GEMSS: A Variational Bayesian Method for Discovering Multiple Sparse Solutions in Classification and Regression Problems

Kateřina Henclová*
Datamole†

Václav Šmídl, PhD.
Faculty of Electrical Engineering, Czech Technical University

January 22, 2026

Abstract

Selecting interpretable feature sets in underdetermined ($n \ll p$) and highly correlated regimes constitutes a fundamental challenge in data science, particularly when analyzing physical measurements or sensor data. In such settings, multiple distinct sparse subsets may explain the response equally well. Identifying these alternatives is crucial for generating domain-specific insights of the underlying mechanisms, yet conventional methods typically isolate a single solution, obscuring the full spectrum of plausible explanations.

We present GEMSS[1] (Gaussian Ensemble for Multiple Sparse Solutions), a variational Bayesian framework specifically designed to simultaneously discover multiple, diverse sparse feature combinations. The method employs a sparsifying structured spike-and-slab prior, a mixture of Gaussians to approximate the intractable multimodal posterior, and a Jaccard-based diversity penalty to further control diversity among solutions. Unlike sequential greedy approaches, GEMSS optimizes the entire ensemble of solutions within a single objective function via stochastic gradient descent.

The method is validated on a comprehensive benchmark comprising 128 synthetic experiments across classification and regression tasks. Results demonstrate that GEMSS scales effectively to high-dimensional settings ($p = 5000$) with sample sizes as small as $n = 50$, generalizes seamlessly to continuous targets, and exhibits remarkable robustness to class imbalance and Gaussian noise.

GEMSS is implemented as an open-source Python package suitable for research and industrial R&D.

Keywords: *feature selection, predictive multiplicity, Rashomon effect, variational inference, sparse learning, alternative feature selection, statistically equivalent signatures, Markov boundary multiplicity, ultra-high dimensions.*

1 Introduction

Selecting features that yield interpretable and accurate predictive models constitutes a fundamental challenge in data science. This challenge is exacerbated in settings where the number of features

*katerina.henclova@datamole.ai
†www.datamole.ai
[1] `https://github.com/kat-er-ina/gemss/`

exceeds the number of samples (commonly denoted as $n \ll p$), or where features exhibit high correlation. In such regimes, conventional sparse selection methods often fail to converge or arbitrarily select a single solution from a set of statistically equivalent candidates, thereby obscuring alternative explanatory mechanisms.

These underdetermined problems occur frequently in scientific domains such as omics, as well as in industrial and agricultural applications. In many real-world scenarios, the primary objective of statistical modeling shifts from pure prediction to generating actionable scientific or operational insights. Consequently, practitioners seek a limited set of explanatory variables whose roles can be justified, interpreted, and validated by domain experts.

A critical issue in these contexts is the inherent non-uniqueness of sparse feature selection. There may exist multiple sparse solutions, each offering a distinct (but equally valid) explanation of the observed data. Traditional statistical metrics are often insufficient to adjudicate between these solutions. Instead, presenting multiple candidate hypotheses to a domain expert allows for the utilization of accessible domain knowledge to validate the most plausible mechanism.

GEMSS (Gaussian Ensemble for Multiple Sparse Solutions) addresses the gap between this practical need and the limitations of single-solution modeling. Originally inspired by biomedical applications [12, 11], GEMSS generalizes to a broad set of applied problems and is now actively developed in the context of data-driven agriculture and industry by Datamole.

A particularly relevant use case for GEMSS at Datamole involves R&D projects where understanding the underlying mechanism is as critical as predictive performance. We frequently collaborate with domain experts who possess essential context, yet neither the client nor our team initially knows which subset of features is most informative. In this workflow, feature selection becomes a tool for hypothesis generation and mechanism isolation. By identifying multiple valid feature sets, we facilitate a robust discovery process that serves both business goals and scientific inquiry.

This technical report presents the GEMSS methodology and its open-source implementation. The main contributions are:

- **Methodology:** A variational Bayesian algorithm that employs sparsity-inducing priors and Gaussian mixtures to approximate intractable multimodal posteriors. Unlike sequential greedy methods, GEMSS optimizes for multiple diverse solutions simultaneously using a single objective function.
- **Comprehensive validation:** An extensive empirical benchmark comprising 128 experiments organized into 7 tiers. We evaluate performance across baseline ("$n < p$"), high-dimensional (up to $p = 5000$), sample-rich ("$n \geq p''$), and adverse scenarios (noise, missing values, class imbalance) for both classification and regression problems.
- **Key insights:** Experimental evidence demonstrating that the method scales effectively in $n \ll p$ regimes, generalizes seamlessly to regression tasks, and exhibits remarkable robustness to class imbalance. Furthermore, we identify a hierarchy of stressors, establishing that while the algorithm tolerates noise well, data missingness requires careful management.

The remainder of this report is organized as follows:

- **Section 2** situates GEMSS within the broader landscape of feature selection and the Rashomon effect.
- **Section 3** details the mathematical formulation, the choice of priors, and the variational optimization procedure.

- **Section 4** describes the strategies for extracting candidate solutions from the probabilistic model.
- **Section 5** outlines the methodology for the synthetic data benchmark.
- **Section 6** presents a comprehensive analysis of the experimental results.
- **Section 7** discusses the broader implications, strengths, and limitations of the method.
- **Section 8** summarizes the contributions and outlines future directions.
- **Appendix A** provides tables with detailed performance metrics for the experimental tiers.
- **Appendix B** contains plots that visualize results of the experiments.
- **Appendix C** covers implementation details and the structure of the open-source repository.
- **Appendix D** offers practical recommendations for practitioners using GEMSS on real-world problems.

2 Related work

The problem of feature selection is well-studied, yet the specific challenge of identifying multiple diverse and valid feature sets has only recently gained traction in the broader machine learning community. This problem is known by various names: *alternative feature selection* [3], *predictive multiplicity* [19], the *Rashomon effect* [6], finding *statistically equivalent signatures* [24] or *Markov boundary multiplicity* [22], and even *multimodal optimization* in evolutionary computing [16].

2.1 The Rashomon effect

Conventional methods, such as the Lasso [23], standard Bayesian variable selection [10] or recent models for tabular data like TabPFN [13], typically aim to identify a single "best" subset of features.

This approach effectively collapses the Rashomon set (the set of all models with near-optimal loss [6]) into a single point estimate [9]. Robustness techniques like Stability Selection (SS) [20] further entrench this by aggregating features based on high selection probabilities across subsamples. This approach prioritizes a single "stable core" of features, often discarding valid but competing scientific hypotheses where correlated predictors split their selection probability across runs.

Frameworks such as Model Class Reliance (MCR) [9] and Variable Importance Clouds (VIC) [7] have been developed to characterize the geometry of the Rashomon set.

The MCR framework asks: "Are there other models in a given model class that perform roughly as well as my reference model?", assuming that a reference model has been provided by a user and there is a way to efficiently generate all models in a given class. For example, the algorithm TreeFARMS [25] generates sparse decision trees to enable MCR, resulting in a more robust (interval) estimates of feature importances.

The MCR is fundamentally different from GEMSS which *simultaneously* identifies sparse "candidate solutions" of a presumed linear problem, all without any reference model. Moreover, MCR primarily bounds feature importance ranges, offering intervals for each feature rather than concrete candidate feature sets for downstream modeling.

Similarly, the VIC too is a framework to analyze variable importance in the context of other variables by exploring the Rashomon set.

2.2 Sequential LASSO and iterative masking

A simple heuristic for identifying multiple solutions is to use iterative masking. In this approach, a standard feature selection method is run to find a primary feature set. These selected features are then removed (masked) or heavily penalized in the objective function, and the algorithm is re-run on the residual data to find a secondary, orthogonal solution.

This strategy has been formally extended to ultra-high dimensional settings ($p \gg n$) by Luo et al. [18] in the form of Sequential LASSO, which selects features in steps to handle massive feature spaces.

In the context of causal discovery, Statnikov et al. [22] developed the TIE* family of algorithms, which iteratively constrains the search space to uncover all valid Markov boundaries.

While effective for uncovering nested signatures, these greedy "peeling" strategies enforce strict orthogonality and implicitly assume that valid alternative solutions are disjoint. Consequently, they often struggle to recover overlapping solutions - where a new explanation requires a specific combination of new features *and* previously discovered ones.

GEMSS avoids this limitation by learning a mixture model where components can share features if necessary, driven by the data rather than rigid exclusion constraints.

2.3 Statistically equivalent signatures

In bioinformatics, the search for multiple solutions is framed as finding Statistically Equivalent Signatures (SES) [24, 15]. The SES algorithm identifies multiple minimal feature subsets that are statistically indistinguishable using constraint-based conditional independence testing.

While methodologically aligned with our goals, SES relies on combinatorial search and discrete independence tests, which can be computationally intensive and sensitive to thresholding.

Similarly, methods like the Knockoff filter [4] control the False Discovery Rate for a single feature set but do not aim to dissect the ambiguity of correlated sets.

In contrast, GEMSS operates within a continuous variational probabilistic framework, allowing it to identify multiple diverse candidate solutions (sparse feature sets) simultaneously. These solutions represent distinct explanations with comparable predictive potential, effectively treating the "false positives" of one solution as potential "true positives" of an alternative explanation.

2.4 Alternative feature selection

A direct methodological competitor to our work is Alternative Feature Selection (ALFESE) [3, 2]. ALFESE formalizes the search for diverse feature sets as a sequential constrained optimization problem, where subsequent solutions are penalized for similarity to previous ones.

While effective, ALFESE employs a greedy sequential search strategy. This introduces a specific bias: the discovery trajectory is defined relative to the initial (usually LASSO) solution, potentially missing optimal configurations that are distinct from the start point but similar to each other. Furthermore, sequential methods allocate computational budget serially.

GEMSS distinguishes itself by discovering diverse solutions *simultaneously* via a multimodal variational objective, sharing statistical strength across the ensemble to avoid these blind spots in the search space.

2.5 Evolutionary and multimodal optimization

Finally, evolutionary algorithms have long addressed multimodal optimization through niching techniques, which maintain population diversity to explore multiple local optima [16].

One may think of GEMSS as a Bayesian counterpart to niching. GEMSS shares the objective of maintaining diversity but achieves it primarily through a rigorous Bayesian framework, using mixture models to approximate multimodal posteriors [17]. While the Jaccard penalty in GEMSS is functionally equivalent to fitness sharing in e.g. Particle Swarm Optimization, evolutionary heuristics typically rely on gradient-free search. Consequently, they often scale poorly to high-dimensional parameter spaces ($p > 1000$). GEMSS's formulation enables the use of efficient gradient-based stochastic optimization, facilitating scalability.

Table 1: Comparison of GEMSS with related approaches.

Method	**Primary objective**	**Approach**	**Solution structure**	**Search strategy**
GEMSS	Distinct sparse feature sets	Variational Bayes + mixtures	Probability density	Simultaneous
Sequential Lasso	Orthogonal alternatives	Masking + retraining	Disjoint sets	Sequential, greedy
ALFESE	Sequence of diverse feature sets	Constrained optimization	Discrete sets	Sequential
SES	All equivalent signatures	Causal discovery (CI tests)	Equivalence classes	Combinatorial
MCR / VIC	Bound variable importances	Rashomon set analysis	Interval bounds	Analytical / sampling
SS	Robust core features	Resampling / bootstrapping	Single robust set	Intersection-based
LASSO et al.	A single optimal model	Convex optimization	Single sparse vector	Optimization
Multimodal opt.	Multiple optima	Evolutionary heuristics	Set of candidate solutions	Heuristic optimization

3 Problem formulation and key algorithm components

3.1 Theoretical formulation

Let $X \in \mathbb{R}^{n\times p}$ denote the data matrix, $y \in \mathbb{R}^n$ the target vector, and typically $n \ll p$. The target is to identify m diverse, sparse vectors $\beta^{(1)}, \ldots, \beta^{(m)}$ such that

$$\|X\beta^{(k)} - y\|_2 \leq \varepsilon, \qquad \|\beta^{(k)}\|_0 \leq D$$

for user-chosen fit tolerance $\varepsilon > 0$ and sparsity level $D \ll p$.

3.2 A Bayesian model with sparsifying priors and multimodal posterior

We adopt the linear model

$$y = X\beta + e, \qquad e \sim \mathcal{N}(0, \sigma^2 I)$$

and enforce sparsity by placing a prior on β that *enables a multimodal posterior*.

Prior choice

We choose the structured spike-and-slab (`sss`) as our default prior [1]. It enforces exactly D nonzero entries via a mixture of supports:

$$p(\beta) = \frac{1}{|\mathcal{A}|} \sum_{A\in\mathcal{A}} p_{\text{slab}}(\beta_A)\, p_{\text{spike}}(\beta_{A^c})$$

where each A indicates a support of size D, p_{slab} and p_{spike} are wide and narrow Gaussians, respectively.

For small p, all supports are enumerated; for large p, they are randomly sampled.

As an alternative, the following priors can also be enabled:

- Standard spike-and-slab (`ss`) [10] - less direct sparsity control, does not capture

- Student-t (`student`) [5] - flexible but less interpretable.

For either choice, the resulting posterior

$$p(\beta|X,y) \propto \exp\left(-\tfrac{1}{2\sigma^2}\|y - X\beta\|_2^2\right) p(\beta)$$

exhibits multiple well-separated modes, each corresponding to a plausible sparse explanation.

Note that with priors leading to unimodal posteriors, only single-solution inference is possible.

Posterior approximation

Because the posterior is multimodal, GEMSS uses a mixture of m diagonal Gaussians to simultaneously approximate the multiple modes:

$$q(\beta) = \sum_{k=1}^{m} \alpha_k \mathcal{N}(\beta; \mu^{(k)}, \mathrm{diag}((\sigma^{(k)})^2))$$

with mixture weights $\sum_k \alpha_k = 1, \alpha_k \geq 0$. This design enables recovery of several distinct sparse supports – each mixture component can represent a different optimum.

3.3 Tractable univariate optimization

The ELBO objective

To measure the discrepancy between the true posterior $p(\beta \mid X, y)$ and its approximating Gaussian mixture $q(\beta)$ we use the Kullback–Leibler (KL) divergence:

$$\mathrm{KL}\big(q \,\|\, p\big) = \int q(\beta)\, \log \frac{q(\beta)}{p(\beta \mid X, y)}\, d\beta \;\geq 0.$$

Equivalently, we can write

$$\log p(y \mid X) = \mathcal{L}(q) + \mathrm{KL}\big(q(\beta) \,\|\, p(\beta \mid X, y)\big),$$

where

$$\mathcal{L}(q) = \mathbb{E}_{q(\beta)}\left[\log p(y|X,\beta) + \log p(\beta) - \log q(\beta)\right]$$

is the Evidence Lower Bound (ELBO). Because $\log p(y \mid X)$ does not depend on the variational parameters, maximizing $\mathcal{L}(q)$ is exactly equivalent to minimizing $\mathrm{KL}(q\|p)$.

Therefore, we use ELBO maximization with respect to all variational parameters ($\mu^{(k)}$, $\sigma^{(k)}$, α_k) as our objective.

Diversity regularization

To encourage each mixture component to capture genuinely different solutions, a regularization term based on average Jaccard similarity between supports can be applied:

$$\mathcal{J}_{avg}(q) = \frac{1}{\binom{m}{2}} \sum_{k<l} \frac{|\,\mathrm{supp}(\mu^{(k)}) \cap \mathrm{supp}(\mu^{(l)})|}{|\,\mathrm{supp}(\mu^{(k)}) \cup \mathrm{supp}(\mu^{(l)})|}.$$

The penalty is tractable due to the limited number of mixture components.

Optimization

As a result, the objective function is as follows:

$$\mathcal{L}_{\text{reg}}(q) = \mathcal{L}(q) - \lambda_J \mathcal{J}_{avg}(q),$$

where λ_J is the regularization parameter that governs solution diversity.

All variational parameters are optimized simultaneously using the stochastic gradient descent (Adam optimizer) [14]. The implicit reparameterization trick for mixtures [8] is applied to ensure gradients can be computed efficiently.

Missing data handling

GEMSS natively supports data with missing values. For any given sample, it computes the predictive likelihood using only the features observed for that sample. Missing values are ignored in both loss evaluation and gradient computation. No imputation or sample filtering is required.

