Exploring the STABL Method for Sparse, Reliable Omic Biomarker Discovery

JONAS AMAR
VIVIEN BRANDT
JÉRÉMIE TOUATI

1 INTRODUCTION TO THE STABL METHOD

The discovery of biomarkers from high-dimensional omic data has become a critical task in clinical and biomedical research. However, translating these findings into reliable clinical applications remains challenging. The STABL method, introduced by Julien Hedou and al [1], addresses this issue by proposing a machine learning framework that identifies a sparse, reliable set of biomarkers. This method utilizes noise injection and a data-driven signal-to-noise threshold in multivariate predictive modeling.

The STABL algorithm extends traditional sparse regression methods (SRM), such as Lasso, Elastic Net (EN), Adaptive Lasso (AL), and Sparse Group Lasso (SGL), by integrating artificial features into the feature selection process. The goal is to identify sparse and reliable omic biomarkers while controlling for false discovery rate (FDR).

STABL Method Overview

- **(a) Base Sparse Regression Method (SRM) Selection**: The first step is to choose an SRM, such as Lasso or EN, to serve as the base estimator. This procedure works on the original dataset $X \in \mathbb{R}^{n \times p}$, where n is the number of samples and p is the number of covariates (features). The model assumes a linear relationship between the outcome vector $Y \in \mathbb{R}^n$ and the feature matrix X, parameterized by the vector of coefficients $\beta \in \mathbb{R}^p$, in the form:

$$Y = X\beta + \epsilon$$

  where ϵ is the error term. The set of informative features $S \subset \{1, \dots, p\}$ is defined as those features that are related to the outcome (i.e., $\beta_i \neq 0$).

- **(b) Generation of Artificial Features**: To mitigate overfitting and control FDR, artificial features $\tilde{X} \in \mathbb{R}^{n \times p}$ are generated and appended to the original dataset. These features are constructed through methods like MX knockoffs [2] or random permutations of the original features. This results in an augmented matrix $\mathcal{X} = [X|\tilde{X}] \in \mathbb{R}^{n \times 2p}$, where half of the features are informative (original) and the other half are uninformative (artificial).

- **(c) Subsampling and Model Fitting**: Next, the STABL algorithm performs B subsampling iterations. For each iteration $k \in \{1, \dots, B\}$, a random subsample of size $\lfloor n/2 \rfloor$ is drawn from the original dataset $(Y, \mathcal{X})$, and the chosen SRM (such as Lasso) is applied to this subsample with varying regularization parameters $\lambda \in \Lambda \subset \mathbb{R}_{\geq 0}$. The resulting estimate of the coefficients is denoted as:

$$\hat{\beta}^{(k)}(\lambda) = \arg\min_{b \in \mathbb{R}^{2p}} \left(\|Y - \mathcal{X}b\|_2^2 + \lambda \|b\|_1 \right)$$

- **(d) Feature Selection Frequency and Stability Path**: After performing the B subsampling iterations, the frequency with which each feature is selected is computed. For each feature i, the selection frequency $f_i(\lambda)$ is the proportion of subsamples where the feature is included in the model:

$$f_i(\lambda) = \frac{1}{B} \sum_{k=1}^{B} \mathbb{1}[\hat{\beta}_i^{(k)}(\lambda) \neq 0]$$

  Plotting $f_i(\lambda)$ against $1/\lambda$ for all features results in a "stability path" graph, where features with a higher frequency of selection across subsamples are more stable.

- **(e) Stability Threshold and Selection**: A feature is considered reliable if its maximum selection frequency $\max_\lambda f_i(\lambda)$ exceeds a given threshold $t \in [0, 1]$. The set of stable features is defined as:

$$\hat{S}(t) = \{i \in [p] : \max_{\lambda} f_i(\lambda) \geq t\}$$

- **(f) False Discovery Rate (FDR) Control and Reliability Threshold**: The final step of STABL involves controlling for false discoveries by optimizing the false discovery proportion (FDP). The augmented FDP at a threshold t is defined as:

$$\text{FDP+}(t) = \frac{1 + \sum_{j \in A} \mathbb{1}[f_j \geq t]}{\sum_{j \in O} \mathbb{1}[f_j \geq t] \vee 1}$$

  where A denotes the set of artificial features and O denotes the set of original features. The reliability threshold θ is selected as the value of t that minimizes the augmented FDP:

$$\theta = \arg\min_{t \in [0,1]} \text{FDP+}(t)$$

  At this threshold θ, the final set of selected features $\hat{S}(\theta)$ is used to construct the final predictive model.

