Pattern Recognition 172 (2026) 112727

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Variable priority for unsupervised variable selection

Lili Zhou, Min Lu, Hemant Ishwaran *

Division of Biostatistics, Miller School of Medicine, University of Miami, 33136, Miami, FL, USA

ARTICLE INFO

Keywords:
Autoencoder forest
Release region
s-dependent variable
Signal variable

ABSTRACT

In unsupervised settings where labeled data is unavailable, identifying informative features is both challenging and essential. Although numerous methods for unsupervised feature selection have been proposed, significant opportunities for improvement remain. This paper introduces a new method that extends the supervised Variable Priority (VarPro) framework to the unsupervised domain. The central idea is to recast feature selection as a collection of localized two-class classification problems, where class labels are defined by membership in regions derived using decision tree rules and their corresponding releases. This transformation introduces a form of implicit supervision without requiring outcome labels and is combined with lasso-based regression to encourage sparsity and mitigate noise in high-dimensional settings. Extensive experiments on synthetic benchmarks demonstrate consistent improvements over existing methods across a range of latent, correlated, and clustered scenarios. Real-world validation in biological and image data further confirms the method's effectiveness, including recovery of known cancer-associated genes and improved clustering in lung cancer subtyping.

1. Introduction

Feature selection is a cornerstone of data preprocessing in machine learning, especially in high-dimensional domains such as bioinformatics, text mining, and image analysis. Removing irrelevant or noisy variables can improve model accuracy, reduce computation, and enhance interpretability. The challenge becomes especially difficult in unsupervised settings, where no outcome labels exist to guide the search. Unlike supervised methods that assess importance by association with a response, unsupervised approaches must rely entirely on relationships hidden within the features themselves. Yet the need is just as pressing, since effective unsupervised feature selection can improve clustering, visualization, and change detection while reducing overfitting.

This raises a basic question: what does it mean for a variable to be informative without labels? Simple heuristics such as variance ranking or cluster separation often overlook complex dependencies. A more principled answer arises from statistical dependence, where a variable is informative if it helps explain the joint distribution of the data. Under this view, signal variables are those that capture dependence among features, while noise variables can be removed without altering the distribution. This parallels the Markov blanket in graphical models [1], in which a minimal subset of variables renders all others conditionally independent.

Guided by this perspective, we introduce *Unsupervised Variable Priority* (UVarPro), an extension of the VarPro framework [2,3] to the unsupervised setting. UVarPro operationalizes this dependence view by testing whether variables contribute to local conditional relationships in the feature space. At the same time, it bridges supervised and unsupervised paradigms: supervised methods identify variables predictive of an outcome, while unsupervised methods reveal dependencies among the features themselves. UVarPro combines these ideas through a series of localized classification tasks that provide implicit supervision derived directly from the structure of the data.

To implement this framework, UVarPro generates the localized tasks by partitioning the feature space into simple, rule-defined regions using an unsupervised forest composed of many decision trees. Each region R (a terminal node from a tree) corresponds to a set of conditions that define a compact portion of the feature space. To assess the role of a variable s, UVarPro relaxes the condition on that variable, creating a slightly expanded *release region*. The portion extending beyond R forms the *near-miss set*, containing observations that almost, but not quite, satisfy the same rule. The resulting two-class problem asks whether a sample lies inside R or in its near-miss set. UVarPro trains a localized classifier on this contrast, measuring how strongly each variable helps separate the two groups. Variables that consistently improve classification performance across many such local tasks are identified as driving key dependencies in the data. Although UVarPro uses tree rules, these are derived entirely from the unsupervised forest structure and serve only to define geometric partitions of the feature space, not to predict a target label. Fig. 1 illustrates the framework and provides simple geometric

* Corresponding author.
E-mail addresses: lxz516@miami.edu (L. Zhou), m.lu6@umiami.edu (M. Lu), hishwaran@miami.edu (H. Ishwaran).
URL: https://ishwaran.org (H. Ishwaran)

https://doi.org/10.1016/j.patcog.2025.112727
Received 22 May 2025; Received in revised form 6 November 2025; Accepted 7 November 2025
Available online 15 November 2025