3.4 Algorithm pseudocode

Algorithm 1 GEMSS

Require: X, y, number of mixture components m, sparsity D, prior parameters (*must be multimodal*), optimization settings

Initialize $\{\mu^{(k)}, \sigma^{(k)}, \alpha_k\}_{k=1}^{m}$
for iteration $t = 1, 2, \ldots, T$ **do**
  For each $k = 1, \ldots, m$: sample $\beta^{(k)} \sim \mathcal{N}(\mu^{(k)}, \text{diag}[(\sigma^{(k)})^2])$
  Compute $\log p(y|X, \beta^{(k)})$ masking missing features
  Compute ELBO $\mathcal{L}(q)$ and regularization $\mathcal{J}_{avg}(q)$
  Compute (stochastic) gradients via implicit reparametrization
  Update mixture parameters $(\mu^{(k)}, \sigma^{(k)}, \alpha_k)$ using Adam
end for
Output: The $\mu^{(k)}$ and supports $\text{supp}(\mu^{(k)})$ (via thresholding) for each k

4 Solution extraction

In theory, all insignificant features in a component will have their μ converging to 0. In practice, there might be variations. Most notably, the user-defined hyperparameters affecting sparsity might not match the actual truth in every component.

The GEMSS implementation offers three primary ways to extract candidate solutions (feature subsets) from the fitted variational model:

Solution types:

1. **"Full" solutions**
   The most direct extraction method is to select all features in each candidate solution whose mean parameter value (μ) magnitude is nonzero. More precisely, when $|\mu|$ exceeds a given threshold. This approach yields the *full solutions*: complete sets of features with nontrivial importances. These solutions are typically interpretable and comprehensive, but they may include more features than desired for highly sparse interpretation.

2. **"Top" solutions – selection of top few features:**
   For cases requiring strict sparsity, GEMSS provides a method to reduce each *full* solution to exactly D features (the desired sparsity). For each component, the D features with the largest (absolute) μ values – among those passing the initial threshold, if provided – are selected as the *short solutions*. This approach ensures each solution is of desired size, prioritizing interpretability and parsimony.

3. **"Outlier" solutions:**
   GEMSS includes tooling for extracting features considered outliers in the component's importance profile. Features whose importance exceeds a multiple of the standard deviation from the mean (or mean absolute deviation from the median) across all features in a solution ("z-score" outliers) can be reported as the driving factors for that solution. This approach offers more flexibility and solid statistical grounding.

All three modes are accessible, enabling users to balance the trade-off between explainability, strict sparsity, and comprehensiveness in reporting alternative solutions.

5 Experiments on artificial data: methodology

To validate the proposed algorithm concept, we designed a comprehensive experimental benchmark using synthetic data. This approach allows us to precisely measure the algorithm's ability to recover ground truth supports in the presence of correlated features, noise, missing data, and class imbalance conditions where "correct solution" is known by design.

It is important to note that, in these tests, we do *not* aim to evaluate predictive potential of the discovered candidate solutions since that would require incorporating a predictive model and the model's properties and tuning would inherently affect the predictions. However, detailed results of the experiments provided in the repository do contain evaluation of simple linear/logistic regression models, wherever possible.

These experiments are intended as an extended proof of concept. At this point, we do not benchmark GEMSS against other algorithms.

5.1 Experiment reproduction

Summary results of the experiments are provided in `scripts/results` as CSV files. They can be viewed using interactive notebooks in `notebooks/analyze_experiment_results`.

Detailed output for each experiment is given in the `scripts/results` as a TXT file with a corresponding name.

To replicate the entire experiment suite, run `./scripts/run_tiers.ps1` in PowerShell terminal and then view the results in the notebooks.

5.2 Artificial data generation

The data used in experiments were generated to guarantee the existence of multiple distinct sparse solutions that explain the response variable equally well. The generation process, implemented in module `generate_artificial_dataset.py`, constructs data such that the target variable y can be predicted by N_{sol} disjoint sets of features.

Construction of multiple solutions

1. **Support selection:** N_{sol} distinct subsets of features (supports) are selected randomly from the feature space. Each support has fixed size S (sparsity).
2. **Primary solution generation:**
   - For the first support (S_0), feature values X_{S_0} are drawn from a standard normal distribution $\mathcal{N}(0, 1)$.
   - Weights w_0 are drawn from a uniform distribution (range $[2.0, 10.0]$).
   - The latent continuous response is calculated as $y_{latent} = X_{S_0} \cdot w_0$.
3. **Dependent solution generation:** To create alternative explanations, the feature values for subsequent supports ($S_k, k > 0$) are generated as linear combinations of the first support's data: $X_{S_k} = X_{S_0} \cdot C$, where C is a random coefficient matrix.
4. **Weight solving:** The weights w_k for these alternative supports are solved analytically using the pseudo-inverse to ensure that $X_{S_k} \cdot w_k \approx y_{latent}$. This construction ensures that any of the N_{sol} feature sets is a valid predictive model for the target.

Noise and complexity injection

- **Irrelevant features:** All features not belonging to any support are filled with Gaussian noise $\mathcal{N}(0, \sigma_{noise})$, where the default $\sigma_{noise} = 0.1$ (parameter `NOISE_STD`) is used, unless specified otherwise.
- **Signal corruption:** Additional white noise (using the same σ_{noise}) is added to the entire feature matrix X (affecting both relevant and irrelevant features). This prevents the linear relationships from being exact, making the optimization problem non-trivial and representative of real-world measurements.
- **Missing data:** When testing the native missing data handling, a specified ratio of values in X (parameter `NAN_RATIO`) is randomly set to `NaN` post-generation. No imputation is performed; the algorithm must handle these gaps during inference. In other, non-adverse scenarios, we use default value `NAN_RATIO` $= 0$.

Response generation

- **Regression:** The target y is the y_{latent} computed directly from the first generating solution.
- **Binary classification:** The continuous response is passed through a sigmoid function. A threshold is selected to achieve a specific class balance (default 0.5, or variable in Tier 7), and labels are assigned as 0 or 1.

5.3 Structure of experiments

The experimental validation is structured hierarchically. At the base level, we define 7 experimental tiers, which represent broad categories of data scenarios (e.g., high-dimensional, noisy, or unbalanced). These tiers are then recombined into 47 specific test cases, each designed to answer a targeted research question by aggregating relevant experiments across different tiers.

7 tiers

The experiments are organized into 7 tiers, totaling 128 individual experimental configurations. Each tier systematically varies parameters to stress-test specific aspects of the algorithm:

- **Tier 1: Basic validation (18 experiments).** This tier establishes baseline performance on sensibly clean data (noise $\sigma = 0.1$, no missing values). It systematically varies the number of samples (n) and features (p) to cover the transition from $n < p$ to $n \ll p$ regimes with small to medium dimensionality. It tests two low sparsity levels ($k = 3$ and $k = 5$) and serves to verify core algorithm correctness.
- **Tier 2: High-dimensional stress test (9 experiments).** This tier tests scalability and performance in extreme high-dimensional contexts where $p \geq 1000$ and $n \ll p$. It includes tests at the $p = 2000$ boundary and the extreme $p = 5000$ limit, using realistic small/medium sample sizes to assess support recovery in ultra-sparse settings.
- **Tier 3: Sample-rich scenarios (14 experiments).** This tier tests performance in traditional machine learning regimes where information is abundant ($n \geq p$). It includes square matrix scenarios ($n = p$) and high oversampling ($n \gg p$) to ensure the algorithm converges correctly when data is plentiful. This is the only tier where experiments pointed towards using zero Jaccard penalty.
- **Tier 4: Robustness under adversity (22 experiments).** This tier tests stability under severe data quality issues. It uses fixed configurations ($[n = 100, p = 200]$ and $[n = 200, p = 500]$) and systematically varies data quality by introducing high noise ($\sigma \in [0.1, 1.0]$) and high ratios of randomly missing data (up to 50%).
- **Tier 5: Effect of Jaccard penalty (28 experiments).** This tier investigates the influence of the diversity-promoting regularization parameter ($0 \leq \lambda_{Jaccard} \leq 10000$) on support recovery. In this tier, we reduce the number of candidate solutions to bare minimum: number of generating solutions. Considering the limited number of features in each solution, this effectively stresses the algorithm due to lack of "buffer" that would allow for combinations of the correct features in a novel way (novel but legitimate, due to properties of linear vector spaces). In order to achieve perfect recall, the algorithm must diversify optimally.
- **Tier 6: Regression validation (29 experiments).** This tier verifies core functionality for continuous response variables. It mirrors the experimental conditions (sample size, dimensionality, noise, missing data, and sparsity) of Tiers 1, 2, and 4 but uses a continuous regression target instead of binary labels.
- **Tier 7: Class imbalance (8 experiments).** This tier tests stability when dealing with unbalanced binary responses. It assesses robustness to severe class imbalance by testing minority class ratios of 10%, 20%, and 30% across different sample sizes.

47 test cases

The 128 experiments from the 7 tiers are regrouped into 47 distinct test cases. Each test case addresses a specific research question by combining experiments that share common characteristics but differ in a key parameter of interest. These cases often draw data from multiple tiers; for example, robustness analysis combines baseline results from Tier 1 with adverse scenarios from Tier 4.

The test cases are categorized into the following sets:

- **Baseline performance (cases 1–3, 14–16):** Analyzes baseline and overall basic performance on clean data for the most important use cases, including summaries of Tiers 1-3.
- **High dimensions (cases 4–10):** Investigates performance scaling with increasing dimensionality ($p \geq 1000$) and sample size, utilizing data from Tiers 1, 2, and 3.
- **Sample-rich scenarios (cases 11–13):** Examines performance in more traditional problem settings of $n \geq p$, where a single solution usually dominates and makes the search for alternatives harder. (Tier 3).
- **Adversity (cases 17–23):** Evaluates robustness by isolating the effects of varying noise, varying missing data ratios, and their combination across different problem sizes ($n = 100, p = 200$ vs. $n = 200, p = 500$). This set combines baseline experiments from Tier 1 with the stress tests of Tier 4.
- **Jaccard penalty (cases 24–28):** Specifically analyzes the effect of the Jaccard penalty weight on solution diversity for different sparsity levels and problem sizes (Tier 5).
- **Class imbalance (cases 29–31):** Studies the impact of varying the binary response ratio, comparing balanced baselines from Tier 1 with unbalanced scenarios from Tier 7.
- **Regression analysis (cases 32–42):** Mirrors the classification analysis structure for regression problems. It covers baseline performance, scalability to high dimensions, and robustness to noise and missing data, drawing entirely from Tier 6.
- **Regression vs. classification (cases 43–47):** Directly compares performance between binary classification and regression tasks across matched scenarios (basic, high-dimensional, noisy, and incomplete data) by combining results from Tiers 1, 2, 3, 4, and 6.

For detailed description and analysis of each case, see the notebook *analysis_per_testcase.ipynb*.

5.4 Evaluation metrics

To quantitatively assess the quality of the discovered feature sets, we compute a comprehensive suite of coverage metrics. These metrics compare the set of all features found by the algorithm (F) against the ground-truth generating features ($P_{generating}$) for all solutions.

The primary metric used for general performance assessment is the **F1 score**, which provides a balanced view of precision and recall. However, to fully capture the nuances of sparse feature selection in high-dimensional spaces, we rely on a set of metrics defined below.

Note that:

- We do *not* evaluate the individual candidate solutions, since any one that generates the correct subspace is equally valid.
- The performance metrics relate solely to the recovery of support features. They do *not* evaluate quality of predictions. Hence, they all are eligible to describe both classification and regression scenarios.

Main evaluation metrics

We focus on four key indicators to evaluate the success of support recovery:

- **F1 score:** The harmonic mean of precision and recall. It serves as the default single-number summary of performance.
$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$
- **Recall (Sensitivity):** The fraction of true generating features that were successfully identified by the algorithm.
$$\text{Recall} = \frac{|F \cap P_{generating}|}{|P_{generating}|}$$
High recall is critical in scientific discovery to ensure no explanatory mechanism is overlooked.
- **Precision:** The fraction of selected features that are actually relevant.

Additional metrics

Beyond the core metrics, the detailed experiment results include a broader range of statistics to provide deeper insight into specific failure modes. These include:

- **Jaccard Similarity:** Measures the intersection-over-union between the discovered set and the true set.
$$\text{Jaccard} = \frac{|F \cap P_{generating}|}{|F \cup P_{generating}|}$$
This metric provides a strict measure of similarity that penalizes both missed features and false positives equally.
- **Success Index (SI):** In high-dimensional settings ($n \ll p$), achieving high recall is significantly harder as the sparsity of the problem increases. Standard Recall does not account for the "difficulty" of finding the correct features amidst a vast number of irrelevant ones. To address this, we introduce the Success Index, which normalizes the number of correctly identified features by the problem's sparsity ratio.
$$\text{SI} = \frac{\text{Recall}}{\text{Problem Sparsity}} = \frac{|F \cap P_{generating}|/|P_{generating}|}{|P_{generating}|/p} = \frac{p \cdot |F \cap P_{generating}|}{|P_{generating}|^2}$$
where p is the total number of features. The SI effectively measures how many times better the algorithm performs compared to random guessing in a sparse environment. A higher SI indicates successful recovery in a more challenging (higher dimensional) feature space.
- **Adjusted Success Index (ASI):** While SI rewards finding true features in difficult search spaces, it can be trivially maximized by selecting all features (achieving perfect Recall but poor Precision). To penalize solutions that are not selective, we introduce the Adjusted Success Index, which scales SI by the solution's precision.
$$\text{ASI} = \text{Precision} \cdot \text{SI} = \frac{|F \cap P_{generating}|}{|F|} \cdot \frac{p \cdot |F \cap P_{generating}|}{|P_{generating}|^2}$$
This composite metric balances the difficulty of the search (SI) with the efficiency of the result (Precision). A perfect solution (exact recovery) yields an ASI equal to the maximum possible SI ($p/|P_{generating}|$). Conversely, if the algorithm "cheats" by selecting too many features, the drop in Precision will severely penalize the ASI, distinguishing precise recovery from noisy over-selection.

5.5 Solution extraction

The type of solution used was determined as the overall best one per case set (measured by average F1 score), with the exception of 'Jaccard' cases, where the `top` solution type was mandated.

As a result, `outlier (STD = 3.0)` solution type was used in sample-rich and unbalanced scenarios (for both classification and regression problems). This solution type does not require that each solution has the same number of features (the requirement is encoded in the prior setting, though). It can achieve greater recall at the expense of precision. For both these properties, outlier-based solutions are highly desirable in practice.

For all other cases, the `top` solution type was used. This solution extraction method selects exactly S features (matching the known true sparsity) with the highest absolute importance (μ) from each mixture component. This provides a strict test of the model's ranking ability. Using the `top` solutions is also the strictest way of recall evaluation.

5.6 Algorithm tuning

All experiments share the same default settings regarding the prior choice and parameters. The diversifying penalty was tuned slightly (values 0/500/1000) per tiers 1–4 and 6–7. The number of iterations varies (values 3500/4000/4500/5000) depending on the difficulty of a tier. The batch size (values 16/24/32/48/64) and number of candidate solutions (values 3/6/9/12) were set roughly per experiment and follow similar patterns across the whole test suite.

5.7 Known limitations of the test suite

1. In practice, parameters of a prior have a significant effect on performance. Their correct setting empowers the outlier detection to choose quality solutions without strict supervision (like the selection of "top" few features). By using the default setting in all experiments, we sacrifice this property. However, we also verify the stability of a single setup across various problem dimensions.

2. The experiments do not fully account for inherent stochasticity of the algorithm. This is compensated by the number of experiments and consistency of results across many problems.

3. The number of candidate solutions (k) is a parameter with great influence. Unless specified otherwise, we use k to be 2-3x the number of generating solutions. However, the redundancy is mitigated by the algorithm's option to direct multiple mixture components to the same solution and/or to assign zero importance to a component.

4. The primary evaluation of the tests does not include prediction accuracy of the solutions due to additional complexity of the task. However, partial simple evaluations are available in the detailed results and advanced tools are readily provided for users of GEMSS.

6 Experiments on artificial data: results

The following results provide evidence that GEMSS recovers multiple solutions and selects important features in various scenarios with generally favorable recall and precision.

The metrics are calculated based on recovery of the set of the true generating features. There is no evaluation of predictive performance of the feature sets.

Ultra sparse settings, unbalanced responses and high noise pose a challenge but not a major threat. The performance is notably worsened in difficult scenarios with high ratios of missing values.

