high SHAP score to the lowest score. We used the R package "SHAPforxgboost" and the XGBoost model to generate these values.

16. Supplementary Figure F1: A histogram of alcohol consumption in terms of DPW across sex. As shown males dominate the heavy drinking spectrum compared to females.

Histogram of Drinks Per Week (DPW) Across Sex

17. Supplementary Figure F2: A dendrogram plot of the full list of the blood biomarkers used to predict DPW. Variables that are close to each other represent a strong relationship.

References

- 1. Tibshirani R. Regression shrinkage and selection via the lasso. J R Stat Soc Series B Stat Methodol. 1996;58: 267–288.
- 2. Ehsanes Saleh AK, Arashi M, Golam Kibria BM. Theory of ridge regression estimation with applications. Saleh AKME, Arashi M, Kibria BMG, editors. Nashville, TN: John Wiley & Sons; 2019.
- 3. Hastie T, Qian J, Tay K. An introduction to glmnet. CRAN R Repositary. 2021;5: 1–35.
- 4. Hofner B, Mayr A, Robinzonov N, Schmid M. Model-based boosting in R: a hands-on tutorial using the R package mboost. Comput Stat. 2014;29: 3–35.
- 5. Natekin A, Knoll A. Gradient boosting machines, a tutorial. Front Neurorobot. 2013;7: 21.
- 6. Chen T, Guestrin C. XGBoost. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA: ACM; 2016. doi:10.1145/2939672.2939785
- 7. Ridgeway G. Generalized Boosted Models: A guide to the gbm package. Update. 2007;1.
- 8. Chen T, He T, Benesty M, Khotilovich V, Tang Y, Cho H, et al. Xgboost: extreme gradient boosting. R package version. 2015;1: 1–4.
- 9. Sekerogiu B, Ever YK, Dimililer K, Al-turjman F. Comparative evaluation and comprehensive analysis of machine learning models for regression problems. Data Intell. 2022;4: 620–652.
- 10. Rainio O, Teuho J, Klén R. Evaluation metrics and statistical tests for machine learning. Sci Rep. 2024;14. doi:10.1038/s41598-024-56706-x
- 11. Yang L, Shami A. On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing. 2020;415: 295–316.
- 12. Sidak D, Schwarzerová J, Weckwerth W, Waldherr S. Interpretable machine learning methods for predictions in systems biology from omics data. Front Mol Biosci. 2022;9: 926623.
- 13. Lundberg S, Lee S-I. A unified approach to interpreting model predictions. arXiv [cs.AI]. 2017. Available: http://arxiv.org/abs/1705.07874