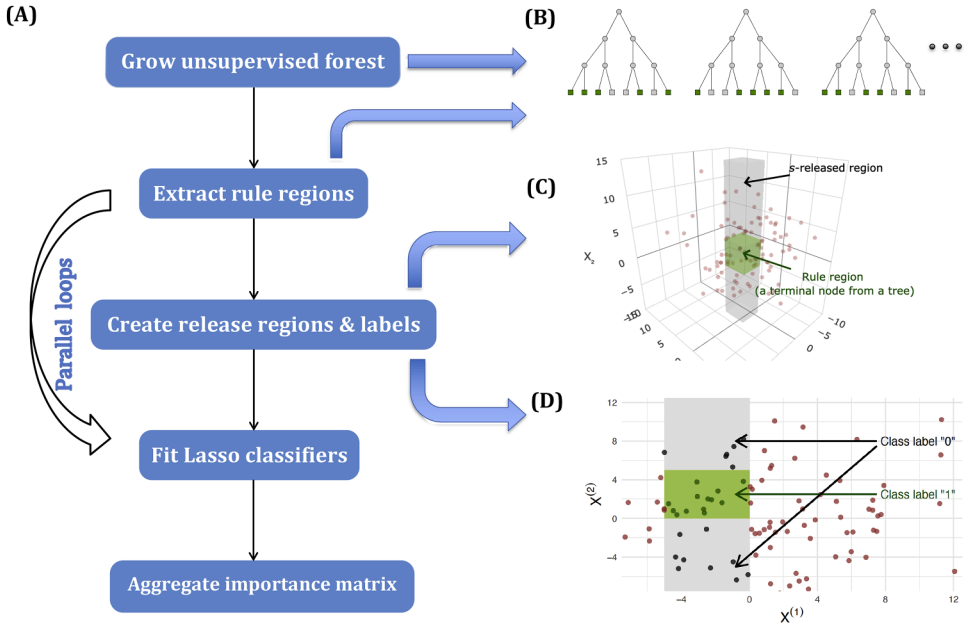

Fig. 1. Overview of the UVarPro framework. (A) Unsupervised forest pipeline: the feature space is partitioned into rule-defined regions, release regions and pseudo-labels are generated for each variable, local lasso classifiers are fit, and variable importances are aggregated into a global ranking. (B) Example of unsupervised trees defining geometric partitions without outcome guidance, where rule regions (green) are obtained by randomly sampling terminal nodes from the trees. (C) Three-dimensional illustration of a rule region (green cube) and its s-released extension (gray slab, including the green cube), showing how relaxing a variable expands the region (here $s = 2$). (D) Two-dimensional schematic of the same concept, classifying samples inside the rule region (dark green, label "1") versus the near-miss set (gray, label "0"). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

examples showing how relaxing a variable expands a rule-defined region to form the release and near-miss sets used for local classification. Below we summarize the key contributions of our work.

1. We propose a formal criterion for defining *signal variables* in the unsupervised setting, framed in terms of conditional independence. This criterion separates informative features that participate in the joint dependence structure from noise features that do not.
2. We introduce *UVarPro*, a novel task-driven framework that transforms unsupervised feature selection into a collection of localized classification problems, creating implicit supervision from the geometry of the feature space itself.
3. We incorporate ℓ_1-regularized logistic models within each local task, which shrink irrelevant variables to zero and retain only those with predictive power. This embedded regularization enforces sparsity, suppresses noise, and stabilizes importance assessment, leading to robust and scalable feature rankings.
4. We develop a theoretical foundation showing that UVarPro recovers the true signal set under a property we term *s-dependence*, where a variable is s-dependent if it appears in at least one conditional dependency within the joint distribution of the features. This formalism connects the algorithmic design of UVarPro to the underlying conditional independence factorization.