The following tables in Appendix A provide general performance overviews:

- **Table 12**: various $[n, p]$ combinations under standard setting.
- **Table 5**: average performance metrics, aggregated per cases.
- **Table 6**: counting experiments per case that reach performance thresholds.

Appendix B provides more detailed plots of the experimental results.

All tables and figures were generated in notebook `analysis_per_testcase.ipynb`, using settings in `case_analysis.py`.

6.1 Performance in basic scenarios

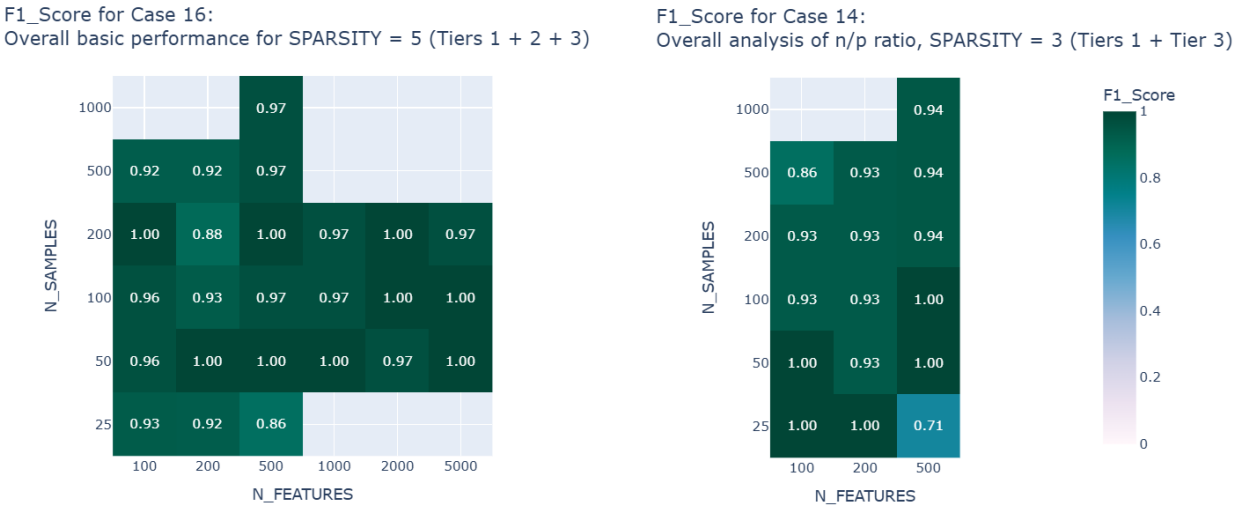
The basic scenarios include baseline problems where $n < p$, high-dimensional settings where $n \ll p$ and the more traditional sample-rich problems where $n \geq p$. All experiments share a default noise level of `NOISE_STD` $= 0.1$ and contain no missing data.

Based on a brief performance comparison, the Jaccard penalty was set to 1000 for the baseline and high-dimensional scenarios, and 0 for sample-rich scenarios. The parameter setup is described in Table 2.

Table 2: Setup of basic experiments

Setup	Baseline	High-dimensional	Sample-rich
No. samples	25, 50, 100, or 200	50, 100, or 200	100, 200, 500, or 1000
No. features	100, 200, or 500	1000, 2000, or 5000 (+ baseline)	100, 200, or 500
Sample/feature ratio	0.05 – 0.4	0.01 – 0.2 (+ baseline)	1.0 – 5.0
Sparsity regimes	3 and 5	5	3 and 5

Figure 1: Summary of F1 scores in basic scenarios. (Cases 14 and 16)



Baseline experiments. The baseline experiments test $n < p$ and $n << p$ scenarios with dimension up to 500 and sample sizes as low as 25.

As can be seen in Figure 1, the performance is consistently high, with a single exception of F1 score < 0.85. The most challenging problem involves 15 generating features (i.e. sparsity = 5), 25 samples, and 500 total features. Even then, the lowest F1 score is 0.71 with a recall of 0.67 (compared to F1 = 0.86 and recall = 0.80 for sparsity = 3). Performance recovers rapidly as the number of samples increases or the total number of features decreases.

For more details, refer to Tables 7 and 8 in Appendix A and Figure 10 in Appendix B.

High dimensions. In these experiments, we test whether the performance holds for dimensions $p \geq 1000$. In addition to the overview in Figure 1, Figure 11 and Table 9 provide further evidence that no performance loss occurs in $n \ll p$ scenarios where the ratio n/p gets as low as $50/5000 = 0.01$. Across all experiments, precision remained at 1.0 while recall exceeded 0.93, resulting in F1 scores ≥ 0.97.

Taken together with the baseline results, this confirms that the previously observed performance decline was driven by insufficient sample size rather than the n/p ratio itself.

Sample-rich scenarios. The sample-rich experiments serve to validate the GEMSS's framework in well-determined settings. The results align with the baselines, achieving perfect F1 scores in nearly all cases (see Figure 1). As visualized in Figure 12, 11 out of 14 experiments attained $F1 = 1.0$, with all experiments maintaining $F1 \geq 0.93$.

Notably, zero Jaccard penalty proves sufficient in these scenarios. The data evidence alone provides a signal strong enough to separate modes without the need for additional diversity promotion.

Finally, as detailed in Table 10, performance metrics show no apparent dependency on the n/p ratio, regardless of specific dimensions or sparsity levels.

Comparison of sparsity regimes. Tiers 1 and 3 allow for a performance comparison between the two sparsity regimes. Theoretically, a default sparsity of 5 offers an advantage due to the higher ratio of generating features to total features (with 3 generating solutions, sparsities of 3 and 5 yield up to 9 and 15 generating features, respectively). Conversely, recovering a larger number of generating features is generally more challenging, particularly with small sample sizes.

Despite these counteracting factors, the evidence in Table 11 indicates that performance remains comparable across both sparsity regimes. For detailed visualization, see Figure 13.

6.1.1 Summary of basic scenarios

Based on the experiments conducted in Tiers 1 through 3, we observe the following:

- **Reliability:** GEMSS demonstrates high reliability across the entire spectrum of standard problem sizes ($25 \leq n \leq 1000$, $100 \leq p \leq 5000$).
- **Scalability:** The algorithm seamlessly transitions from the $n > p$ to the $n \ll p$ regime without degradation, provided the underlying signal remains sparse and sample size sufficient.
- **Sparsity:** There is no significant difference in performance between the two sparsity regimes tested.

- **Precision:** In high-dimensional settings (using the "top" solution type), the algorithm favors precision (often achieving 1.0), meaning identified features are almost certainly relevant, even if some generating features are missed due to strict cut-offs.
- **Failure Modes:** The primary failure mode in clean data is extreme sample scarcity (e.g., 25 samples for 500 features). However, even in these extreme cases, results remain significantly above random chance and the algorithm recovers most generating features. Furthermore, performance improves rapidly with small increases in sample size.
- **Data:** Table 12 provides detailed results for all basic experiments using the default sparsity.

It is worth noting that the "top" solution recovery mode imposes a strict recall criterion. A more detailed analysis reveals that even when generating features were missed in the final selection, they were consistently ranked highly, "merely" failing to meet the cut-off threshold defined by the prescribed number of top features.

Despite this strictness, the "top" strategy yielded the highest F1 scores in both baseline and high-dimensional scenarios. Conversely, the "outlier (STD = 3.0)" strategy proved most effective for sample-rich problems.

6.2 Performance under adversity

The adverse scenarios encompass conditions of high noise, missing data, and a combination of both (Tier 4), as well as scenarios featuring class imbalance (Tier 7). The specific parameter setups are detailed in Table 3.

To ensure consistency, all experiments operate under a single sparsity regime (sparsity = 5) with 3 generating solutions. Two distinct dimensional configurations are evaluated for each scenario: $[n = 100, p = 200]$ and $[n = 200, p = 500]$.

Given the increased complexity of these problems, hyperparameters were adjusted as follows:

- **Iterations:** Increased to 5000 for all experiments.
- **Regularization:** Jaccard penalty fixed at 500.
- **Batch Size:** Scaled in proportion to problem difficulty, ranging from 24 to 64 in Tier 4, and fixed at 64 for Tier 7.

Table 3: Setup of experiments in adverse scenarios

Setup	Noise	NaNs	Noise + NaNs	Imbalance
Noise std	0.1 – 1.0	0.1	0.1 – 0.5	0.1
Ratio of missing values	0.0	0.0 – 0.5	0.0 – 0.5	0.0
Minority class prevalence	0.5	0.5	0.5	0.1 – 0.5

6.2.1 The effect of noise

Problem $[n = 100, p = 200]$**.** When noise is introduced in isolation, performance remains relatively stable between $STD = 0.1$ and $STD = 0.5$, with F1 scores consistently exceeding 0.86. A significant degradation in performance (F1 score = 0.69) is observed only at the extreme noise level of $STD = 1.0$.

However, the presence of missing data exacerbates the negative impact of noise. This synergistic decline is particularly pronounced in this problem setting, likely due to the smaller sample size.

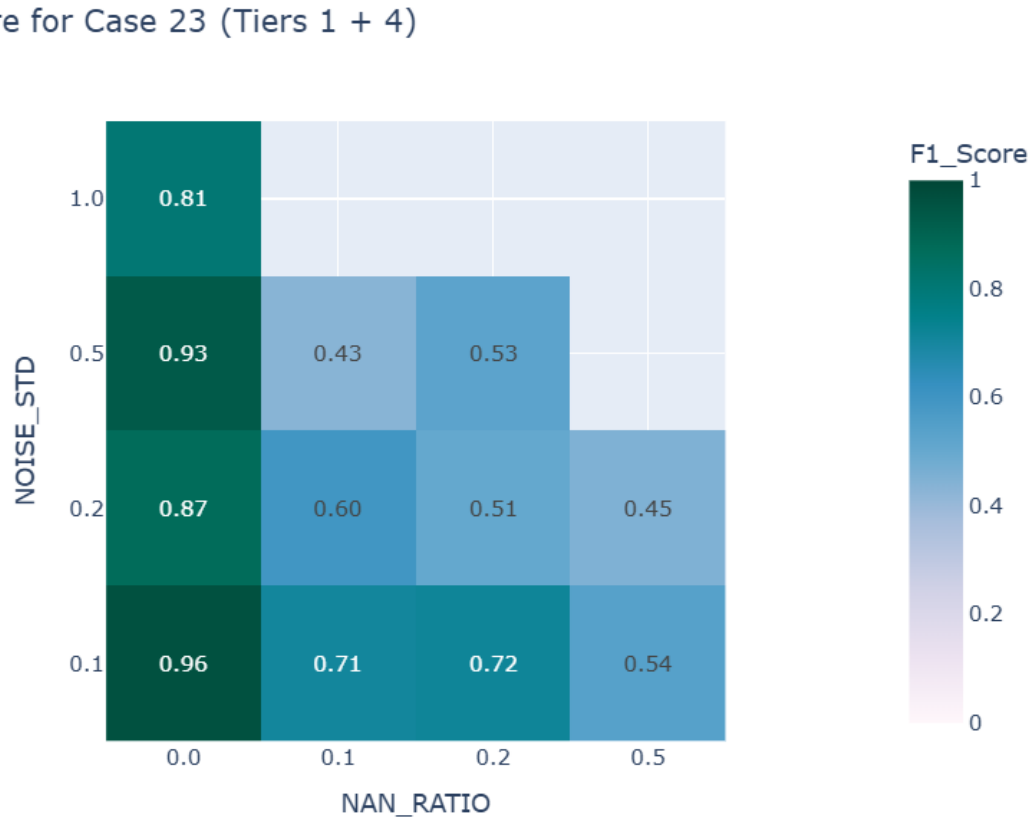
See Figure 14 and Table 13 for details.

Problem $[n = 200, p = 500]$**.** In this configuration, even extreme noise levels exert only a minor negative effect on performance (F1 score ≥ 0.89), provided that missing data is absent or minimal ($\leq 10\%$). This robustness is attributable to the increased sample size compared to the previous problem.

Conversely, when severe data missingness is introduced, performance deteriorates rapidly and exhibits higher volatility. While increasing the noise level does negatively impact results, the magnitude of this effect is generally subordinate to the impact of missing data.

See Figure 15 and Table 14 for details. Figure 2 shows the F1 scores averaged over the two problems.

Figure 2: The effect of noise and missing data on F1 score, averaged over the two problem sizes. (Case 23)



6.2.2 Missing data

Although GEMSS is designed to handle missing data natively, the loss of information severely diminishes F1 scores, particularly when compounded by small sample sizes and elevated noise levels.

Problem $[n = 100, p = 200]$**.** The F1 score drops below 0.6 when either parameter exceeds its default value (`NAN_RATIO` > 0 or `NOISE_STD` > 0.1). However, the results for `NAN_RATIO` ≥ 0.1 are characterized by high variance and poor general performance, making it difficult to establish a strictly monotonic relationship between increased missingness and performance degradation in this specific setting.

See Figure 16 and Table 15 for details.

Problem $[n = 200, p = 500]$**.** In contrast to the smaller problem size, this dataset exhibits a clear negative correlation between F1 scores and the ratio of missing data across all noise levels.

See Figure 17 and Table 16 for details. Figure 2 shows the F1 scores averaged over the two problems.

6.2.3 Class imbalance

For binary classification, we evaluated minority class prevalences of $10\%, 20\%, 30\%$, and 40% against a baseline of 50%, across both problem sizes.

As detailed in Table 18 and Figure 3, high class imbalance does not appear to negatively impact performance.

Unexpectedly, the balanced 50% scenario was the only case to achieve an F1 score < 1 (averaged over the two problems). This anomaly is likely attributable to the global hyperparameter settings being suboptimal for the balanced case, or potentially due to stochastic fluctuations.

This stability suggests that the method's high sample efficiency (demonstrated in $n \ll p$ baselines) extends to the minority class. Since the algorithm can recover features with as few as 25 samples, the effective sample size of the minority class (even at a low prevalence) appears sufficient to drive feature selection, preventing the majority class from dominating the objective function.

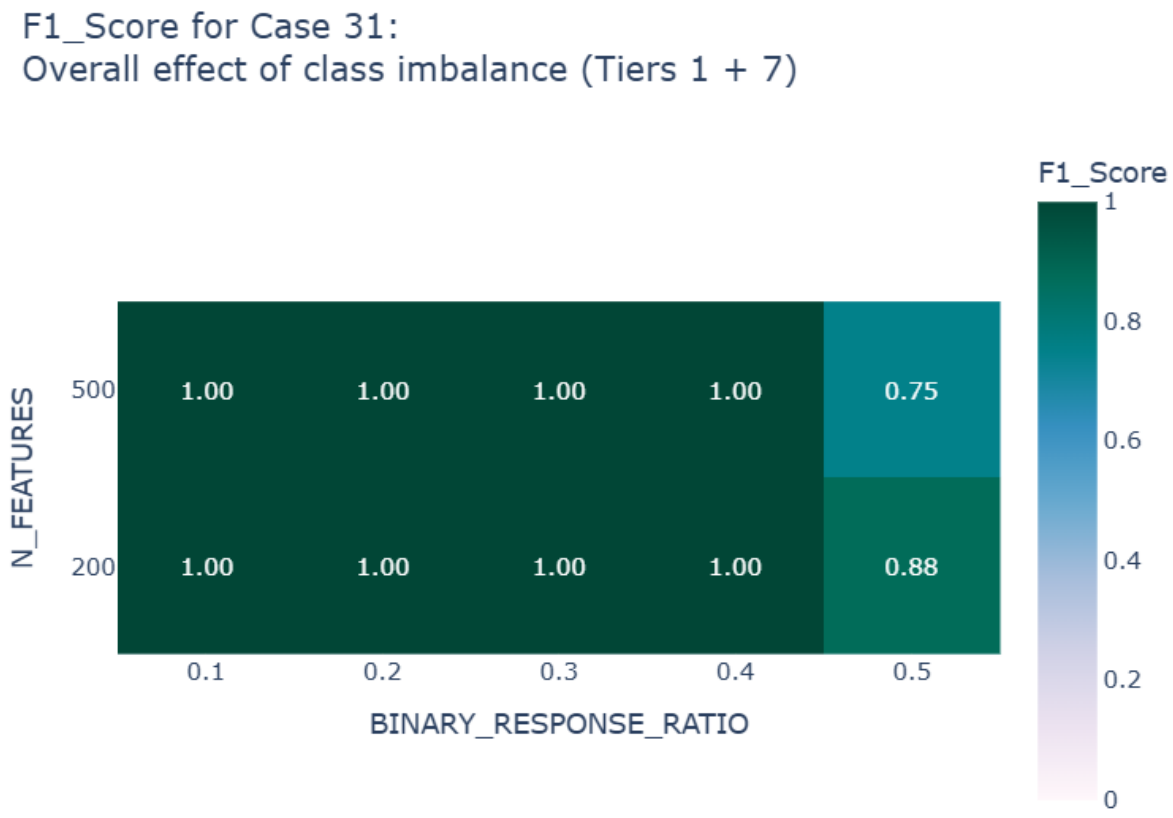
6.2.4 Summary of performance under adversity

Figure 2 displays averaged F1 scores for both problem sizes, when noise level and missingness are varied. Figure 18 further details the interaction between varying noise levels and missing data ratios across the two problem sizes. In 9 out of 12 paired experiments, the larger sample size yields superior F1 scores. This is driven by great difference in precision, while recall does not indicate strong preference for bigger sample size.