Authors' addresses: Jonas Amar, jonas.amar@etu.minesparis.psl.eu; Vivien Brandt, vivien.brandt@polytechnique.edu; Jérémie Touati, jeremie.touati@polytechnique.edu;

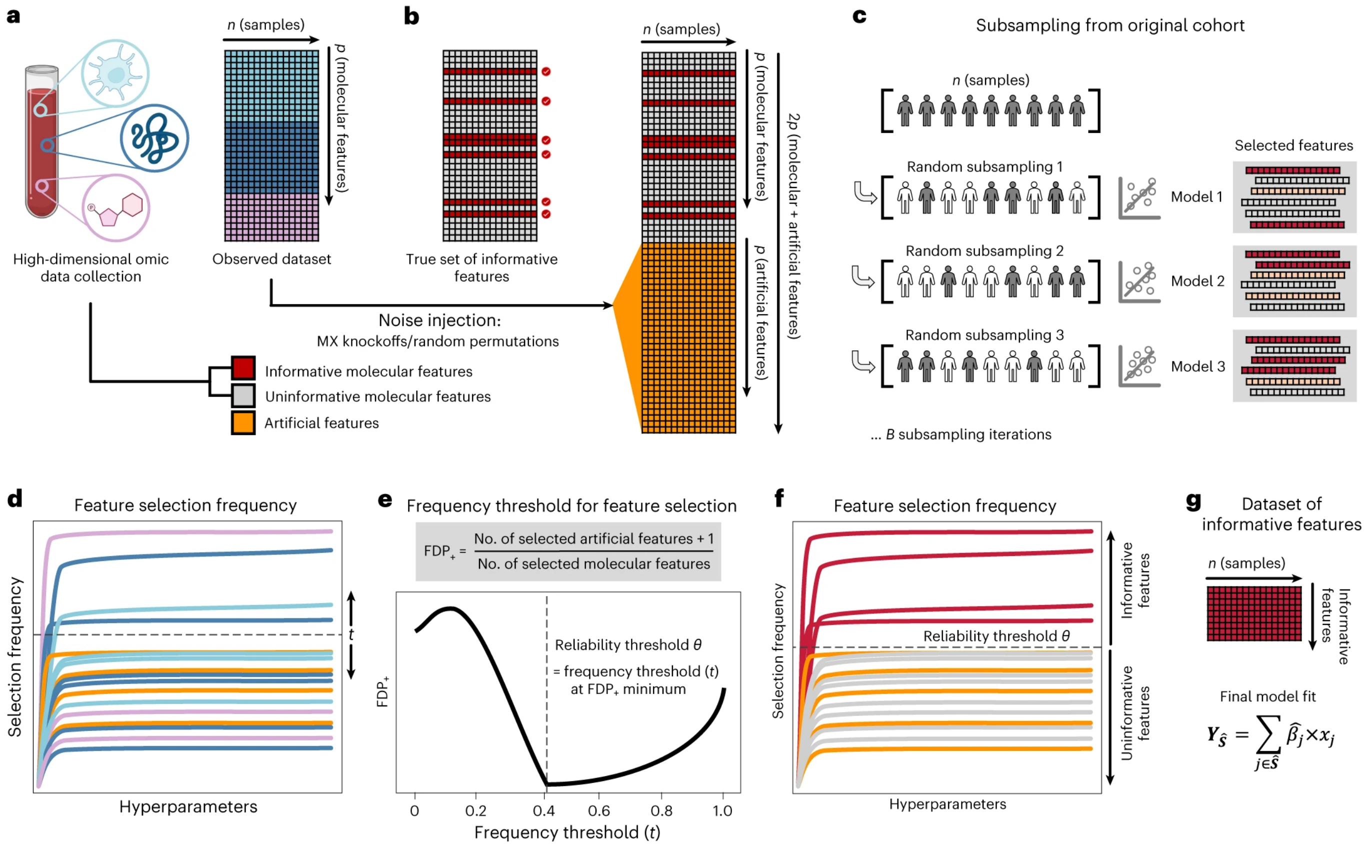

a
High-dimensional omic data collection
n (samples)
p (molecular features)
Observed dataset
b
True set of informative features
n (samples)
p (molecular features)
p (artificial features)
2p (molecular + artificial features)
Noise injection:
MX knockoffs/random permutations
Informative molecular features
Uninformative molecular features
Artificial features
c
Subsampling from original cohort
n (samples)
Selected features
Random subsampling 1
Model 1
Random subsampling 2
Model 2
Random subsampling 3
Model 3
... B subsampling iterations
d
Feature selection frequency
Selection frequency
t
Hyperparameters
e
Frequency threshold for feature selection
FDP+ = No. of selected artificial features + 1 / No. of selected molecular features
FDP+
Reliability threshold θ
= frequency threshold (t) at FDP+ minimum
0
0.2
0.4
0.6
0.8
1.0
Frequency threshold (t)
f
Feature selection frequency
Selection frequency
Reliability threshold θ
Informative features
Uninformative features
Hyperparameters
g
Dataset of informative features
n (samples)
Informative features
Final model fit
$Y_{\hat{S}} = \sum_{j \in \hat{S}} \hat{\beta}_j \times x_j$

Fig. 1. The STABL framework: Main figure illustrating the pipeline for feature selection and stability-based model building.

- **(g) Final Model Construction**: The final model is then built using the selected features $\hat{S}(\theta)$.

2 PROPOSED IMPROVEMENT: REPLACING LINEAR MODELS WITH TREE-BASED MODELS

2.1 Limitation of STABL

One limitation of the STABL method, as originally proposed, is its reliance on linear models for feature selection. This reliance assumes linear relationships between the omic features and the outcome, which may not capture the complex, nonlinear interactions that are often present in biological data.

2.2 Proposed Improvement

