

Supplementary Information

Supplementary table S1: Benchmarked FS methods

Method	Description	Reference
Embedded methods		
LASSO	Logistic regression with the L1 penalty. Regularization strength C chosen with stratified 5-fold CV, scoring <code>roc_auc</code> .	[29]
Adaptive LASSO	LASSO variant that applies adaptive weights to each coefficient, allowing to penalize unimportant variables more than important ones. Initial regularization parameter $C = 1$, halved iteratively until the selected feature subset size satisfies $ S \leq \text{max_num_selected}$.	[30]
Exclusive LASSO	Applies the L1 penalty at group level: $[\arg \min_{\beta} \frac{1}{n} \sum_{i=1}^n \ell(y_i, x_i^{\top} \beta) + \lambda \sum_{g \in \mathcal{G}} \frac{\ \beta_g\ _1^2}{2}]$ In <i>ROOFS</i> , groups $\mathcal{G}$ are defined via hierarchical clustering, with $ \mathcal{G} = 0.1 \cdot p$ (p is the total number of features in the data). 5-fold CV, <code>n_lambda</code> = 150.	[31]
Ensemble embedded methods		
Bolasso	Modified bootstrap LASSO. Selects features with a bootstrap frequency of selection of ≥ 0.5 instead of 1.	Our modification of [32]
Stability selection	LASSO-based subsampling method that performs repeated fits across a range of regularization parameters and selects features selected with a frequency above a specified threshold. Regularization $C \in \text{logspace}(-5, -1, 50)$, <code>sample_fraction</code> = 0.5, selection probability threshold $\tau = 0.6$.	[33]
Stabl LASSO	Recent extension of stability selection that injects artificial noise variables to enable data-driven choice of the selection frequency threshold. <code>n_bootstraps</code> = 1000, <code>artificial_type</code> = "random_permutation".	[22]
RENT	Repeated Elastic Net Technique, a resampling-based method that selects features that satisfy three criteria: a sufficiently high selection frequency across resampled models $\tau_1 = 0.9$, consistent coefficient sign across models $\tau_2 = 0.9$, and a sufficiently large ratio of the mean coefficient to its standard error $\tau_3 = 0.975$.	[34]
Block LASSO	HSIC Method that the Hilbert-Schmidt Independence Criterion (HSIC) to identify non-linear dependencies between features and the outcome while penalizing redundant features. To reduce memory complexity for the case of high-dimensional data, the method partitions samples into $B = 10$ smaller blocks. The final bagging block HSIC estimator is computed by averaging HSIC values across multiple random permutations of the data partitions.	[36]
Filter methods (if not specified otherwise, number of features selected $k = 15$)		

Continued on next page

Supplementary table S1 – continued from previous page

Method	Description	Reference
CIFE	Iteratively selects features with the highest Conditional Infomax Feature Extraction (CIFE) score until the user-specified set size is reached. For each feature X_k: $\text{CIFE}(X_k) = I(X_k; Y) - \sum_{X_j \in \mathcal{S}} I(X_j; X_k) + \sum_{X_j \in \mathcal{S}} I(X_j; X_k Y)$ The first term maximizes mutual information (MI) between the feature and the outcome, the second penalizes redundancy with previously selected features, and the third rewards conditional redundancy given the outcome.	[2, 37]
CMIM	Iteratively selects features with the highest Conditional Mutual Information Maximization (CMIM) score until the user-specified set size is reached. For each feature X_k: $\text{CMIM}(X_k) = \min_{X_j \in \mathcal{S}} [I(X_k; Y X_j)]$ The CMIM score maximizes the minimal conditional MI between the candidate feature X_k and the outcome Y, with the minimum value chosen across all previously selected features $X_j \in \mathcal{S}$.	[2, 38]
DISR	Iteratively selects features with the highest Double Input Symmetrical Relevance (DISR) score until the user-specified set size is reached. For each feature X_k: $\text{DISR}(X_k) = \sum_{X_j \in \mathcal{F}} \frac{I(X_j X_k; Y)}{H(X_j X_k Y)}, \text{ where}$ $I(X_j X_k; Y) = I(X_k; Y) + I(X_j; Y X_k), \text{ and}$ $H(X_j X_k Y) = H(X_k) + H(X_k X_j) + H(Y X_k) - I(Y; X_j X_k).$	[2, 39]
JMI	Iteratively selects features with the highest Joint Mutual Information (JMI) score until the user-specified set size is reached. For each feature X_k: $\text{JMI}(X_k) = I(X_k; Y) - \beta \sum_{X_j \in \mathcal{S}} I(X_j; X_k) + \gamma \sum_{X_j \in \mathcal{S}} I(X_j; X_k Y), \text{ where}$ $\beta = \gamma = \frac{1}{ \mathcal{S} }, \text{ where } \mathcal{S} $ is the cardinality of a set to be selected.	[2, 40]
ReliefF	Ranks features by a distance-based score and selects top-k features. For each sample, the algorithm penalizes the features with different values in the nearest neighbors (5 by default) of the same class and rewards the features with different values for the nearest neighbors (similarly, 5 by default) of the opposite class.	[2, 41]
Fisher score	Ranks features by the ratio of between-class variance to within-class variance and selects top-k features.	[2]
Gini index	Ranks features by their minimum Gini impurity (computed by testing all possible binary splits) and selects top-k features.	[2]
t-score	Ranks features by their two-sample t-statistic and selects top-k features.	[2]
mRMR	Iteratively adds features with the highest relevance-redundancy ratio until the desired number of features is reached. Relevance is measured by the ANOVA F-value, redundancy is iteratively updated as the mean correlation with previously selected features.	