1.1. Outline

The paper is organized as follows. Section 2 reviews related work, grouping existing unsupervised feature selection approaches by methodological theme and discussing their strengths and limitations in relation to our framework. Section 3 introduces a formal definition of informativeness in the absence of labels, using conditional independence to derive a factorization that separates signal from noise variables. We also discuss why the signal set need not be unique and explain how the formulation remains valid under natural generalizations. Section 4 develops the theoretical foundation of UVarPro and details the algorithmic construction, including the integration of lasso regularization within each local task. Section 5 presents synthetic simulations illustrating how the method distinguishes signal from noise in controlled settings, while Section 6 benchmarks UVarPro against competing approaches on real-world datasets from multiple application domains. Section 7 concludes with a summary of contributions and directions for future work.

2. Related work

Unsupervised feature selection has been a longstanding topic in machine learning, with methods developed from diverse theoretical perspectives. Existing approaches can be broadly grouped into: (1) variance based and redundancy based criteria [4]; (2) graph, clustering, and manifold preserving formulations (e.g., structure preserving or graph aware filters) [5,6]; (3) dependence and relationship based analysis [7]; (4) reconstruction and joint learning frameworks that couple selection with latent structure discovery (including multiview settings) [8,9]; and (5) unsupervised learning pipelines that embed feature selection within broader workflows such as clustering, segmentation, and change detection. Below we review these categories, emphasizing their methodological principles and limitations in relation to our proposed method.

2.1. Variance and redundancy based methods

Early methods focus on maximizing variance or minimizing redundancy among selected variables. Forward Selection Component Analysis (FSCA) [10] sequentially adds variables that explain the greatest portion of total variance, analogous to a greedy PCA. Its extension, Lazy-FSCA (L-FSCA) [11] adopts a lazy greedy evaluation [12] to substantially reduce computations while delivering selection performance that is empirically comparable to FSCA; the frame potential approach (FSFP) [13] greedily minimizes the frame potential (the squared Frobenius norm of the Gram matrix of the selected columns), thereby indirectly discouraging redundancy by selecting the feature that causes the smallest increase in total squared pairwise correlations. These criteria are computationally efficient but depend solely on second order structure, making them

less effective in capturing nonlinear or multimodal dependencies. Recent work also advances redundancy aware selection via maximum relevance and minimum global redundancy selection, improving discrimination while explicitly penalizing global redundancy [4].

2.2. Clustering and manifold preserving methods

A major branch of research connects feature relevance to clustering or geometric structure preservation. Laplacian Score [14] ranks features by their ability to preserve local neighborhood structure, while SPEC [15] generalizes spectral criteria to integrate local and global geometry. More sophisticated graph based approaches follow two related patterns. MCFS [16] selects features via a spectral embedding followed by sparse regression, whereas UDFS [17] and NDFS [18] jointly optimize feature weights and latent cluster indicators with sparsity and orthogonality constraints to better align features with cluster structure. Recent work continues this trend: higher order, view specific similarity graphs for multiview selection (DCVSG) further strengthen geometric fidelity [19], while dual space fuzzy graphs with orthogonal basis clustering enhance robustness to ambiguous neighborhoods [5]. In parallel, structure preserving filters learn feature weights to retain neighborhood relations in the feature subspace [6]. Although powerful for revealing latent clusters, these methods often rely on assumptions of smooth manifolds or prespecified neighborhood scales, which may limit adaptability in heterogeneous data.

2.3. Dependence and causal approaches

Dependence based methods identify relevant variables through measures of association or conditional independence. The PC-simple algorithm [20] performs sequential tests of conditional independence between each feature and the response, removing variables that become independent given small conditioning sets; it is inspired by the PC algorithm but targets supervised variable selection rather than full causal skeleton recovery. Measures such as the Maximal Information Coefficient (MIC) [21] and HHG [22] provide flexible, nonparametric dependence detection that can uncover nonlinear relationships. While conceptually appealing, these procedures are sensitive to estimation noise and scale poorly with dimensionality, motivating approaches that integrate dependence with structure learning. Complementing classical dependence measures, robust relationship learning formulations have been shown to improve selection in the presence of outliers and heterogeneous local structure [7].