To address this limitation, we propose replacing the linear base models with tree-based models, such as XGBoost or Random Forest. These models are capable of capturing nonlinear relationships by learning complex interactions between features without requiring explicit specification of those interactions. By leveraging decision trees, we aim to improve the model's ability to select meaningful biomarkers from datasets with nonlinear patterns.

We modify the STABL framework by replacing the linear regression component with a tree-based model. Taking XGBoost as an example, we would have slight modifications in the following steps:

- **(c) New Model Fitting**: The SRM in step (c) is replaced with XGBoost, a tree-based model. In this case, the hyperparameter space is defined by a combination of parameters specific to XGBoost, such as:
  - `max_depth`: Maximum depth of a tree.
  - `learning_rate` (η): Step size shrinkage used in updates.
  - `min_child_weight`: Minimum sum of instance weights (Hessian) needed in a child.
  - `subsample`: Fraction of samples used for training each tree.

- `colsample_bytree`: Fraction of features used for training each tree.
- `alpha`: L_2-regularization term on weights.
- `lambda`: L_1-regularization term on weights.

For each subsample iteration $k \in \{1, \ldots, B\}$, XGBoost is fitted on a random subsample of the augmented dataset $(Y, \mathcal{X})$ with varying hyperparameter configurations. Instead of selecting features based on the coefficient estimates $(\hat{\beta}^{(k)}(\lambda))$ as in Lasso, we utilize the normalized feature importance of each feature.
The normalized feature importance for feature i in iteration k is given by:

$$\tilde{\texttt{importance}}_i^{(k)} = \frac{\texttt{importance}_i^{(k)}}{\sum_{j=1}^{2p} \texttt{importance}_j^{(k)}}$$

where $\texttt{importance}_i^{(k)}$ quantifies the contribution of feature i to the reduction of the model's loss function during tree splitting.

- In **XGBoost**, $\texttt{importance}_i^{(k)}$ corresponds to the "gain," representing the average improvement in the objective function when feature i is used for a split.
- In **Random Forest**, $\texttt{importance}_i^{(k)}$ measures the average decrease in impurity (e.g., Gini or variance) attributed to feature i across all trees.

Normalization ensures that feature importance scores remain comparable across iterations and hyperparameter settings.