Synthesizing results from the noise and missing data experiments (Tables 17, 13, 15) and the class imbalance analysis (Table 18), we draw the following conclusions regarding adverse effects:

- **Hierarchy of stressors:** Missing data is the dominant adverse factor. While the algorithm can tolerate moderate to high noise levels, even a small ratio of missing values (10%) causes significant performance degradation.
- **Compound effects:** The combination of missing data and high noise levels leads to a compound negative effect, reducing performance to poor levels.

Figure 3: Performance on datasets with unbalanced class prevalence. (Case 31).



- **Value of partial recovery:** It is crucial to contextualize the "poor" performance observed in extreme scenarios. Even when F1 scores degrade significantly (e.g., to 0.5), the method often retains a recall of ≈ 0.5. While far from ideal, recovering half of the true generating features offers actionable starting points for domain experts, which is significantly more valuable than complete failure or random guessing (as indicated by Success Index and Adjusted Success Index).
- **Robustness to imbalance:** Contrary to typical classification challenges, GEMSS demonstrates remarkable robustness to class imbalance. Performance remained stable even when minority class prevalence dropped to 10%.
- **Compensation via sample size:** Increased sample size acts as a buffer against adversity. The larger dataset ($n = 200$) maintained performance thresholds under noisy/incomplete conditions where the smaller dataset ($n = 100$) exhibited severe degradation. This suggests that data quantity can partially compensate for deficiencies in data quality.

6.3 The effect of regularization

This section investigates the impact of varying the regularization penalty applied to the inverse Jaccard similarity of solutions (`LAMBDA_JACCARD`). Increasing this penalty encourages the algorithm to identify more diverse sets of features by penalizing overlap. However, too large penalization of the solutions' overlap may lead to the selection of noise features for the sole purpose of diversity.

Setup:

- **Problem dimensions:** $[n = 100, p = 200]$ and $[n = 100, p = 500]$.
- **Sparsity regimes:** 3 and 5.
- **Solution type:** "top".
- **Search constraints:** 3 candidate solutions = 3 generating solutions.

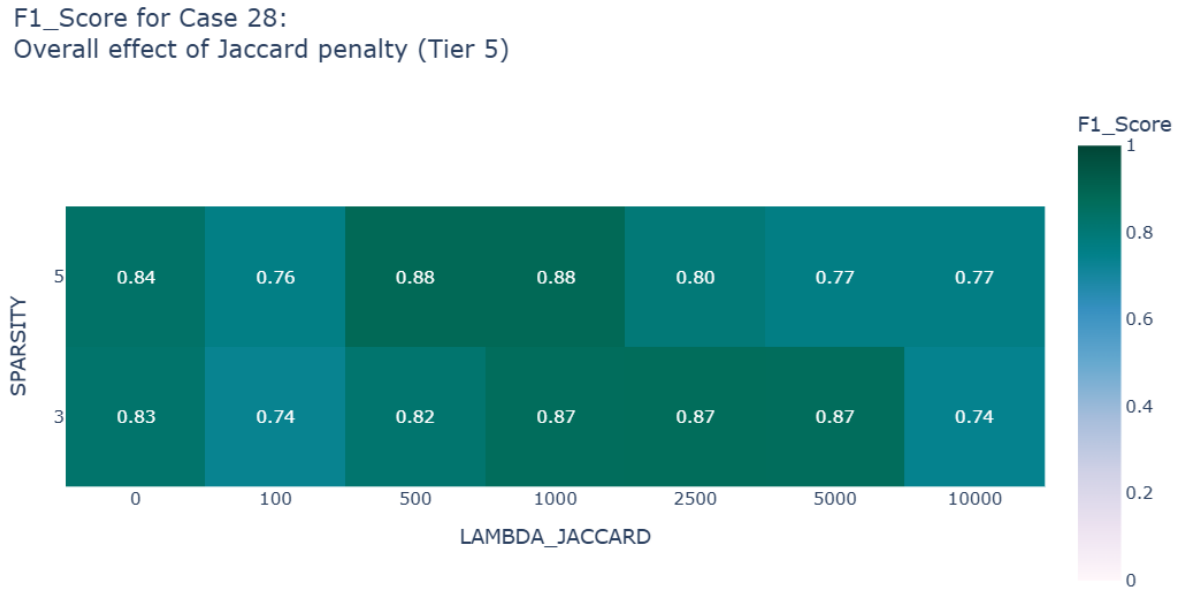
By limiting the number of candidate solutions to exactly match the number of generating solutions, this setup imposes a strict constraint. To achieve perfect recall under these conditions, the algorithm is forced to optimize diversity effectively.

Tables 19 and 20 compare performance across the two sparsity regimes, with each entry representing an average over the two problem configurations. See Figure 4 for a quick overview and Figure19 for more detailed visualization.

Observations:

- **Penalty effect**: As expected, the introduction of a diversity-promoting penalty has a positive effect on performance, particularly on recall.
- **Sparsity dependency:** The benefit of regularization is more pronounced in scenarios with sparser solutions. Vice versa, stricter sparsity requirements necessitate higher penalty values.
- **Parameter sensitivity:** While both under- and over-regularization negatively impact performance, a broad range of intermediate values yields satisfactory results, indicating a stable hyperparameter window.
- **Risk of poor tuning:** Notably, a completely unregularized model may outperform a model with poorly tuned regularization penalties.

Figure 4: Overall effect of Jaccard penalty in two sparsity regimes (Case 28).



Recommendation: To enhance feature recovery and general robustness, it is advisable to decouple the number of candidate solutions from the the number of generating solutions. Specifically, increasing the number of candidate solutions is generally more effective than relying solely on regularization (as evidenced by the performance difference between Tier 5 and Tier 1).

6.4 Regression

Preceding sections focused exclusively on binary classification problems. This section extends the evaluation to regression problems (Tier 6). The experimental framework remains identical to the classification tests, with the sole distinction that the response variable is continuous rather than binarized. Given the richer informational content inherent in continuous targets, we hypothesize that the feature selector will achieve comparable or superior performance.

To ensure conciseness while maintaining comparability with previous results, the experimental setup mirrors Tiers 1, 2, and 4 (baseline, high-dimensional, and adverse scenarios). Specific parameters are detailed in Table 4.

Table 4: Setup of regression experiments

Setup	**Baseline**	**High-dimensional**	**Adversity**
No. samples	25, 50, 100, or 200	50, 100, or 200	100
No. features	100, 200, or 500	1000, 2000, or 5000 (+ baseline)	200
Sparsity regimes	5	5	5
Noise STD	0.1	0.1	0.1 – 1.0
Ratio of missing values	0	0	0 – 0.5

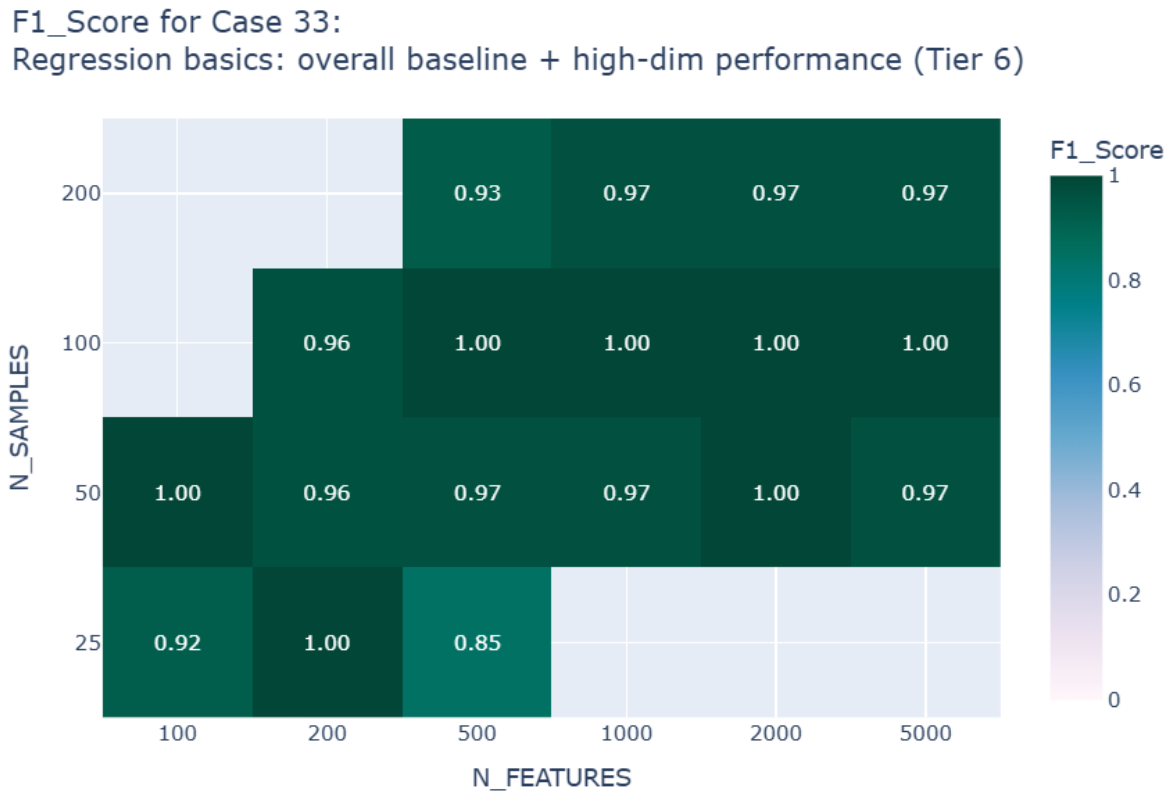
6.4.1 Baseline and high-dimensional scenarios

Remarkably, as can be seen in Figure 5, all baseline and high-dimensional regression scenarios achieved a perfect precision of 1.0. Consequently, the F1 scores are identical to the recall values.

Performance is consistently high across the board (F1 ≥ 0.92), with the sole exception of the most severely undersampled configuration ($[n = 25, p = 500]$). This aligns with the limitations previously observed in the classification experiments.

See Table 21 for more details.

Figure 5: Performance overview for baseline and high-dimensional regression problems (Case 33).



6.4.2 Adverse scenarios

As shown in Figure 6, the adverse scenarios for regression exhibit behavioral patterns analogous to those observed in classification tasks.

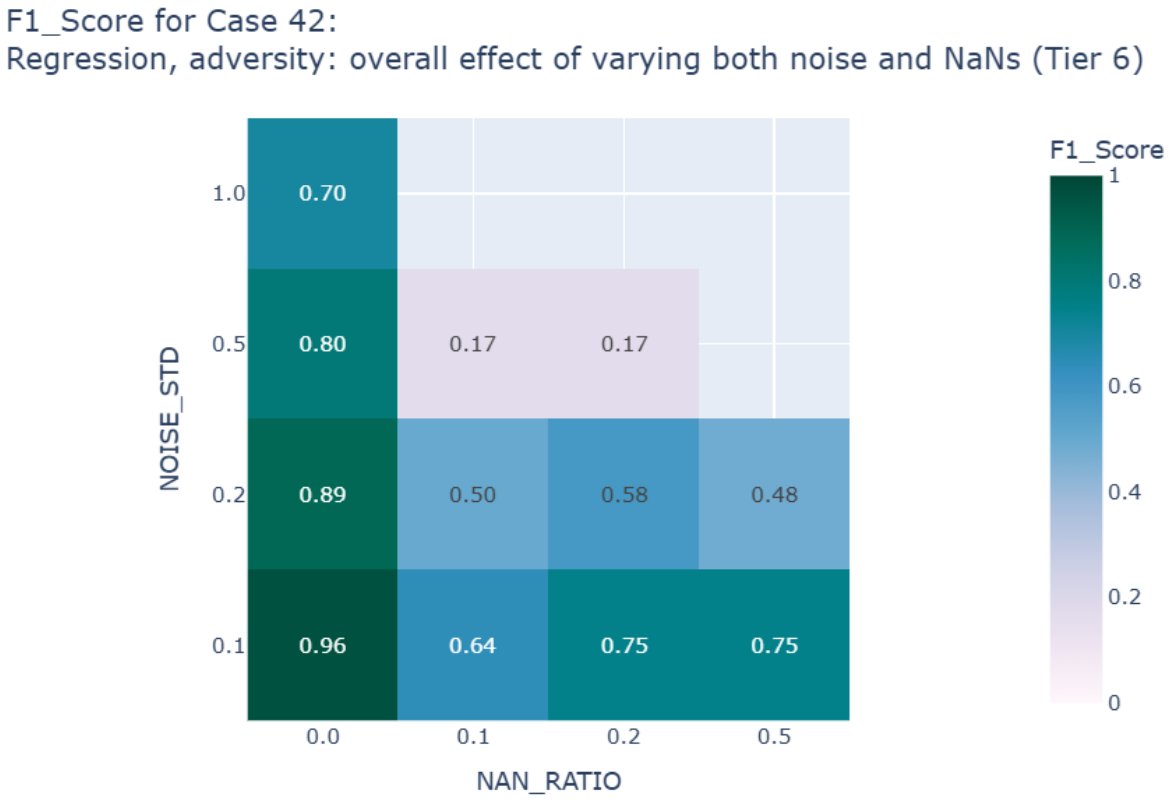
Observations:

- **Missing data sensitivity:** Figure **??** illustrates that missing data is the primary driver of performance degradation. Unlike noise, even moderate ratios of missing values lead to a significant decline in feature recovery.

- **Worsened noise tolerance:** In noisy scenarios, figure **??** suggests that regression problems are more prone to fail than their classification counterparts, provided missing data is minimal.
- **Compound effect:** As shown in the heatmap (Figure 6), the combination of high noise and high missingness creates a "dead zone" where performance drops significantly, mirroring the compound effects seen in Tier 4.

See Table 22 for the detailed results.

Figure 6: Performance for varying levels of noise and missing data (Case 42).



6.4.3 Summary of regression experiments

The evaluation of GEMSS on regression problems confirms the method's versatility.

- **High precision:** A notable finding is the perfect precision (1.0) achieved across all baseline and high-dimensional experiments. This suggests that the continuous nature of the response variable may lead to elimination of false positives in well-posed problems.
- **Consistency:** The failure modes in regression are identical to those in classification. Specifically, extreme undersampling ($n = 25$) and high ratios of missing data.
- **Generalizability:** The successful extension to regression without modification to the core algorithm validates the general applicability of the framework.