2.4. Reconstruction and joint learning frameworks

A complementary line of work embeds feature selection within reconstruction or subspace learning objectives. JAMEL [23] preserves manifold geometry while learning an embedding for feature selection, whereas nonnegative Laplacian embedding [24] exploits pseudo-labels and regularized objectives for more discriminative subsets. Similarly, SCUFS [25] integrates representation based subspace clustering and iterative refinement to retain features aligned with latent subspaces. These approaches offer strong geometric interpretability but require matrix factorization or spectral decomposition steps that scale quadratically or cubically with data size. Multiview joint formulations that marry low rank reconstruction with graph learning also report gains on high dimensional benchmarks, e.g., tensor RPCA with consensus graph learning [9]. Joint formulations that learn cluster structure and feature weights concurrently have recently been explored, yielding gains on high dimensional benchmarks [8].

2.5. Unsupervised learning paradigms

Another line of work, prominent in remote sensing and change detection, situates feature selection considerations within broader pipelines that include unsupervised or weakly supervised components (e.g., pixel clustering, object based segmentation, and multitemporal differencing). Recent studies combine handcrafted spectral and texture descriptors with classical classifiers to evaluate performance across sensors, spatial resolutions, and temporal settings [26–29]. These pipelines underscore the importance of segmentation scale and spatial context for accurate mapping, but explicit unsupervised feature selection is rarely incorporated; most approaches retain all descriptors or rely on ad hoc pruning. From a feature selection standpoint, the resulting feature spaces are high dimensional and strongly redundant, motivating the integration of principled selection criteria. Embedding geometry preserving or dependence based filters within such workflows could reduce noise, enhance interpretability, and improve transferability across sensors and time.

2.6. Summary and positioning of UVarPro

The methods described above emphasize either global structure (variance and reconstruction approaches) or local neighborhood preservation (graph and manifold methods), with relatively few methods linking variable selection directly to local predictive capacity. Our method, UVarPro, advances this direction by reframing unsupervised feature selection as a series of localized prediction tasks derived from decision tree rules. This task driven formulation identifies variables that drive local dependence, allowing UVarPro to adapt to complex, heterogeneous structures beyond global variance or smooth manifolds.

3. Notation and definition of the signal feature set

In this section we formally define what it means for a variable to be informative without access to an observed outcome. Let $X^{(1)}, \dots, X^{(p)}$ denote the p observed features, and write $\mathbf{X} = (X^{(1)}, \dots, X^{(p)})$ for the feature vector. For any subset $S \subseteq \{1, \dots, p\}$, let $\mathbf{X}^{(S)} = \{X^{(j)}\}_{j \in S}$ be the corresponding subvector, and $\mathbf{X}^{\setminus(S)} = \{X^{(j)}\}_{j \notin S}$ its complement.

Our goal is to identify a subset $\mathscr{S} \subseteq \{1, \dots, p\}$, called the *signal set*, such that conditioning on $\mathbf{X}^{(\mathscr{S})}$ renders the remaining variables mutually independent. In this sense, $\mathbf{X}^{(\mathscr{S})}$ serves as a minimal sufficient set for the dependence structure in $\mathbf{X}$. The definition is given below.

Definition 1. A subset $\mathscr{S} \subseteq \{1, \dots, p\}$ is a set of signal variables if it is a minimal set of coordinates such that all other variables are conditionally independent given $\mathbf{X}^{(\mathscr{S})}$. Formally, $\mathscr{S}$ is any minimal set satisfying:

$$X^{(j)} \perp X^{(k)} \mid \mathbf{X}^{(\mathscr{S})}, \forall \{j, k\} \nsubseteq \mathscr{S}.$$

This condition implies a key factorization of the joint feature density:

$$\begin{aligned} f_{\mathbf{X}}(\mathbf{x}) &= f_{\mathbf{X}^{\setminus(\mathscr{S})} \mid \mathbf{X}^{(\mathscr{S})}}(\mathbf{x}^{\setminus(\mathscr{S})})\, f_{\mathbf{X}^{(\mathscr{S})}}(\mathbf{x}^{(\mathscr{S})}) \\ &= \left(\prod_{j \notin \mathscr{S}} f_{X^{(j)}}(x^{(j)} \mid \mathbf{X}^{(\mathscr{S})}) \right) f_{\mathbf{X}^{(\mathscr{S})}}(\mathbf{x}^{(\mathscr{S})}). \end{aligned} \tag{1}$$

For convenience, we refer to the complement of the signal set, $\mathscr{N} := \{1, \dots, p\} \setminus \mathscr{S}$, as the set of *noisy variables*. These variables contribute no additional dependence beyond what is explained by the signal set.