- **(d) New Feature Selection Frequency**: For each feature i, the selection frequency f_i is defined as the average normalized gain across all subsamples. This is computed as:

$$f_i = \frac{1}{B} \sum_{k=1}^{B} \tilde{\texttt{importance}}_i^{(k)}$$

This replaces the original frequency computation used in Lasso-based STABL. Instead of counting the number of times a feature's coefficient is non-zero, XGBoost-STABL and Random Forest-STABL use the aggregated importance scores as a proxy for feature selection frequency. Features with higher aggregated normalized importance scores are considered more stable and informative.

The other steps will stay identical as in the paper.

3 EXPERIMENT DESIGN

The goal of this study was to assess the performance of our proposed improvement to the STABL algorithm compared to the original version in both linear and non-linear settings. Specifically, we aimed to determine whether the improved STABL maintains its performance in linear scenarios while outperforming the original in non-linear cases. The source code can be found at the following link: `STABL with tree-based models`.

3.1 Datasets

To achieve this, we created three synthetic datasets and utilized one real-world dataset:

1. **Linear Synthetic Dataset:** Contains 15 informative variables and 85 noisy variables. The relationship between features and outcomes follows a linear pattern.
2. **Rippling Hyper-Shell Dataset:** Similar in structure to the linear dataset but with a non-linear relationship between features and outcomes.
3. **Toroidal Wave Dataset:** A non-linear dataset where 3 out of 20 features are informative, simulating a toroidal wave function.
4. **Real Dataset (COVID-19):** From the original STABL paper, used as a benchmark to evaluate model performance on real-world data.

All synthetic datasets were generated using the `synthetic-data` library, which enables the creation of datasets with predefined characteristics. The library uses a multivariate normal distribution with specified covariance matrices to model dependencies between features and supports both linear and non-linear data generation. Each synthetic dataset contains 1000 samples and was evaluated for both regression and classification tasks.

The datasets are represented in Figure 2, where we see the nice Toroidal Wave in Figure 2 (b) and a mixed cloud of points for both the Linear and Rippling Hyper-Shell datasets due to the higher number of informative features.

3.2 Experiment Setup

3.2.1 Models and hyperparameter grids. The STABL framework includes hyperparameter tuning for each base model to ensure optimal performance. The following models were evaluated:

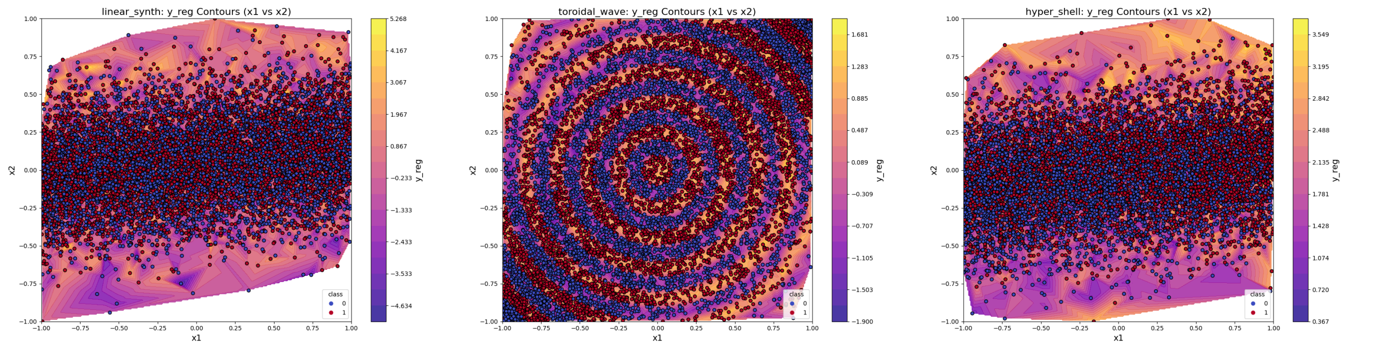
- **Base Models:** Lasso, ElasticNet, Random Forest, and XGBoost.
- **STABL Variants:** STABL Lasso, STABL ElasticNet, STABL Random Forest, and STABL XGBoost.

The hyperparameter grids were defined as follows:

- **Lasso:** `alpha` values were explored on a logarithmic scale: $\{10^{-2}, \ldots, 10^{2}\}$.
- **ElasticNet:** `alpha` values ($\{10^{-2}, \ldots, 10^{2}\}$) and `l1_ratio` ($\{0.5, 0.7, 0.9\}$).
- **Random Forest:** Maximum tree depths ($\{3, 5, 7, 9, 11\}$) were evaluated.
- **XGBoost:** Hyperparameters included `max_depth` ($\{3, 6, 9\}$) and `alpha` ($\{0, 1, 2, 5\}$).