6.5 Regression vs. classification

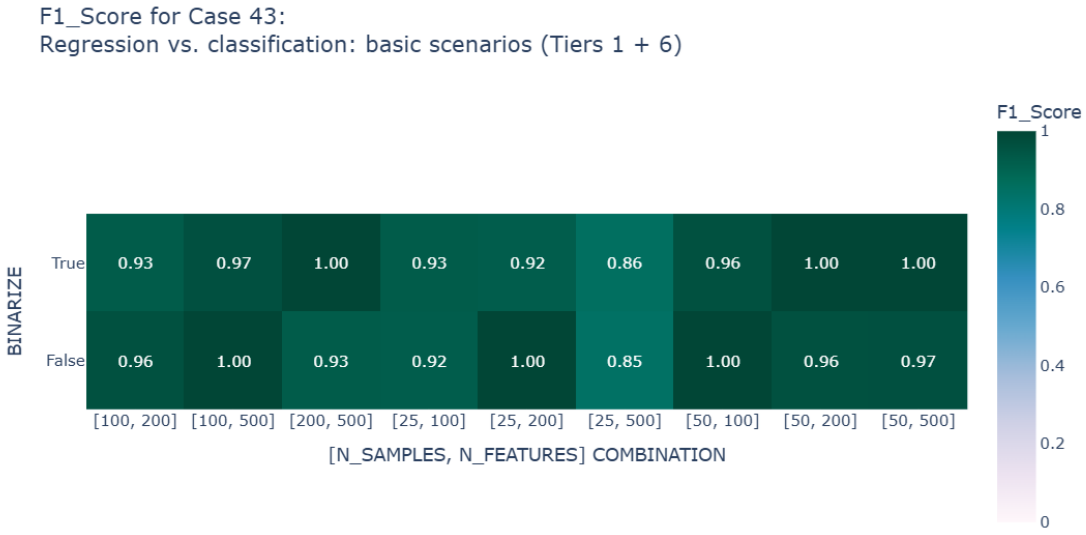
In this section, we compare the performance of GEMSS on regression versus classification tasks across baseline and high-dimensional problems and under adverse conditions.

6.5.1 Baseline and high-dimensional scenarios

Figures 7 and 8 illustrate that baseline and high-dimensional scenarios exhibit similar performance metrics across both task types. While regression achieved perfect precision in all experiments (compared to classification, where 3 instances showed precision < 1, though still > 0.92), the generally higher recall observed in classification tasks resulted in comparable overall F1 scores.

These results do not indicate an inherent difficulty gap between regression and classification formulations in standard settings. For more details, see Figure 20.

Figure 7: Performance comparison of regression and classification on baseline problems (Case 43).



6.5.2 Adverse scenarios

Effect of noise. In scenarios with varying noise levels but no missing data, the F1 score favored regression in 3 out of 4 experiments. As noise levels increased, precision dropped sharply across both formulations, while recall remained comparatively high. Thus, precision became the dominant factor driving the performance differential.

See Figure 21 and Table 23 for more details.

Effect of missing data. Similarly, in experiments with a fixed default noise level and varying ratios of missing data, regression outperformed classification in 3 out of 4 cases with respect to F1 score. In both classification and regression scenarios, the performance decline is more pronounced than in the variable noise experiments. In this context, the primary driver of degradation was a sharp plunge in recall, alongside a decline in precision.

See Figure 22 and Table 24 for more details.

Figure 8: Performance comparison of regression and classification on high-dimensional problems (Case 44).

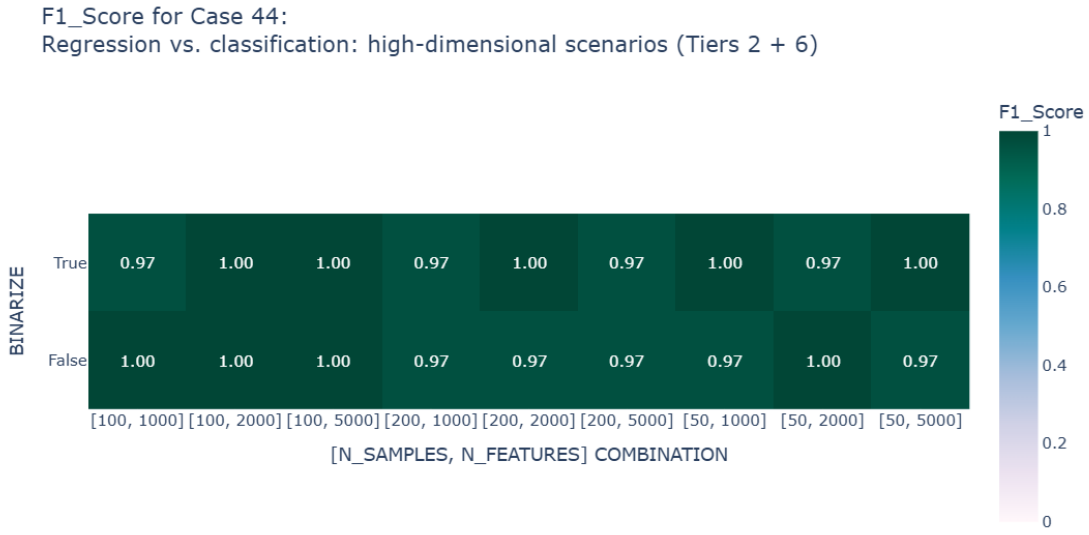
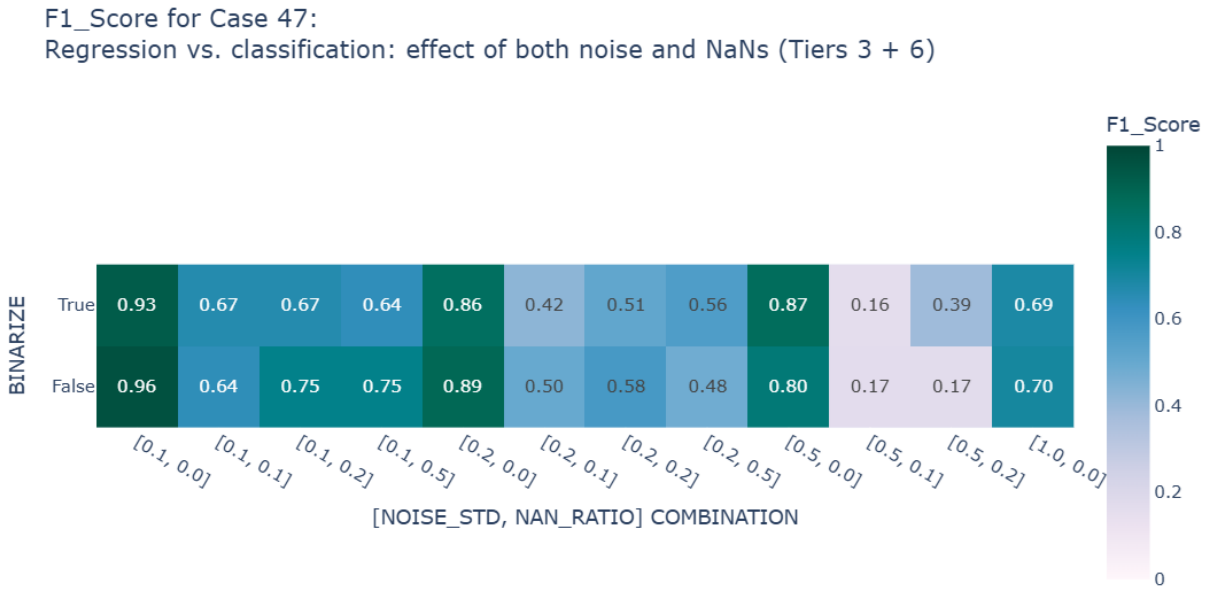


Figure 9: Performance comparison of regression and classification in adverse scenarios (Case 47).



Overall comparison. Despite specific instances of superior performance, the comprehensive overview of scenarios with mixed adversities points toward only a slight superiority for regression, yet the performance differential is marginal (see Figure 9). When averaging over all relevant experiments, classification and regression yield F1 scores 0.614 and 0.617, respectively.

Consequently, the results remain inconclusive; neither classification nor regression demonstrates a consistent advantage over the other under mixed adverse conditions.

See Table 25 and Figure 23 for more details.

6.5.3 Summary of regression vs. classification experiments

Based on the comparative analysis of regression and classification tasks, we observe the following:

- **Equivalence in clean settings:** In both baseline and high-dimensional, both formulations perform comparably with respect to F1 score. Regression offers a slight advantage in precision (consistently 1.0), while classification tends to yield slightly higher recall.
- **Regression robustness:** In isolation, regression appears slightly more robust to specific stressors. It outperformed classification in 75% of cases involving either pure noise or pure missing data.
- **Overall parity:** When subjected to complex, mixed adverse conditions (noise + missing data), the advantage of regression diminishes. The aggregated results suggest that for practical purposes, the choice of formulation (regression vs. classification) is less critical than managing data quality, as neither provides a decisive edge in extreme scenarios.

7 Discussion

The experimental validation across seven tiers demonstrates that GEMSS successfully bridges the gap between purely predictive modeling and the need for multiple, interpretable hypotheses, most notably in underdetermined systems. This section contextualizes the findings and outlines the method's applicability and limitations.

7.1 Measures of performance

Unlike standard predictive modeling competitions, our primary evaluation focuses on the accurate recovery of the underlying generative mechanisms. Therefore, rather than minimizing prediction error, we prioritize metrics regarding the retrieval of generating features: **recall** (to ensure no valid hypothesis is overlooked), **precision** (to minimize false leads), and the **F1 score** of the joint set of all features in the extracted "top" solutions.

7.2 Classification vs. regression

While the primary experimental focus lay on binary classification, comparative analysis confirms that the framework generalizes seamlessly to regression tasks. Empirical results demonstrate that GEMSS achieves comparable but no worse performance when applied to continuous response variables, validating the method's versatility across different target types.

7.3 Applicability in high-dimensional settings

GEMSS is specifically designed for the "$n \ll p$" regime, where the number of features exceeds the number of samples, potentially by orders of magnitude. The results from Tiers 1 and 2 confirm that the method scales effectively to dimensions up to $p = 5000$ with sample sizes as low as $n = 50$.

Crucially, our experiments reveal that performance degradation is driven primarily by absolute sample insufficiency (extreme information poverty) rather than the dimensionality ratio (n/p). Provided there is a sufficient absolute number of samples to capture the signal structure, GEMSS maintains high precision and recall even when the search space expands significantly. This makes the method a strong candidate for omics, sensor data analysis, and other domains characterized by high-dimensional measurements.

7.4 Robustness and hierarchy of stressors

The comprehensive stress-testing in Tier 4 (Adversity) and Tier 7 (Imbalance) establishes a clear hierarchy of data quality factors affecting performance:

1. **Missing Data:** This is the dominant stressor. While the algorithm handles missing values natively without imputation, ratios exceeding 10% lead to a significant drop in the recall of generating solutions.

2. **Noise:** The method exhibits high tolerance to Gaussian noise. Performance degradation is minimal until noise levels become extreme ($STD \geq 1.0$), suggesting that the variational framework effectively filters irrelevant variance.

3. **Class Imbalance:** Contrary to many classification algorithms, GEMSS is remarkably robust to class imbalance, maintaining performance even with minority class prevalence as low as 10%.

It is important to emphasize the **value of partial recovery** in these adverse scenarios. Even when F1 scores degrade (e.g., recall drops to ≈ 0.5 under the mixed stress of severe missingness and noise), the method does retain a subset of the true generating features. For domain experts, identifying half of the true drivers is significantly more actionable than total failure or random guessing, providing a focused starting point for further investigation.

7.5 Mechanisms of diversity and regularization

A core contribution of GEMSS is the ability to discover diverse solutions simultaneously. Beyond the sparsifying prior, users can govern solution diversity by adding a penalty based on the average Jaccard similarity of solutions. Tier 5 investigated the sensitivity of this penalty coefficient.

While the Jaccard penalty effectively enforces diversity and improves performance when properly tuned, relying on it exclusively carries risks if the penalty is set incorrectly. Fortunately, empirical results suggest that the effective hyperparameter window is broad, facilitating practical use.

A critical insight from our validation is that *decoupling the search space from the signal size* (setting the number of candidate solutions m to be higher than the expected number of distinct "true" solutions) is a safer and more robust strategy than aggressive regularization. This allows the mixture model to naturally capture diverse modes without being forced into suboptimal configurations by excessive penalties.

7.6 Solution extraction strategies

The choice of how to interpret the variational posterior is critical. For a detailed guide on configuring these parameters, refer to the practical recommendations in Appendix D.

7.7 Limitations

While promising, the current validation relies on synthetic data generated under linear assumptions. Real-world biological or industrial systems often contain non-linear interactions that a linear spike-and-slab model may only approximate.

It also remains to evaluate quality of the discovered candidate solutions. In the current experimental setup, a candidate solution is equally valid as long as it generates the correct subspace. However, they do not carry a practical meaning.

Additionally, while the native handling of missing data is theoretically sound, the performance drop at high missingness ratios suggests that for datasets with severe gaps ($> 20\%$), specialized imputation strategies might still be necessary prior to feature selection.

Finally, as a variational Bayesian method involving iterative gradient-based optimization, GEMSS is computationally more intensive than greedy heuristics, though this cost is justified by the simultaneous discovery of alternative solutions.

7.8 Handling non-linearity via feature engineering

Although GEMSS is formulated as a linear model, its capacity to handle extreme dimensionality ($n \ll p$) allows it to model non-linear systems effectively through feature engineering.

In many applications, the linear assumption can be relaxed by expanding the feature space with non-linear transformations and interaction terms. While this explosion in dimensionality typically renders standard methods intractable, GEMSS's robustness to correlated features and its ability to prune vast search spaces allows it to identify sparse, non-linear mechanisms within this expanded set. Consequently, the challenge of non-linear modeling is effectively converted into a linear feature selection problem at scale.

8 Conclusion & future directions

This work introduced GEMSS, a variational Bayesian framework designed to address the challenge of alternative feature selection in underdetermined ($n \ll p$) and highly correlated systems. By approximating the multimodal posterior with a mixture of Gaussians and employing a structured spike-and-slab prior, the method simultaneously identifies multiple sparse feature sets that provide distinct, valid explanations for the target variable.

8.1 Summary of contributions

The proposed methodology moves beyond the constraints of greedy, sequential search algorithms by optimizing for solution diversity within a single objective function. Extensive empirical validation on synthetic data demonstrated the algorithm's reliability across a wide spectrum of problem sizes ($25 \leq n \leq 1000$, $100 \leq p \leq 5000$). Key findings include:

- **Scalability:** The method seamlessly transitions from sample-rich to high-dimensional regimes, limited primarily by absolute information content rather than dimensionality ratios.

- **Robustness:** GEMSS exhibits high tolerance to class imbalance and moderately high tolerance to Gaussian noise, offering a viable alternative to standard selection methods in datasets where minority classes drive the signal.
- **Versatility:** The framework generalizes effectively to regression tasks and natively handles missing data, though high missingness ratios remain the primary performance bottleneck.

8.2 Readiness and implementation

The algorithm is implemented as a modular Python package, leveraging stochastic optimization for efficiency. It is currently suitable for deployment in research and industrial R&D contexts, particularly for tasks requiring mechanistic insight alongside predictive modeling.

To facilitate practical application, the package includes integrated extensions for downstream modeling and evaluation using linear/logistic regression or TabPFN on the discovered candidate solutions. Further details on the implementation can be found in Appendix C.

8.3 Future directions

Future development will prioritize transitioning from synthetic benchmarks to pilot studies on real-world datasets. This step is essential to assess practical usability and the impact of adverse effects common in physical systems.

Furthermore, it is highly desirable to establish a benchmarking framework to evaluate GEMSS against other methods.

Finally, we aim to enhance the user interface of GEMSS to facilitate adoption by both data scientists and domain experts. This initiative aligns with our broader mission to bridge the operational gap between advanced data science and domain-specific disciplines.

Acknowledgments

A big thank you is due to the Datamole Data science team members, who supported this research by providing valuable feedback and reviews for both the code and this manuscript. Most notably: Marketa Juzlova, Marek Nevole, and Vojtech Sramek.

During this work, multiple AI tools were utilized as a coding copilot, for finding related literature and for stylistic aid.

References

[1] Michael Riis Andersen, Ole Winther, and Lars Kai Hansen. Bayesian inference for structured spike and slab priors. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 27, 2014.

[2] Jakob Bach. Finding optimal diverse feature sets with alternative feature selection. arXiv:2307.11607 [cs.LG], 2023.

[3] Jakob Bach and Klemens Böhm. Alternative feature selection with user control. *International Journal of Data Science and Analytics*, 2024.

[4] R. Foygel Barber and Emmanuel J. Candès. Controlling the false discovery rate via knockoffs. *The Annals of Statistics*, 43(5):2017–2032, 2015.

[5] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[6] Leo Breiman. Statistical modeling: The two cultures. *Statistical science*, 16(3):199–231, 2001.

[7] Jiayun Dong and Cynthia Rudin. Exploring the cloud of variable importance for the set of all good models. *Nature Machine Intelligence*, 2(12):810–824, 2020.