As illustration, consider a latent variable model with a single unobserved variable Z and three observed features $X^{(1)}, X^{(2)}, X^{(3)}$. Suppose the joint distribution satisfies

$$X^{(j)} \perp X^{(k)} \mid Z, \quad \text{for all } j \neq k \in \{1, 2, 3\}.$$

That is, Z induces all dependence among the observed features (see Fig. 2). Now assume $X^{(1)}$ is a deterministic function of Z, while $X^{(2)}$ and $X^{(3)}$ are noisy but conditionally independent given Z. In this case, $X^{(1)}$ alone suffices to account for the dependence between $X^{(2)}$ and $X^{(3)}$, meaning that $\mathscr{S} = \{1\}$ satisfies the conditional independence criterion in Definition 1.

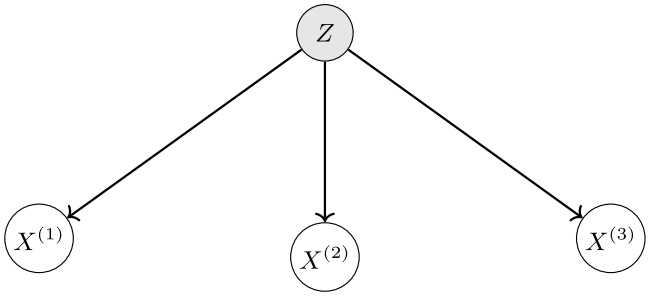

Fig. 2. Latent variable model with Z inducing dependence among observed features. The features $X^{(1)}, X^{(2)}, X^{(3)}$ are conditionally independent given Z.

3.1. Non-uniqueness and structure of the signal set

The density factorization (1) is central to our unsupervised extension of VarPro because it separates signal from noise variables. While Definition 1 characterizes signal sets as minimal sufficient sets, such sets need not be unique. In practice, non-uniqueness arises because some variables play equivalent statistical roles, while others may be deterministically related [1]. In these cases the signal set must be expanded to preserve valid dependency structure. Our theoretical justification remains intact so long as (1) holds for this expanded set. We next consider two representative scenarios, *interchangeability* and *redundancy*, and illustrate how they are accommodated within this framework.

3.1.1. Interchangeability

An example of non-uniqueness in $\mathscr{S}$ arises through the concept of *interchangeability*. Consider a pair of variables $\hat{\theta} = \{i, j\}$ such that

$$\forall k \notin \mathscr{S} \cup \hat{\theta}, \text{ we have } X^{(k)} \perp X^{(i)} \mid \{\mathbf{X}^{(\mathscr{S}\setminus\hat{\theta})}, X^{(j)}\} \text{ and}$$
$$X^{(k)} \perp X^{(j)} \mid \{\mathbf{X}^{(\mathscr{S}\setminus\hat{\theta})}, X^{(i)}\}. \tag{2}$$

This condition implies that either $X^{(i)}$ or $X^{(j)}$ may be included in $\mathscr{S}$ without affecting the conditional independence structure of the remaining variables. That is, $X^{(i)}$ and $X^{(j)}$ are *interchangeable*, and $\mathscr{S}$ is not uniquely defined. To preserve (1), we resolve this ambiguity by expanding $\mathscr{S}$ to include both variables: $\mathscr{S} := \mathscr{S} \cup \{i, j\}$.