For STABL-based models, hyperparameter grids were applied within the STABL pipeline, which includes bootstrap sampling and feature selection thresholds. All STABL models used 100 bootstraps, an artificial knockoff noise proportion of 1.0, and a false discovery rate threshold range of $\{0.1, \ldots, 1.0\}$.

3.2.2 Evaluation Metrics. To assess the relevance of the selected features, we used:



(a) Projection of the Linear synthetic dataset on (x_1, x_2)

(b) Projection of the Toroidal Wave dataset on (x_1, x_2)

(c) Projection of the Rippling Hyper-Shell dataset on (x_1, x_2)

Fig. 2. Projection of the three synthetic datasets on x_1 and x_2 which are two informative variables.

- **Intersection-over-Union (IoU):** To measure the overlap between the selected features and the true informative features.
- **Number of Selected Features:** To evaluate the sparsity of the feature selection process.

We also focused on predictive performance using:

- **ROC AUC for binary classification tasks:** Area under the Receiver Operating Characteristic curve.
- **R2 for regression tasks:** The R^2 score measures the proportion of variance in y_{true} explained by y_{pred}, comparing the residual sum of squares to the total variance of y_{true}.

All models were validated using 5-fold cross-validation repeated 3 times to ensure robustness of the results.

4 RESULTS AND ANALYSIS

4.1 Results

Table 1 details the IoU (Intersection-over-Union) ratios obtained using Lasso, ElasticNet, Random Forest, XGBoost and the STABL versions of these four models. Results are given for both binary classification and regression tasks. The IoU calculation relies on the assumption that we can distinguish between actually informative and uninformative features. This is not the case in real-life datasets like the COVID-19. We will thus give the results on the three synthetic datasets (Linear, Rippling Hyper-Shell and Toroidal Wave), for which we know the information.

We also give in Table 1 the number of features selected by each model for each dataset on each task. This data allows us to quantify the sparsity of the feature selection. Here, we do include the COVID-19 binary classification task as the number of selected features does not require any knowledge on the informative/uninformative nature of those features. The figures in this table are the mean of IoU and sparsity metrics over the folds of our cross-validation step.

In Table 2, we present the mean performance over the cross-validation folds of the different models after feature selection. We choose the ROC AUC to measure the performance of the binary tasks and the R2 for the regression tasks. Once again, we include the results for the COVID-19 binary task.

4.2 Sparsity analysis

Whereas the standard versions of Lasso and ElasticNet already perform an interesting feature selection by keeping less than 45% of the initial features, this is not the case for the two tree-based models. XGBoost preserves 50% of the features in the binary task of the Rippling Hyper-Shell dataset, and between 88% and 100% for all other task. As for Random Forest, the whole 100% of the features are systematically kept, whatever the dataset. This stems from the fact that Random Forest does not use any L1 regularization term and hence does not penalize the high number of features.

One of the main results from the original paper [1] is STABL's ability to select a sparse number of features. This idea is verified on our three synthetic datasets for the models already studied in the paper, Lasso and ElasticNet. The results show that this conclusion remains true on our tree-based Random Forest and XGBoost models. For instance, for the binary classification on the Linear dataset (100 features), STABL reduces the number of selected features from 45 to 10 on Lasso, from 39 to 10 on ElasticNet, from 100 to 4 on Random Forest and from 88 to 7 on XGBoost. The same conclusion stems out from the two types of tasks on both the Linear and the Rippling Hyper-Shell datasets.

Additionally, we notice that the sparsity obtained with STABL is generally stronger with the tree-based models than with Lasso and ElasticNet. On the regression task for instance, 15 and 15 features are respectively selected with STABL Lasso and STABL ElasticNet for the Linear dataset, against 5 and 5 for Random Forest and XGBoost. For the Rippling Hyper-Shell dataset, we have a similar result with 5 and 5 features selected against 3 and 2. This remark highlights STABL's interest when using non-linear tree-based models: it allows us to achieve a very sparse feature selection with methods that were initially not designed at all for this purpose.