[8] Mikhail Figurnov, Shakir Mohamed, and Andriy Mnih. Implicit reparameterization gradients. In *Advances in Neural Information Processing Systems*, pages 441–452, 2018.

[9] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, 20(177):1–81, 2019.

[10] Edward I George and Robert E McCulloch. Approaches for bayesian variable selection. *Statistica sinica*, pages 339–373, 1997.

[11] K. Henclová. Little data analysis of bone marrow transplant patients. In *SPMS 2018 Stochastic and Physical Monitoring Systems, Proceedings of the international conference*. ČVUT, Prague, 2018.

[12] Kateřina Henclová. Multisolution approach to classification tasks in biomedicine. In Tomáš Hobza and J. Franc, editors, *SPMS 2020/21 Stochastic and Physical Monitoring Systems, Proceedings of the international conferences*, pages 95–100, Praha, 2021. Ceské vysoké učení technické v Praze.

[13] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations (ICLR)*, 2023.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Vincenzo Lagani, Giorgos Athineou, Alessio Farcomeni, Michail Tsagris, and Ioannis Tsamardinos. Feature selection with the r package mxm: discovering statistically equivalent feature subsets. *Journal of Statistical Software*, 80:1–25, 2017.

[16] Xiaodong Li, Michael G Epitropakis, Kalyanmoy Deb, and Andries Engelbrecht. Seeking multiple solutions: an updated survey on niching methods and their applications. *IEEE Transactions on Evolutionary Computation*, 21(4):518–538, 2016.

[17] Wu Lin, Mohammad Emtiyaz Khan, and Mark Schmidt. Fast and simple natural-gradient variational inference with mixture of exponential-family approximations. *arXiv preprint arXiv:1906.02914*, 2019.

[18] Shan Luo and Zehua Chen. Sequential lasso for feature selection with ultra-high dimensional feature space. *Journal of the American Statistical Association*, 112(519):1189–1200, 2017.

[19] Charles Marx, Flavio du Pin Calmon, and Berk Ustun. Predictive multiplicity in classification. *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

[20] Nicolai Meinshausen and Peter Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

[21] Ofer Michael Shir. Niching in derandomized evolution strategies and its applications in quantum control. 2008.

[22] Alexander Statnikov, Constantin F Aliferis, et al. Algorithms for discovery of multiple markov boundaries. *Journal of Machine Learning Research*, 14:499–566, 2013.

[23] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[24] Ioannis Tsamardinos, Vincenzo Lagani, and Dimitris Pappas. Discovering multiple, equivalent biomarker signatures. *Proceedings of the 7th conference on Pattern Recognition in Bioinformatics*, pages 152–163, 2012.

[25] Rui Xin, Chudi Zhong, Zhi Chen, Takuya Takagi, Margo Seltzer, and Cynthia Rudin. Exploring the whole rashomon set of sparse decision trees. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, pages 14071–14084. Curran Associates, Inc., 2022.

Appendix A: Tabular results of experiments on artificial data

Table 5: Performance metrics overview: mean metrics over all experiments in a case

Case	Description	F1 Score	Recall	Precision
1	Baseline performance for SPARSITY = 3 (Tier 1)	0.946	0.923	0.972
2	Baseline performance for SPARSITY = 5 (Tier 1)	0.952	0.931	0.976
3	Overall baseline performance (Tier 1)	0.949	0.927	0.974
4	Scalability analysis: p = 1000 (Tier 2)	0.977	0.956	1.0
5	Scalability analysis: p = 2000 (Tier 2)	0.989	0.978	1.0
6	Scalability analysis: p = 5000 (Tier 2)	0.989	0.978	1.0
7	Scalability analysis: n = 50 for SPARSITY = 5 (Tiers 1 + 2 + 3)	0.988	0.977	1.0
8	Scalability analysis: n = 100 for SPARSITY = 5 (Tiers 1 + 2 + 3)	0.97	0.954	0.988
9	Scalability analysis: n = 200 for SPARSITY = 5 (Tiers 1 + 2 + 3)	0.969	0.942	1.0
10	Overall high-dim performance (Tier 2)	0.985	0.97	1.0
11	Sample rich performance for SPARSITY = 3 (Tier 3)	0.992	1.0	0.986
12	Sample rich performance for SPARSITY = 5 (Tier 3)	0.986	1.0	0.973
13	Overall sample-rich performance (Tier 3)	0.989	1.0	0.979
16	Overall basic performance for SPARSITY = 5 (Tiers 1 + 2 + 3)	0.962	0.936	0.991
26	Effect of Jaccard penalty for SPARSITY = 3 (Tier 5)	0.819	0.706	0.99
27	Effect of Jaccard penalty for SPARSITY = 5 (Tier 5)	0.815	0.694	0.993
28	Overall effect of Jaccard penalty (Tier 5)	0.817	0.7	0.991
31	Overall effect of class imbalance (Tiers 1 + 7)	0.962	1.0	0.938
32	Regression baseline: baseline performance (Tier 6)	0.954	0.916	1.0
33	Regression basics: overall baseline + high-dim performance (Tier 6)	0.968	0.94	1.0
34	Regression scalability: high-dim p = 1000 (Tier 6)	0.977	0.956	1.0
35	Regression scalability: high-dim p = 2000 (Tier 6)	0.989	0.978	1.0
36	Regression scalability: high-dim p = 5000 (Tier 6)	0.977	0.956	1.0
37	Regression scalability: baseline + high-dim for n = 50 (Tier 6)	0.977	0.955	1.0
38	Regression scalability: baseline + high-dim for n = 100 (Tier 6)	0.993	0.986	1.0
39	Regression scalability: baseline + high-dim for n = 200 (Tier 6)	0.956	0.917	1.0
43	Regression vs. classification: basic scenarios (Tiers 1 + 6)	0.953	0.924	0.988
44	Regression vs. classification: high-dimensional scenarios (Tiers 2 + 6)	0.983	0.967	1.0
45	Regression vs. classification: effect of noise (Tiers 3 + 6)	0.837	0.884	0.806
46	Regression vs. classification: effect of NaNs (Tiers 3 + 6)	0.751	0.696	0.824
47	Regression vs. classification: effect of both noise and NaNs (Tiers 3 + 6)	0.615	0.634	0.615

Table 6: Performance metrics overview: numbers of experiments in each case, whose F1 score reach a predefined performance threshold (Excellent: 0.85, Good: 0.71, Moderate: 0.565, Poor otherwise).

Case	Description	Excellent	Good	Moderate	Poor
1	Baseline performance for SPARSITY = 3 (Tier 1)	8	0	1	0
2	Baseline performance for SPARSITY = 5 (Tier 1)	9	0	0	0
3	Overall baseline performance (Tier 1)	17	0	1	0
4	Scalability analysis: p = 1000 (Tier 2)	3	0	0	0
5	Scalability analysis: p = 2000 (Tier 2)	3	0	0	0
6	Scalability analysis: p = 5000 (Tier 2)	3	0	0	0
7	Scalability analysis: n = 50 for SPARSITY = 5 (Tiers 1 + 2 + 3)	6	0	0	0
8	Scalability analysis: n = 100 for SPARSITY = 5 (Tiers 1 + 2 + 3)	6	0	0	0
9	Scalability analysis: n = 200 for SPARSITY = 5 (Tiers 1 + 2 + 3)	6	0	0	0
10	Overall high-dim performance (Tier 2)	9	0	0	0
11	Sample rich performance for SPARSITY = 3 (Tier 3)	7	0	0	0
12	Sample rich performance for SPARSITY = 5 (Tier 3)	7	0	0	0
13	Overall sample-rich performance (Tier 3)	14	0	0	0
16	Overall basic performance for SPARSITY = 5 (Tiers 1 + 2 + 3)	25	0	0	0
26	Effect of Jaccard penalty for SPARSITY = 3 (Tier 5)	8	5	1	0
27	Effect of Jaccard penalty for SPARSITY = 5 (Tier 5)	4	10	0	0
28	Overall effect of Jaccard penalty (Tier 5)	12	15	1	0
31	Overall effect of class imbalance (Tiers 1 + 7)	9	1	0	0
32	Regression baseline: baseline performance (Tier 6)	8	1	0	0
33	Regression basics: overall baseline + high-dim performance (Tier 6)	17	1	0	0
34	Regression scalability: high-dim p = 1000 (Tier 6)	3	0	0	0
35	Regression scalability: high-dim p = 2000 (Tier 6)	3	0	0	0
36	Regression scalability: high-dim p = 5000 (Tier 6)	3	0	0	0
37	Regression scalability: baseline + high-dim for n = 50 (Tier 6)	6	0	0	0
38	Regression scalability: baseline + high-dim for n = 100 (Tier 6)	5	0	0	0
39	Regression scalability: baseline + high-dim for n = 200 (Tier 6)	4	0	0	0
43	Regression vs. classification: basic scenarios (Tiers 1 + 6)	17	1	0	0
44	Regression vs. classification: high-dimensional scenarios (Tiers 2 + 6)	18	0	0	0
45	Regression vs. classification: effect of noise (Tiers 3 + 6)	5	1	2	0
46	Regression vs. classification: effect of NaNs (Tiers 3 + 6)	2	2	4	0
47	Regression vs. classification: effect of both noise and NaNs (Tiers 3 + 6)	5	3	7	9

Table 7: Baseline performance in Tier 1 for SPARSITY = 3 (Case 01).

$(n\ p)$	F1 Score	ASI	Recall	Jaccard	Precision	SI
(25, 100)	1.0	12.5	1.0	1.0	1.0	12.5
(50, 100)	1.0	12.5	1.0	1.0	1.0	12.5
(25, 200)	1.0	25.0	1.0	1.0	1.0	25.0
(50, 200)	0.933	21.875	0.875	0.875	1.0	21.875
(100, 200)	0.933	21.875	0.875	0.875	1.0	21.875
(25, 500)	0.706	27.778	0.667	0.545	0.75	37.037
(50, 500)	1.0	55.556	1.0	1.0	1.0	55.556
(100, 500)	1.0	55.556	1.0	1.0	1.0	55.556
(200, 500)	0.941	49.383	0.889	0.889	1.0	49.383

Table 8: Baseline performance in Tier 1 for SPARSITY = 5 (Case 02).

$(n\ p)$	F1 Score	ASI	Recall	Jaccard	Precision	SI
(25, 100)	0.929	6.159	0.929	0.867	0.929	6.633
(50, 100)	0.963	6.633	0.929	0.929	1.0	6.633
(25, 200)	0.923	12.245	0.857	0.857	1.0	12.245
(50, 200)	1.0	14.286	1.0	1.0	1.0	14.286
(100, 200)	0.929	12.318	0.929	0.867	0.929	13.265
(25, 500)	0.857	24.615	0.8	0.75	0.923	26.667
(50, 500)	1.0	33.333	1.0	1.0	1.0	33.333
(100, 500)	0.966	31.111	0.933	0.933	1.0	31.111
(200, 500)	1.0	33.333	1.0	1.0	1.0	33.333

Table 9: High-dimensional scenarios: scalability by n/p ratio (Case 10).

n/p ratio	F1 Score	ASI	Recall	Jaccard	Precision	SI
0.01	1.0	333.333	1.0	1.0	1.0	333.333
0.02	1.0	333.333	1.0	1.0	1.0	333.333
0.025	0.966	124.444	0.933	0.933	1.0	124.444
0.04	0.966	311.111	0.933	0.933	1.0	311.111
0.05	1.0	100.0	1.0	1.0	1.0	100.0
0.1	0.983	97.778	0.967	0.967	1.0	97.778
0.2	0.966	62.222	0.933	0.933	1.0	62.222

Table 10: Sample-rich scenarios, average over both sparsity regimes (Case 13).

n/p ratio	F1 Score	ASI	Recall	Jaccard	Precision	SI
1.0	0.995	24.289	1.0	0.99	0.99	24.636
2.0	0.987	25.744	1.0	0.975	0.975	27.133
2.5	0.967	18.75	1.0	0.938	0.938	19.643
5.0	1.0	9.821	1.0	1.0	1.0	9.821

Table 11: Comparison of two sparsity regimes (Case 14 vs. Case 15). Bold font denotes the greater value of a pair.

$(n\ p)$	F1 Score		Recall		Precision	
sparsity regime	3	5	3	5	3	5
(25, 100)	**1.0**	0.929	**1.0**	0.929	**1.0**	0.929
(50, 100)	**1.0**	0.963	**1.0**	0.929	**1.0**	**1.0**
(25, 200)	**1.0**	0.923	**1.0**	0.857	**1.0**	**1.0**
(50, 200)	0.933	**1.0**	0.875	**1.0**	**1.0**	**1.0**
(100, 200)	**0.933**	0.929	0.875	**0.929**	**1.0**	0.929
(25, 500)	0.706	**0.857**	0.667	**0.8**	0.75	**0.923**
(50, 500)	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
(100, 500)	**1.0**	0.966	**1.0**	0.933	**1.0**	**1.0**
(200, 500)	0.941	**1.0**	0.889	**1.0**	**1.0**	**1.0**
(100, 100)	0.933	**0.963**	0.875	**0.929**	**1.0**	**1.0**
(200, 100)	0.933	**1.0**	0.875	**1.0**	**1.0**	**1.0**
(500, 100)	0.857	**0.923**	0.75	**0.857**	**1.0**	**1.0**
(200, 200)	**0.933**	0.88	**0.875**	0.786	**1.0**	**1.0**
(500, 200)	**0.933**	0.923	**0.875**	0.857	**1.0**	**1.0**
(500, 500)	0.941	**0.966**	0.889	**0.933**	**1.0**	**1.0**
(1000, 500)	0.941	**0.966**	0.889	**0.933**	**1.0**	**1.0**

Table 12: Overview of performance in basic experiments in tiers 1–3 for common sparsity = 5. (Case 16)

$(n\ p)$	F1 Score	ASI	Recall	Jaccard	Precision	SI
(25, 100)	0.929	6.159	0.929	0.867	0.929	6.633
(50, 100)	0.963	6.633	0.929	0.929	1.0	6.633
(100, 100)	0.963	6.633	0.929	0.929	1.0	6.633
(200, 100)	1.0	7.143	1.0	1.0	1.0	7.143
(500, 100)	0.923	6.122	0.857	0.857	1.0	6.122
(25, 200)	0.923	12.245	0.857	0.857	1.0	12.245
(50, 200)	1.0	14.286	1.0	1.0	1.0	14.286
(100, 200)	0.929	12.318	0.929	0.867	0.929	13.265
(200, 200)	0.88	11.224	0.786	0.786	1.0	11.224
(500, 200)	0.923	12.245	0.857	0.857	1.0	12.245
(25, 500)	0.857	24.615	0.8	0.75	0.923	26.667
(50, 500)	1.0	33.333	1.0	1.0	1.0	33.333
(100, 500)	0.966	31.111	0.933	0.933	1.0	31.111
(200, 500)	1.0	33.333	1.0	1.0	1.0	33.333
(500, 500)	0.966	31.111	0.933	0.933	1.0	31.111
(1000, 500)	0.966	31.111	0.933	0.933	1.0	31.111
(50, 1000)	1.0	66.667	1.0	1.0	1.0	66.667
(100, 1000)	0.966	62.222	0.933	0.933	1.0	62.222
(200, 1000)	0.966	62.222	0.933	0.933	1.0	62.222
(50, 2000)	0.966	124.444	0.933	0.933	1.0	124.444
(100, 2000)	1.0	133.333	1.0	1.0	1.0	133.333
(200, 2000)	1.0	133.333	1.0	1.0	1.0	133.333
(50, 5000)	1.0	333.333	1.0	1.0	1.0	333.333
(100, 5000)	1.0	333.333	1.0	1.0	1.0	333.333
(200, 5000)	0.966	311.111	0.933	0.933	1.0	311.111

Table 13: Performance for varying noise STD and default (none) missing data. Problem [$n = 100, p = 200$]. (Case 17)

Noise STD	F1 Score	ASI	Recall	Jaccard	Precision	SI
0.1	0.929	12.318	0.929	0.867	0.929	13.265
0.2	0.857	10.496	0.857	0.75	0.857	12.245
0.5	0.867	10.778	0.929	0.765	0.812	13.265
1.0	0.688	6.859	0.786	0.524	0.611	11.224

Table 14: Performance for varying noise STD and default (none) missing data. Problem [$n = 200, p = 500$]. (Case 20)

Noise STD	F1 Score	ASI	Recall	Jaccard	Precision	SI
0.1	1.0	33.333	1.0	1.0	1.0	33.333
0.2	0.889	26.667	0.8	0.8	1.0	26.667
0.5	1.0	33.333	1.0	1.0	1.0	33.333
1.0	0.929	28.889	0.867	0.867	1.0	28.889

Table 15: Performance for varying ratio of missing data and default noise level. Problem [$n = 100, p = 200$]. (Case 18)

Ratio of NaNs	F1 Score	ASI	Recall	Jaccard	Precision	SI
0.0	0.929	12.318	0.929	0.867	0.929	13.265
0.1	0.667	6.358	0.643	0.5	0.692	9.184
0.2	0.667	6.531	0.571	0.5	0.8	8.163
0.5	0.64	5.937	0.571	0.471	0.727	8.163

Table 16: Performance for varying ratio of missing data and default noise level. Problem [$n = 200, p = 500$]. (Case 21)

Ratio of NaNs	F1 Score	ASI	Recall	Jaccard	Precision	SI
0.0	1.0	33.333	1.0	1.0	1.0	33.333
0.1	0.75	20.0	0.6	0.6	1.0	20.0
0.2	0.769	20.202	0.667	0.625	0.909	22.222
0.5	0.444	6.772	0.533	0.286	0.381	17.778

Table 17: Performance for all combinations of missing data and noise levels. Every line is an average performance over the two problem sizes. (Case 23) Highlighted values denote default settings for the respective parameters.