Continued on next page

Supplementary table S1 – continued from previous page

Method	Description	Reference
Hierarchical clustering	Performs hierarchical clustering of features based on Spearman correlation coefficients, with the number of clusters equal to the desired number of features to be selected. From each cluster, the feature with the highest point biserial correlation with the outcome is selected.	Our modification of [42]
p.adjust t-test + logistic .05	Union-based filter that combines features selected by t-test and logistic regression (controlling for user-specified covariates; PDL1 in the case of PIONeeR data), where p-values from each test were adjusted using the Benjamini-Hochberg procedure with a 0.05 threshold.	Our modification of p-value filtering
Wrapper methods		
LR-RFE	Recursive feature elimination that iteratively removes 10% of features with the lowest absolute coefficients from a logistic regression model with the L2 penalty until the pre-specified number of features is reached. Base learner: Ridge regression with $C = 0.001$.	[44]
RF-RFE	Recursive feature elimination that iteratively removes 10% of the least important feature of a random forest model and selects a subset with the best AUC in cross-validation. Base learner: Random forest (<code>n_estimators = 200</code> , <code>max_depth = 2</code> , <code>min_samples_leaf = 40</code>).	[45]
Forward RF	Recursive forward selection that iteratively adds the most important feature of a random forest model until a pre-specified set size is reached and selects a subset with the best AUC in cross-validation. Base learner: Random forest (same settings as above).	[46]
Shapicant	Wrapper that uses SHAP importances together with outcome permutation. The criterion of selection is the threshold on p-values from testing whether each feature’s SHAP value under the real outcome differs from that under the permuted outcome. In <i>ROOFS</i> , the default procedure begins with a threshold of 0.1 and iteratively decreases it, stopping after three iterations or once the number of selected features k falls below the user-specified <code>max_num_selected</code> . Base learner: Random forest (same settings as above).	[47]
Controls		
Random FS	Selects a feature subset randomly. For the benchmark on PIONeeR data, subset sizes k were randomly sampled from the range 1-20 for the real dataset. For the semi-synthetic dataset, subset size was either set to $k = 19$ (a fixed size corresponding to the number of predictors in the true model) or sampled randomly from $[1, p]$.	
Full model	Model trained with all features.	

Choosing an optimal subset size

For FS methods that select a fixed number of features, we recommend determining the optimal subset size by evaluating each method across a range of sizes using the `optimize_num_selected` function. For each combination of FS method, classifier, and subset size, it computes two metrics m_1 and m_2 (optimism-corrected AUC and FS stability by default). Then, for each method-classifier pair, an optimal subset size is identified via a distance-based criterion. Both metrics are first min-max normalized, and a weighted Euclidean distance to the point $(1, 1)$ is computed as:

$$d = \sqrt{w_1 (1 - m_1)^2 + w_2 (1 - m_2)^2} \quad (1)$$

where by default m_1 and m_2 are the normalized AUC and stability, respectively, and $w_1 = 0.8, w_2 = 0.2$, but these can be modified by the user. The weighting reflects the higher importance of predictive performance.