On the Linear dataset, the very few features selected with the STABL versions of Random Forest and XGBoost also account for

Table 1. Mean IoU over cross-validation folds (and mean number of selected features)

Task type	Dataset	Lasso	EN	RF	XGB	STABL Lasso	STABL EN	STABL RF	STABL XGB
Binary	Linear	0.30 (45/100)	0.31 (39/100)	0.15 (100/100)	0.17 (88/100)	0.66 (10/100)	0.59 (10/100)	0.29 (4/100)	0.35 (7/100)
	Rippling Hyper-Shell	0.12 (7/100)	0.15 (5/100)	0.15 (100/100)	0.18 (50/100)	0.12 (4/100)	0.12 (4/100)	0.16 (2/100)	0.26 (5/100)
	Toroidal Wave	0.10 (0/20)	0.10 (0/20)	0.15 (20/20)	0.15 (20/20)	0.17 (10/20)	0.07 (0/20)	0.15 (20/20)	0.16 (14/20)
	COVID-19	(16/1420)	(298/1420)	(208/1420)	(29/1420)	(8/1420)	(20/1420)	(5/1420)	(2/1420)
Regression	Linear	1.00 (15/100)	1.00 (15/100)	0.15 (100/100)	0.15 (98/100)	1.00 (15/100)	1.00 (15/100)	0.33 (5/100)	0.33 (5/100)
	Rippling Hyper-Shell	0.19 (10/100)	0.20 (8/100)	0.15 (100/100)	0.15 (98/100)	0.16 (5/100)	0.17 (5/100)	0.22 (3/100)	0.13 (2/100)
	Toroidal Wave	0.17 (1/20)	0.18 (1/20)	0.15 (20/20)	0.15 (20/20)	0.25 (6/20)	0.25 (6/20)	0.15 (20/20)	0.08 (0/20)

Table 2. Performance of the models after feature selection (ROC AUC for binary tasks and R2 for regression tasks, both averaged over the cross-validation folds)

Task type	Dataset	Lasso	EN	RF	XGB	STABL Lasso	STABL EN	STABL RF	STABL XGB
Binary	Linear	0.998	0.998	0.931	0.972	0.995	0.996	0.925	0.982
	Rippling Hyper-Shell	0.977	0.976	0.971	0.974	0.977	0.976	0.979	0.978
	Toroidal Wave	0.482	0.475	0.481	0.492	0.479	0.495	0.482	0.481
	COVID-19	0.852	0.861	0.860	0.876	0.851	0.869	0.871	0.890
Regression	Linear	0.999	1.000	0.688	0.943	1.000	1.000	0.824	0.911
	Rippling Hyper-Shell	0.851	0.950	0.780	0.931	0.851	0.852	0.851	0.852
	Toroidal Wave	-0.002	-0.002	0.000	-0.074	-0.008	-0.009	-0.018	-0.03

their relatively low IoUs compared to those of STABL Lasso and STABL ElasticNet. Indeed, selecting only 4, 5 or 7 features (compared to 10 or 15) out of 15 actually informative ones causes the IoU to be stuck in [0.29; 0.35] (compared to [0.59; 1.00]). But this does not question the relevance of the selected features. Furthermore, with the Rippling Hyper-Shell dataset, despite the high sparsity obtained with RF and XGB, we succeed in obtaining a notably better IoU than with Lasso and EN. The gain in sparsity is hence not necessarily compensated with a decrease in IoU.

Regarding the Toroidal Wave dataset, STABL tends to keep either all of the 20 features, either none of them (except for the regression task where STABL Lasso and STABL EN both select 6 features). The difficulty that STABL faces with this dataset can be justified by its small number of features. Indeed, STABL is designed to select features in datasets containing several hundreds of attributes. In fact, the paper presenting STABL shows that the ability to select a sparse subset of features reduces when the number of original features decreases.

Finally, the real-life COVID-19 dataset demonstrates how extreme sparsity can be achieved using STABL, especially on tree models. Out of its 1420 features, only 8 are kept using Lasso, 20 using EN, 5 using RF and only 2 using XGB. This corresponds to a 0.14% ratio of selected features over total features.