Noise STD	Ratio of NaNs	F1 Score	ASI	Recall	Jaccard	Precision	SI
0.1	**0.0**	0.964	22.826	0.964	0.933	0.964	23.299
0.2	**0.0**	0.873	18.581	0.829	0.775	0.929	19.456
0.5	**0.0**	0.933	22.056	0.964	0.882	0.906	23.299
1.0	**0.0**	0.808	17.874	0.826	0.695	0.806	20.057
0.1	0.1	0.708	13.179	0.621	0.55	0.846	14.592
0.1	0.2	0.718	13.366	0.619	0.562	0.855	15.193
0.1	0.5	0.542	6.355	0.552	0.378	0.554	12.971
0.2	0.1	0.597	11.417	0.583	0.447	0.639	14.683
0.5	0.1	0.434	8.621	0.507	0.317	0.381	14.864
0.2	0.2	0.507	6.156	0.555	0.34	0.484	12.37
0.5	0.2	0.528	8.605	0.617	0.371	0.465	15.794
0.2	0.5	0.451	4.273	0.45	0.298	0.468	10.238

Table 18: Performance on datasets with unbalanced class prevalence (Case 31). Each row is an average of two problem.

binary response ratio	F1 Score	ASI	Recall	Jaccard	Precision	SI
0.1	1.0	23.81	1.0	1.0	1.0	23.81
0.2	1.0	23.81	1.0	1.0	1.0	23.81
0.3	1.0	23.81	1.0	1.0	1.0	23.81
0.4	1.0	23.81	1.0	1.0	1.0	23.81
0.5	0.812	15.556	1.0	0.689	0.689	23.81

Table 19: Jaccard penalty testing for SPARSITY = 3 regime (Case 26). Average values over two problem dimensions. Best values per column are printed in bold.

$\lambda_{Jaccard}$	F1 Score	ASI	Recall	Jaccard	Precision	SI
0	0.829	27.894	0.708	0.708	**1.0**	27.894
100	0.736	21.721	0.597	0.597	**1.0**	21.721
500	0.822	29.417	0.701	0.701	**1.0**	29.417
1000	0.866	30.980	0.764	0.764	**1.0**	30.980
2500	**0.871**	**32.727**	**0.819**	**0.778**	0.929	**34.066**
5000	0.866	30.980	0.764	0.764	**1.0**	30.980
10000	0.742	23.245	0.590	0.590	**1.0**	23.245

Table 20: Jaccard penalty testing for SPARSITY = 5 regime (Case 27). Average values over two problem dimensions. Best values per column are printed in bold.

$\lambda_{Jaccard}$	F1 Score	ASI	Recall	Jaccard	Precision	SI
0	0.840	17.324	0.724	0.724	**1.0**	17.324
100	0.764	15.193	0.619	0.619	**1.0**	15.193
500	**0.884**	**18.946**	**0.793**	**0.793**	**1.0**	**18.946**
1000	**0.884**	**18.946**	**0.793**	**0.793**	**1.0**	**18.946**
2500	0.798	16.355	0.688	0.667	0.95	16.814
5000	0.766	14.592	0.621	0.621	**1.0**	14.592
10000	0.766	14.592	0.621	0.621	**1.0**	14.592

Table 21: Performance for baseline and high-dimensional experiments for regression problems in Tier 6 (Case 33).

$(n\ p)$	F1 Score	ASI	Recall	Jaccard	Precision	SI
(25, 100)	0.923	6.122	0.857	0.857	1.0	6.122
(50, 100)	1.0	7.143	1.0	1.0	1.0	7.143
(25, 200)	1.0	14.286	1.0	1.0	1.0	14.286
(50, 200)	0.963	13.265	0.929	0.929	1.0	13.265
(100, 200)	0.963	13.265	0.929	0.929	1.0	13.265
(25, 500)	0.846	24.444	0.733	0.733	1.0	24.444
(50, 500)	0.966	31.111	0.933	0.933	1.0	31.111
(100, 500)	1.0	33.333	1.0	1.0	1.0	33.333
(200, 500)	0.929	28.889	0.867	0.867	1.0	28.889
(50, 1000)	0.966	62.222	0.933	0.933	1.0	62.222
(100, 1000)	1.0	66.667	1.0	1.0	1.0	66.667
(200, 1000)	0.966	62.222	0.933	0.933	1.0	62.222
(50, 2000)	1.0	133.333	1.0	1.0	1.0	133.333
(100, 2000)	1.0	133.333	1.0	1.0	1.0	133.333
(200, 2000)	0.966	124.444	0.933	0.933	1.0	124.444
(50, 5000)	0.966	311.111	0.933	0.933	1.0	311.111
(100, 5000)	1.0	333.333	1.0	1.0	1.0	333.333
(200, 5000)	0.966	311.111	0.933	0.933	1.0	311.111

Table 22: Performance for all combinations of missing data and noise levels for regression problems in Tier 6 (Case 42). Highlighted values denote default settings for the respective parameters.

Noise STD	Ratio of NaNs	F1 Score	Recall	Jaccard	Precision
0.1	**0.0**	0.963	0.929	0.929	1.0
0.2	**0.0**	0.889	0.857	0.8	0.923
0.5	**0.0**	0.8	0.857	0.667	0.75
1.0	**0.0**	0.703	0.929	0.542	0.565
0.1	0.1	0.643	0.643	0.474	0.643
0.1	0.2	0.75	0.643	0.6	0.9
0.1	0.5	0.75	0.643	0.6	0.9
0.2	0.1	0.5	0.5	0.333	0.5
0.5	0.1	0.167	0.214	0.091	0.136
0.2	0.2	0.581	0.643	0.409	0.529
0.5	0.2	0.171	0.214	0.094	0.143
0.2	0.5	0.483	0.5	0.318	0.467

Table 23: Average performance on classification and regression tasks when varying only noise levels. (Case 45) Bold font denotes the greater value of a pair.

Task	F1 Score	ASI	Recall	Jaccard	Precision	SI
Classification	0.835	10.113	0.875	0.726	0.802	12.5
Regression	**0.839**	**10.312**	**0.893**	**0.734**	**0.810**	**12.755**

Table 24: Average performance on classification and regression tasks when varying only missing data ratio. (Case 46) Bold font denotes the greater value of a pair.

Task	F1 Score	ASI	Recall	Jaccard	Precision	SI
Classification	0.725	7.786	0.679	0.584	0.787	9.694
Regression	**0.776**	**8.925**	**0.714**	**0.651**	**0.861**	**10.204**

Table 25: Average performance on classification and regression tasks when varying both noise levels and the ratio of missing data. (Case 47) Bold font denotes the greater value of a pair.

Task	F1 Score	ASI	Recall	Jaccard	Precision	SI
Classification	0.614	6.088	**0.637**	0.476	0.609	**9.099**
Regression	**0.617**	**6.359**	0.631	**0.488**	**0.621**	9.014

Appendix B: Plots for visual comparison of experimental results

Figure 10: F1, recall and precision in baseline experiments (Case 03).

Figure 11: Performance in high-dimensional scenarios (Case 10).

Figure 12: Performance in sample-rich scenarios (Case 13).

Figure 13: Overview of performance in basic experiments in tiers 1–3 for common sparsity = 5 (Case 16).

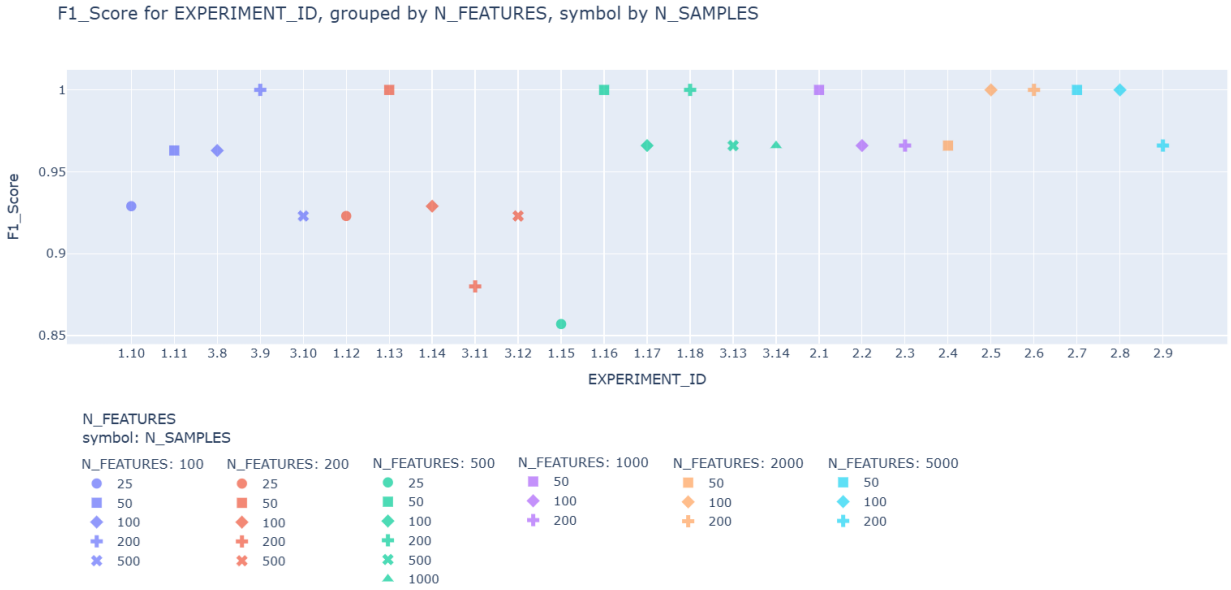


Figure 14: The effect of noise for varying amount of missing data. Problem $[n = 100, p = 200]$. (Case 19)

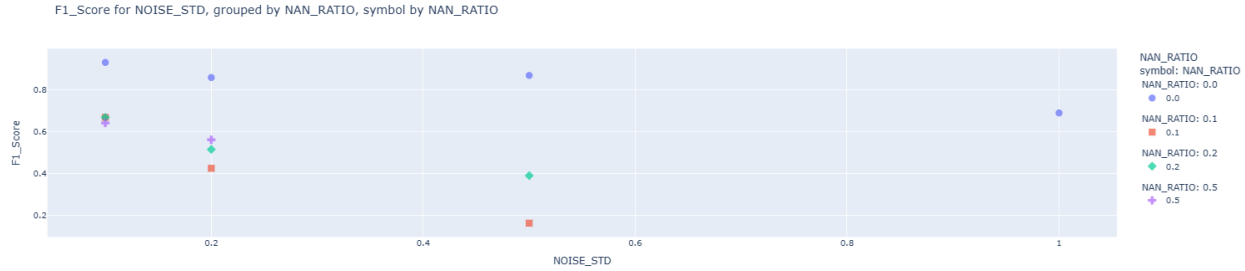


Figure 15: The effect of noise for varying amount of missing data. Problem $[n = 200, p = 500]$. (Case 22)

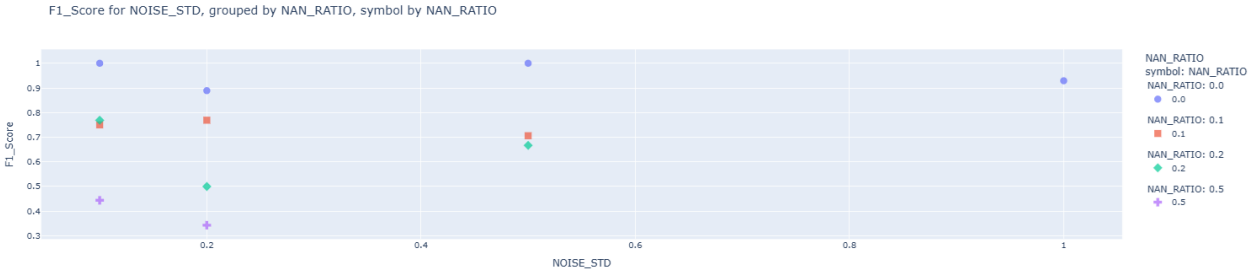


Figure 16: The effect of missing data for varying levels of noise. Problem $[n = 100, p = 200]$. (Case 19)

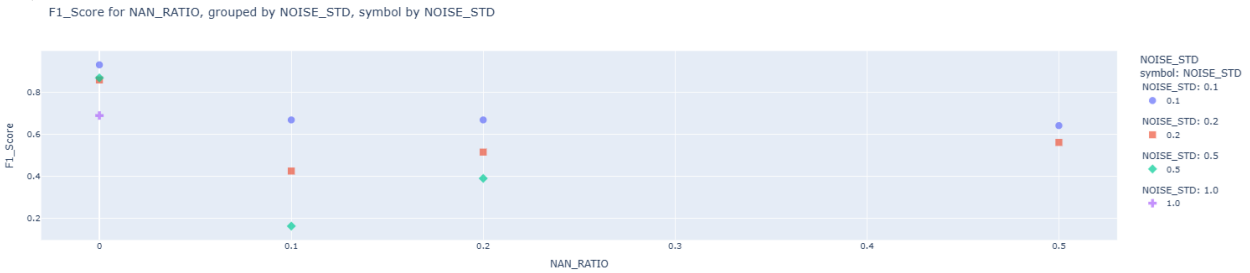


Figure 17: The effect of missing data for varying levels of noise. Problem $[n = 200, p = 500]$. (Case 22)

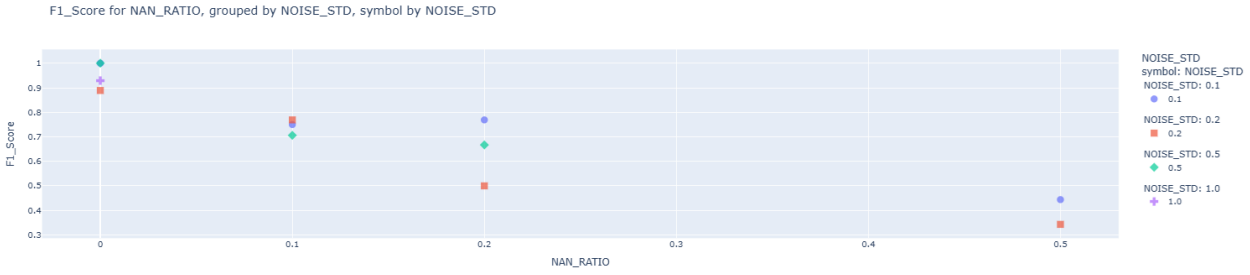


Figure 18: The effect of noise and missing data on performance (Case 23).

Figure 19: Overall effect of Jaccard penalty in both sparsity regimes (Case 28).