Then, a distance threshold is defined as:

$$d^* = d_{\min} + 0.5 \sigma_d \quad (2)$$

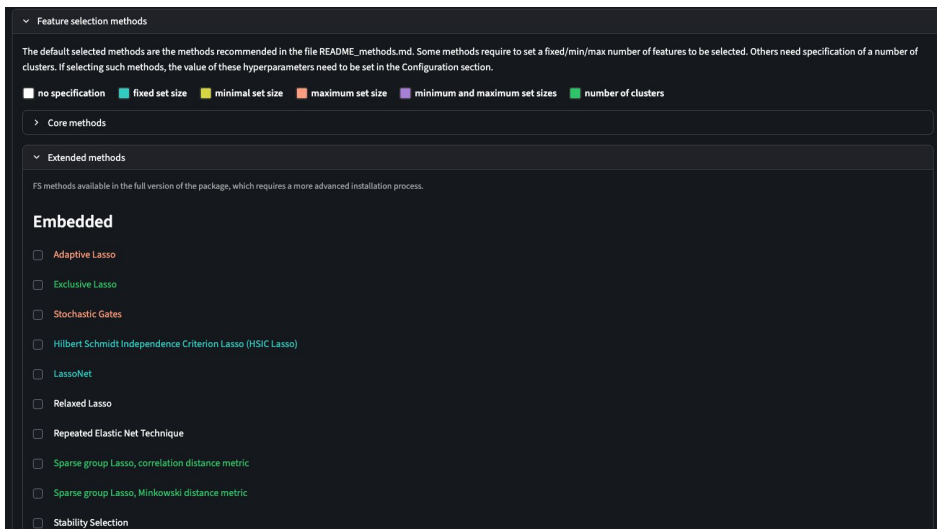
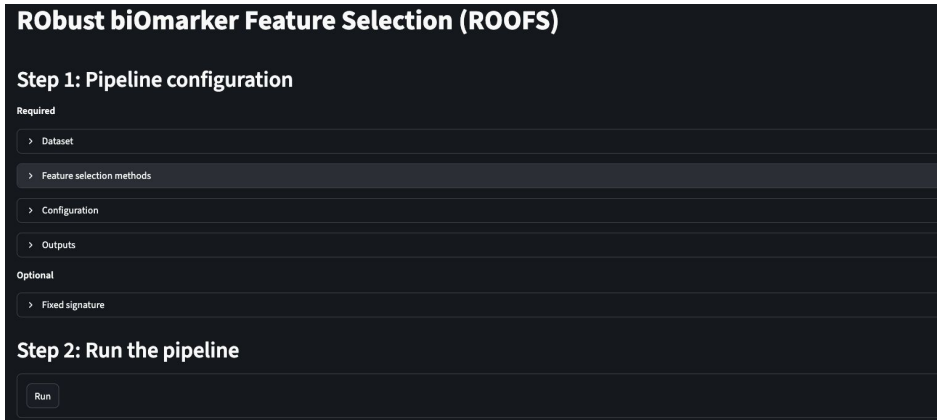
where $d_{\min}$ is the minimum observed distance and σ_d is its standard deviation across all tested settings. Among all subset sizes satisfying $d \leq d^*$, the smallest is selected as the recommended value to favor sparser feature sets that do not compromise predictive performance or stability. Results are visualized using a scatter plot, and the recommended configuration is highlighted for each method-classifier combination.