4.3 Performance analysis

A first notable conclusion that stems out from the results presented in 2 is the conservation of the models performance after STABL's feature selection. Except for ElasticNet (0.950 vs 0.852) and XGBoost (0.852 vs 0.931) on the regression task of the Rippling Hyper-Shell dataset, we observe no notable performance drop between the models versions without and with STABL. Quite the contrary, STABL seems to improve or at least preserve the metrics. On the Linear dataset for the regression task, the RF performs better with STABL (0.824) than without (0.688). Same goes for the Rippling Hyper-Shell dataset (0.851 vs 0.780). We thus show that this profitable characteristic highlighted in STABL's original paper spans to tree-based approaches.

On the Linear dataset, the STABL versions of Lasso and ElasticNet show better performances than those of RF and XGB. For the binary task, they respectively achieve 0.995 and 0.996 of R2 against 0.925 and 0.982. For the regression task, it is also the case with 1.000 and 1.000 against 0.824 and 0.911. We expected the Lasso and EN to be sufficient for this dataset as these models are designed to capture a linear dependence between the features. This justifies why the tree-based models do not work as well.

However, regarding the Rippling Hyper-Shell dataset (whether it is on the binary or the regression task), results obtained with the four STABL models are extremely close (less than 0.002 of difference). RF and XGB even perform slightly better on the binary task. This conclusion is encouraging as our tree approach was thought to capture non-linearities in the data. Unfortunately, there is not significant improvement on those models compared to the linear ones. This may be explained by the higher sparsity of the selected features for such models, preventing them to achieve better performance.

Concerning the Toroidal Wave dataset, all models unfortunately attained performances as low as a dummy model would, with a ROC AUC close to 0.5 in the binary case and a R2 score close to 0 in the regression case. This firstly comes from the difficulty to select a relevant subset of features due to the already small number of features as explained in part 4.2. Also, we can add that the structure of this dataset is particularly complex: it can be hard to model it, even with non-linear based-tree methods with the hyperparameters that we considered for this study.

Finally, on the real-world COVID-19 dataset, the STABL versions of our tree models (0.871 for RF and 0.890 for XGB) succeed in clearly outperforming the other ones (0.851 for Lasso and 0.869 for EN). This result is as impressive as the best score is obtained with XGB using only 2 features (and 5 for RF, compared to 8 and 20 for Lasso and EN). There was a priori no expectation for this dataset to present linearities between its features. In such a case, we thus show that tree-based models together with the STABL framework can be considered as a strong method for relevant feature selection.

5 LIMITATIONS

5.1 Explore other structures for the synthetic datasets

We have underlined how the structure of the dataset had impacted STABL in terms of IoU, sparsity and pure performance. The dataset also accounts for differences in the results between the four models. One first limitation of our study is therefore the small number of synthetic datasets created. For this project we focused on caricatural structures like the Linear dataset and the Toroidal one. Further work could delve into more diverse and more sophisticated relations between the features and the outcome and between the features themselves. The variety of ways to create such sets is infinite and it could be explored beyond what we have done in this work. The goal would be to lead more detailed benchmarks and to determine with more refinement on what type of data which model works the best.

5.2 Extend the parameters grids

Essentially due to limits in terms of computational and time resources, we decided to restrain the size of our grid of parameters for Random Forest and XGBoost. Indeed, the number of models to fit increases exponentially with the number of parameters to vary. We focused on the max_depth parameter for both models and on the L_1 regularization term as well for XGBoost. However, as detailed earlier, those models take as input a considerable amount of other parameters that we could have added to our study. Furthermore, future work could focus on wider ranges of values, allowing to consider more extreme values for every parameter. It is also possible to augment the thinness with which we sample the ranges of values. All these improvements should lead to more robust results and to more consistent analysis of STABL's capacities with tree-based models.

5.3 Use more advanced metrics for feature importance

In our study, the feature importance scores used for the base estimators relied on pre-implemented methods, such as the `feature_importances_` attribute available in scikit-learn and xgboost libraries, due to their computational efficiency and ease of use. However, these scores, which are typically derived from metrics like impurity reduction in Random Forest or gain in XGBoost, may not always provide the most accurate assessment of a variable's importance. An alternative and potentially more robust approach is to leverage Shapley Additive Explanations (SHAP) values [3]. SHAP values quantify each feature's contribution to the model's predictions in a way that considers interactions between features, offering a more nuanced understanding of variable importance. While SHAP provides significant advantages in terms of interpretability and precision, it is computationally expensive, particularly for large datasets or complex models. Future work could explore the integration of SHAP values to refine the feature selection process and better evaluate the impact of each variable, leading to more reliable results in high-dimensional biomarker discovery.