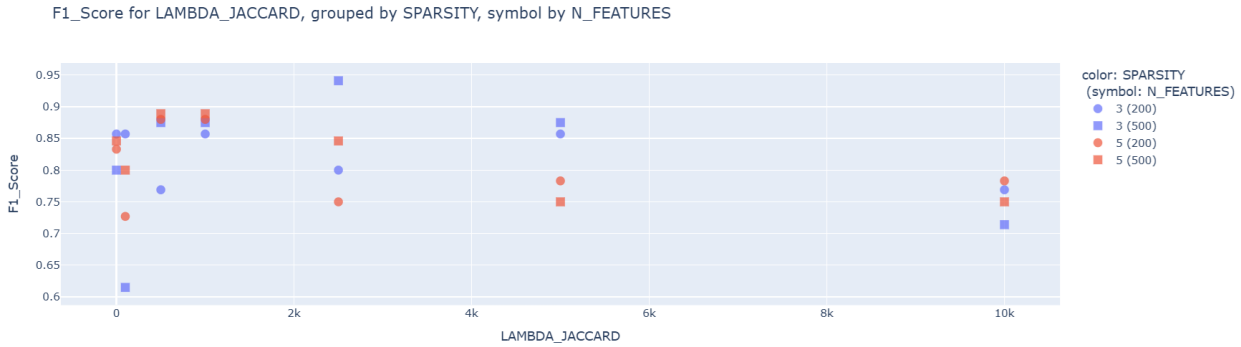


Figure 20: Baseline and high-dimensional scenarios with default noise level and no missing values. Comparing classification (red) vs. regression (blue). (Cases 43 and 44).

Figure 21: Effect of noise with no missing data (default). Comparing classification (red) vs. regression (blue). (Case 45)

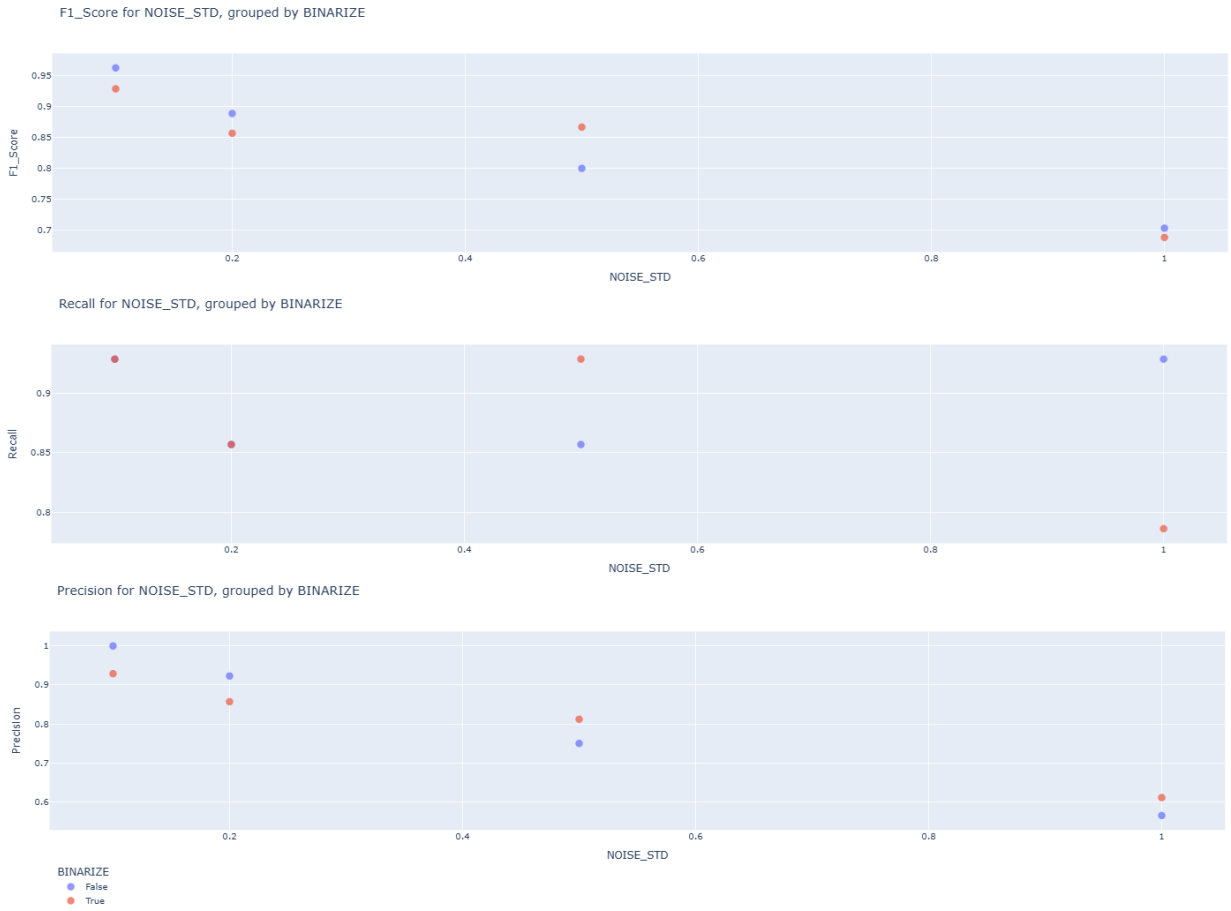


Figure 22: Effect of missing values at default noise level. Comparing classification (red) vs. regression (blue). (Case 46)

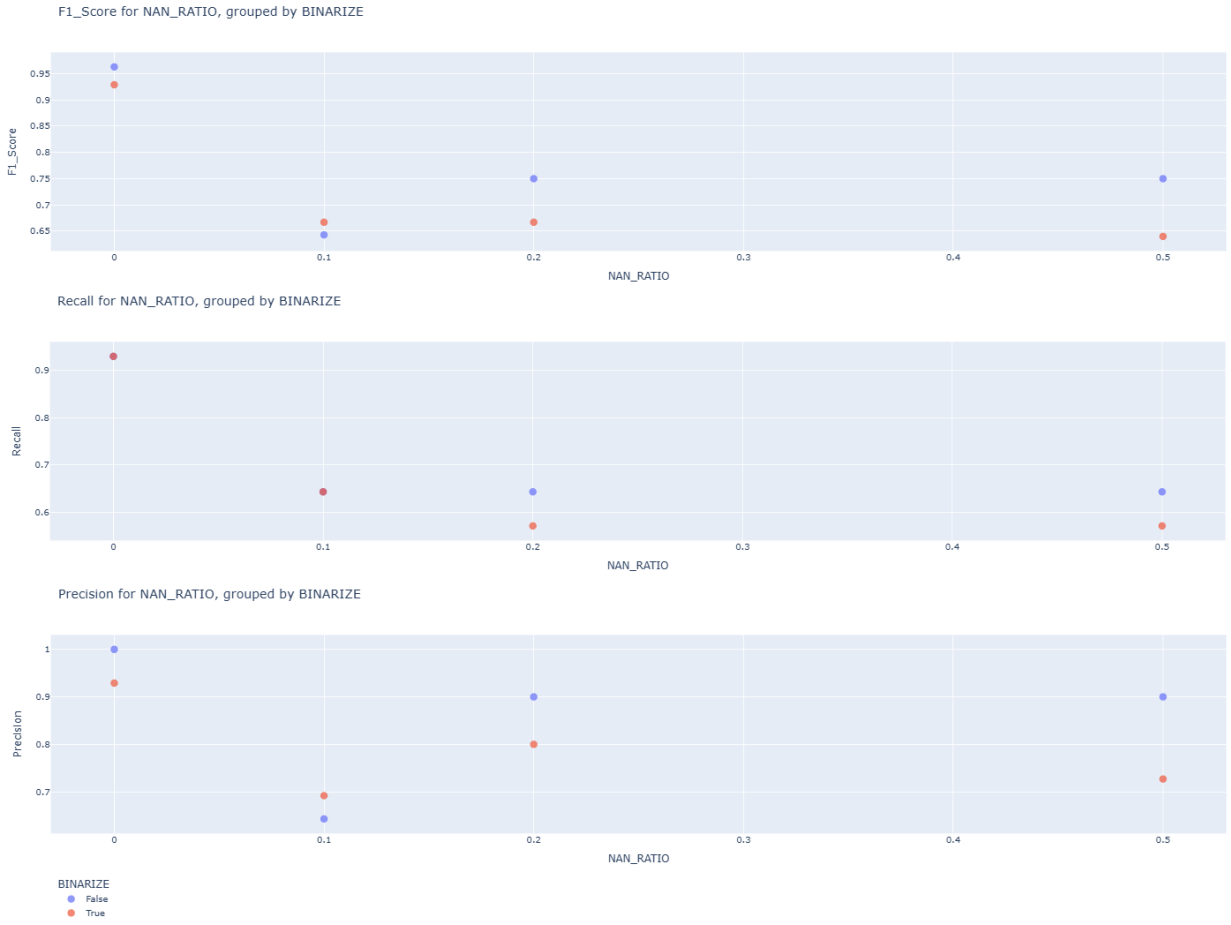
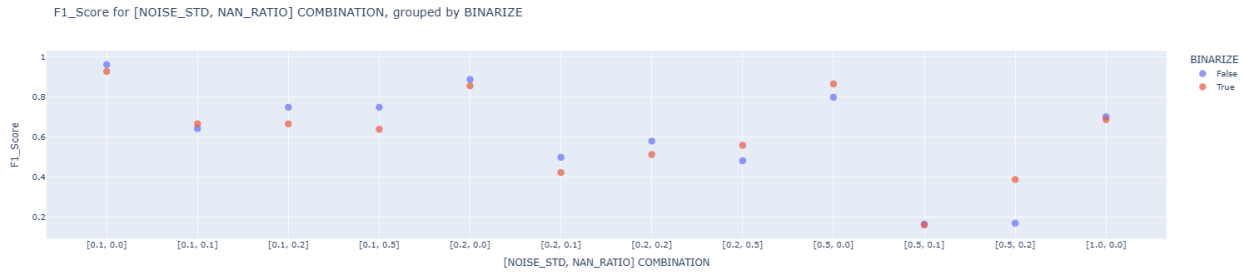


Figure 23: Performance comparison between classification (red) and regression (blue) in adverse scenarios (Case 47).



Appendix C: Implementation details

The GEMSS algorithm is implemented as an open-source Python package, designed for both experiment reproducibility and practical use on custom datasets. Below we highlight the key components; further details, usage instructions, and practical examples can be found in the project's `README.md`.

8.4 Code structure

The repository follows a modular organization:

- `gemss/feature_selection`: The core logic and essential building blocks of the feature selection algorithm.
- `gemss/postprocessing/`: Utilities for retrieving solutions after the feature selector has been run, their analysis, visualization and basic prediction modeling.
- `gemss/data_handling/`: Utilities to load, handle and preprocess data. Includes tools for generating artificial data used for demo and testing.
- `gemss/experiment_assessment/`: Modules dedicated to analyzing the results of experiments on artificial data.
- `gemss/diagnostics/`: Utilities for diagnostics of the algorithm's run that provide guidance for hyperparameter tuning (work in progress).
- `gemss/config/`: Central configuration of all parameters (of a single experiment) in JSON files for reproducible settings.
- `gemss/utils/`: Utility functions for printing and visualizations.
- `notebooks/`: Interactive demos for typical workflows, for both synthetic data and real user datasets. It guides the users through the usage of GEMSS as well as downstream modeling with TabPFN.
- `notebooks/analyze_experiment_results` Interactive notebooks to assess the results of experiments on artificial data.
- `scripts/`: Command-line scripts to run single or batch experiments on artificial data.

8.5 Running GEMSS

For most users, the recommended starting points are the interactive Jupyter notebooks included in the `notebooks/` directory:

- **Demo notebook (`demo.ipynb`):** This notebook demonstrates GEMSS on fully synthetic data explicitly constructed to contain multiple known sparse solutions. It walks the user through all stages: generating artificial data with predefined parameters (size, number of solutions, noise level, ratio of missing values), running the model, extracting and visualizing solutions, and validating results. This is the fastest and most reproducible way to understand the end-to-end workflow, verify installation, and explore algorithm behavior.

- **Exploratory notebook for arbitrary datasets (`explore_custom_dataset.ipynb`):** This notebook guides the user through loading and preprocessing their own dataset (in CSV), configuring algorithm settings, running GEMSS, and interpreting the discovered solutions. The features also include basic performance diagnostics with tuning recommendations and structured postprocessing for in-depth analysis. This notebook is intended for exploratory work on new, real-world datasets.

8.6 Requirements and dependencies

GEMSS is built on Python 3.11+, using PyTorch for backend computation and optimization. Standard libraries for data (Numpy, Pandas), plotting (Plotly), and machine learning (Scikit-learn) are used. Interactive widgets are created using IPyWidgets. Optional advanced postprocessing uses TabPFN and Shap.

Appendix D: Recommendations for practitioners

Based on the extensive experimental validation, we offer the following guidelines for applying GEMSS to new datasets and observed heuristics.

8.7 Monitoring convergence

Visual inspection of the optimization history is highly recommended. Reliable solutions (= mixture components) typically exhibit dynamic feature selection behavior, where importances vary significantly during the early phase before settling on a sparse set of non-negligible coefficients.

ELBO Magnitude. While the absolute value of the Evidence Lower Bound (ELBO) depends on the dataset size, final values with magnitudes in the low thousands are generally acceptable (assuming a Jaccard penalty $\lambda_J \approx 500$). Conversely, magnitudes in the millions often indicate divergence or scaling issues.

8.8 Hyperparameter tuning

The following parameters have the most significant impact on algorithm performance:

- `VAR_SPIKE` (for SSS prior): This is the primary mechanism for controlling sparsity.
  - If all feature coefficients (μ) converge to zero, the spike is likely too narrow (variance too small).
  - If too many features are selected (dense solutions), the spike may be too broad (depending on the properties of the specific dataset).
- `N_CANDIDATE_SOLUTIONS` (m): We strongly recommend setting the number of mixture components to $2\times$ or $3\times$ the expected number of distinct underlying mechanisms. Experiments confirm that "superfluous" components are pruned or converge to duplicate strong solutions. This redundancy is a much safer failure mode than underspecifying m, which risks missing valid hypotheses entirely. Note that as m decreases, higher `LAMBDA_JACCARD` values may prevent mode collapse.
- **Batch Size & Missing Data:** GEMSS handles missing values natively by masking the log-likelihood. This approach is robust for missingness ratios up to $\approx 10\%$. However, Tier 4 results indicate non-linear degradation beyond 20% missingness due to increased gradient variance. In high-missingness regimes, we recommend increasing `BATCH_SIZE` (e.g., from 32 to 64 or higher, depending on sample size) to stabilize gradient estimates, or performing a coarse prior imputation.

8.9 Managing solution diversity

While not strictly mandatory, the Jaccard-based regularization is effective for forcing the discovery of orthogonal support sets, a property that may or may not be desirable depending on the specific problem.

- **Optimal penalty range:** Our analysis (Tier 5) and further empirical observations on normalized datasets indicate that a penalty in the range $\lambda_J \in [500, 1000]$ strikes an optimal balance. Optimal values may differ depending on the general magnitude of the data.

- **High penalty risks:** Excessive penalties can destabilize optimization by creating steep barriers between modes.

8.10 Solution recovery strategies

Selecting the right strategy to extract discrete feature sets from the continuous variational posterior is context-dependent. We recommend comparing the following approaches:

- **"Top-D" selection:** General approach, may be too scrict on recall
- **Outlier detection:** More flexible regarding number of features in a selection. Recommended when the desired sparsity is unknown. STD threshold can be chosen ex-post.
- **Cross-component validation:** Important features often rank highly across multiple mixture components, even if they do not make the final cut in every specific solution. A feature present near the top of only one minor component is more likely to be noise. This is why the "full" solutions are useful.

8.11 Solution recovery strategies

Selecting the right strategy to extract discrete feature sets from the continuous variational posterior is context-dependent. We recommend comparing the following approaches:

- **"Top-D" selection:** This is the standard approach for enforcing strict sparsity. While generally effective for high-dimensional precision, it can be overly strict (especially) on recall if the underlying mechanism involves more features than the pre-defined D, potentially cutting off valid predictors.
- **Outlier detection:** This approach is more flexible regarding the solution size and is highly recommended when the true sparsity level is unknown. A key advantage is that the sensitivity threshold (e.g., number of standard deviations) can be chosen and adjusted.
- **Cross-component validation:** Inspecting the "full" solutions (thresholded only by the minimal μ values) is valuable for validation. Important features often rank highly across multiple mixture components, even if they do not make the final cut in every specific sparse selection. Conversely, a feature present near the top of only one component is more likely to be noise.