Supplementary table S2: ML classifiers used in FS benchmark on PIONeeR dataset

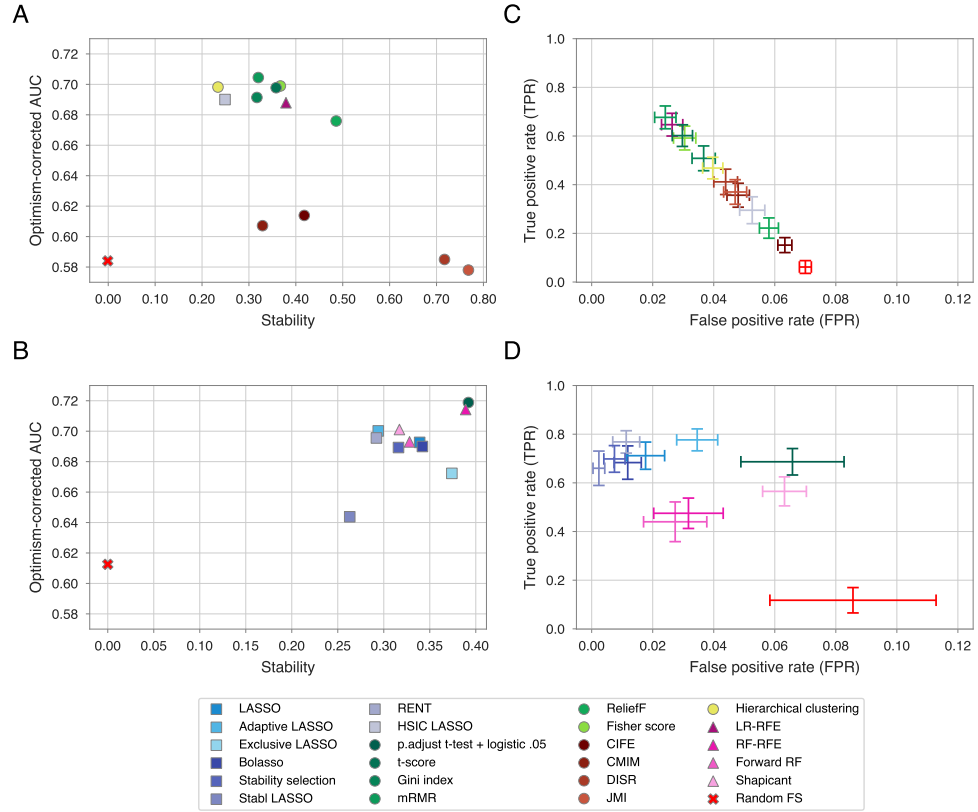
Category	Classifier	Hyperparameters ^a
Tree-based	BaggingClassifier	<code>base_estimator=DecisionTree</code> , <code>n_estimators=200</code> <code>max_depth=2</code> , <code>min_samples_leaf=40</code> , <code>max_features=0.7</code>
	ExtraTreesClassifier	<code>n_estimators=200</code> , <code>max_depth=2</code> <code>min_samples_leaf=40</code> , <code>max_features=0.7</code>
	RandomForestClassifier	<code>n_estimators=200</code> , <code>max_depth=2</code> <code>min_samples_leaf=40</code> , <code>max_features=0.7</code>
Boosting	GradientBoostingClassifier	<code>max_depth=2</code> , <code>min_samples_leaf=40</code> <code>max_features=0.7</code> , <code>subsample=0.7</code>
	HistGradientBoostingClassifier	<code>max_depth=2</code> , <code>min_samples_leaf=40</code> , <code>max_features=0.7</code>
	XGBClassifier	<code>max_depth=2</code> , <code>n_jobs=-1</code> , other parameters at default
Linear	LogisticRegression	<code>penalty=None</code> , <code>n_jobs=-1</code>
	LinearDiscriminantAnalysis	Default hyperparameters
Other	KNeighborsClassifier	Default
	MLPClassifier	Default
	SVC	<code>probability=True</code> , other parameters at default
Ensemble	StackingClassifier ^b	<code>cv=5</code> , <code>stack_method=predict_proba</code> Base estimators: LogisticRegression, RandomForest, SVC, GradientBoosting Final estimator: LogisticRegression

^a All classifiers use scikit-learn default parameters unless specified.

^b StackingClassifier's base estimators use the same hyperparameters as the individual classifiers.



Supplementary figure S1: Graphical user interface (GUI) for configuring and executing the *ROOFS* pipeline. In Step 1 (Pipeline configuration), a user should specify the required inputs, including the dataset, chosen FS methods, pipeline parameters, and output settings. Step 2 (Pipeline execution) launches the analysis. The lower panel illustrates the method selection interface. FS methods are organized into core and extended categories depending on the *ROOFS* version. The color coding indicates subtypes based on whether a method selects a signature of fixed or variable size.



Supplementary figure S2: Performance of all benchmarked methods on PIONeeR data (top: FS methods with fixed, user-defined subset size; bottom: FS methods with varying subset size dependent on algorithm thresholds). Blue: embedded methods; violet-blue: resampling-based ensemble methods; green: statistical and distance-based filters; brown: information theory filters; yellow: correlation-based filter; rose: wrappers. **A, B:** Stability AUC in the experiment on the post-VIF PIONeeR dataset. **C, D:** Trade-off between discovery and error in the experiment on the semi-synthetic PIONeeR dataset with a simulated outcome. The true positive rate (TPR) shows the proportion of true predictors correctly identified by each FS method, while the false positive rate (FPR) shows the proportion of non-informative features incorrectly selected. Error bars represent one-half of the standard deviation (SD).