6 CONCLUSION

Our study demonstrates the versatility and effectiveness of the STABL framework when applied to both linear and non-linear datasets. By integrating tree-based models like Random Forest and XGBoost into the STABL pipeline, we showed that it is possible to achieve remarkable sparsity in feature selection without significant performance degradation. In certain cases, such as the regression task on the Linear dataset and the binary task on the COVID-19 dataset, the STABL framework with tree-based models even outperformed traditional methods like Lasso and ElasticNet in terms of predictive accuracy.

The experiments highlight STABL's capability to preserve or enhance model performance while drastically reducing the number of selected features, especially in high-dimensional settings. This is particularly evident in the COVID-19 dataset, where XGBoost-STABL achieved the best ROC AUC score using only 2 features out of 1420, demonstrating the method's ability to identify highly relevant biomarkers in complex real-world datasets.

However, challenges remain, particularly with datasets containing few features or highly intricate relationships, such as the Toroidal Wave dataset. These limitations underscore the need for further exploration of more diverse dataset structures, extended hyperparameter grids, and advanced feature importance metrics like SHAP values to refine the model's capacity for robust and reliable feature selection.

In conclusion, the STABL framework with tree-based models offers a promising avenue for sparse and reliable biomarker discovery, especially in settings where non-linear interactions are prevalent. Future work could build on these findings to optimize and extend the framework for even broader applicability and impact.

REFERENCES

[1] Julien Hedou, Ivana Maric, Grégoire Bellan, Jakob Einhaus, Dyani Gaudilliere, et al. Stabl: Sparse and reliable biomarker discovery in predictive modeling of high-dimensional omic data. *Nature Biotechnology*, 42(1):22–31, 2024.

[2] Emmanuel Candès, Yingying Fan, Lucas Janson, and Jinchi Lv. Panning for gold: 'model-x' knockoffs for high dimensional controlled variable selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(3):551–577, 2018.

[3] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30:4765–4774, 2017.

Table 3. Mean IoU over cross-validation folds (and mean number of selected features)

Task type	Dataset	Lasso	EN	RF	XGB	STABL Lasso	STABL EN	STABL RF	STABL XGB
Binary	Linear	45/100	39/100	100/100	88/100	10/100	10/100	4/100	7/100
	Rippling Hyper-Shell	7/100	5/100	100/100	50/100	4/100	4/100	2/100	5/100
	Toroidal Wave	0/20	0/20	20/20	20/20	10/20	0/20	20/20	14/20
	COVID-19	16/1420	298/1420	208/1420	29/1420	8/1420	20/1420	5/1420	2/1420
Regression	Linear	15/100	15/100	100/100	98/100	15/100	15/100	5/100	5/100
	Rippling Hyper-Shell	10/100	8/100	100/100	98/100	5/100	5/100	3/100	2/100
	Toroidal Wave	1/20	1/20	20/20	20/20	6/20	6/20	20/20	0/20

Table 4. Performance of the models after feature selection (ROC AUC for binary tasks and R2 for regression tasks, both averaged over the cross-validation folds)

Task type	Dataset	Lasso	EN	RF	XGB	STABL Lasso	STABL EN	STABL RF	STABL XGB
Binary	Linear	0.998	0.998	0.931	0.972	0.995	0.996	0.925	0.982
	Rippling Hyper-Shell	0.977	0.976	0.971	0.974	0.977	0.976	0.979	0.978
	Toroidal Wave	0.482	0.475	0.481	0.492	0.479	0.495	0.482	0.481
	COVID-19	0.852	0.861	0.860	0.876	0.851	0.869	0.871	0.890
Regression	Linear	0.999	1.000	0.688	0.943	1.000	1.000	0.824	0.911
	Rippling Hyper-Shell	0.851	0.950	0.780	0.931	0.851	0.852	0.851	0.852
	Toroidal Wave	-0.002	-0.002	0.000	-0.074	-0.008	-0.009	-0.018	-0.03