3.1.2. Redundancy

A second source of non-uniqueness arises when a signal variable is *redundant*, meaning it is a deterministic function of other variables already in $\mathscr{S}$. Specifically, if $\mathscr{S}$ is a valid signal set, we say variable j is a *redundant signal variable* if

$$X^{(j)} = \phi(\mathbf{X}^{(\mathscr{S})}) \quad \text{for some real-valued function } \phi. \tag{3}$$

In such cases, $\mathscr{S}$ should again be augmented to ensure (1) holds: $\mathscr{S} := \mathscr{S} \cup \{j\}$.

3.2. Examples of non-uniqueness and restructuring the signal set

Example 1. This *V-cluster* example illustrates a case where two variables are interchangeable. Let $p > 2$ and define:

$$X^{(1)} = \epsilon_1, \qquad X^{(2)} = \delta X^{(1)} + \epsilon_2, \qquad X^{(3)} = \epsilon_3, \ldots, X^{(p)} = \epsilon_p,$$

where $\epsilon_1, \ldots, \epsilon_p$ are independent noise variables and $\delta \in \{-1, +1\}$ is independent of ϵ_1. The variable $X^{(2)}$ reflects a sign-flipped copy of $X^{(1)}$, producing a V-shaped structure in $(X^{(1)}, X^{(2)})$.

Only $X^{(1)}$ and $X^{(2)}$ influence dependence structure, as the remaining variables are pure noise. Since $X^{(1)}$ and $X^{(2)}$ can be conditioned on to explain each other, they satisfy the interchangeability condition (2). Hence, the minimal signal set is $\mathscr{S} = \{1, 2\}$.

Example 2. This modified *V-cluster* example shows how introducing independent sources of latency can break interchangeability. Let $\delta_2, \delta_3 \in \{-1, +1\}$ be independent binary variables, and define:

$$X^{(1)} = \epsilon_1, \quad X^{(2)} = \delta_2 X^{(1)} + \epsilon_2, \quad X^{(3)} = \delta_3 X^{(1)} + \epsilon_3,$$
$$X^{(4)} = \epsilon_4, \ldots, X^{(p)} = \epsilon_p,$$

where $\epsilon_1, \ldots, \epsilon_p$ are independent noise variables. Here, $X^{(1)}$ is the shared source of dependence, making it a signal variable, while the rest are noisy.

Despite depending on $X^{(1)}$, neither $X^{(2)}$ nor $X^{(3)}$ is interchangeable with it. This is because their latent signs δ_2 and δ_3 are independent. Conditioning on $X^{(2)}$ does not fully determine $X^{(3)}$'s dependence on $X^{(1)}$, and vice versa:

$$X^{(3)} \not\perp X^{(1)} \mid X^{(2)}, \qquad X^{(2)} \not\perp X^{(1)} \mid X^{(3)}.$$

Thus, $X^{(1)}$ cannot be substituted by either $X^{(2)}$ or $X^{(3)}$, and the minimal signal set is $\mathscr{S} = \{1\}$.

Example 3. This example illustrates redundancy. Let $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4$ be independent, continuous random variables, and define:

$$X^{(1)} = \epsilon_1, \quad X^{(2)} = \epsilon_2, \quad X^{(3)} = X^{(2)} + \epsilon_3, \quad X^{(4)} = X^{(2)} + \epsilon_4,$$
$$X^{(5)} = (X^{(1)})^2.$$

Variables $X^{(1)}$ and $X^{(2)}$ are signals, as they independently influence other features. In contrast, $X^{(3)}$ and $X^{(4)}$ are conditionally independent of all others given $X^{(2)}$ and are therefore excluded from the minimal signal set. Variable $X^{(5)}$, however, is a deterministic function of $X^{(1)}$. By the redundancy criterion (3), such functions must be included to preserve the joint factorization (1). Thus, $\mathscr{S} = \{1, 2, 5\}$.

Example 4. This example shows that a variable may appear irrelevant but is required in the signal set to preserve conditional independence. Let $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4$ be continuous, mutually independent variables, and define:

$$X^{(1)} = \epsilon_1, \quad X^{(2)} = X^{(1)} + \epsilon_2, \quad X^{(3)} = X^{(1)} + \epsilon_3,$$
$$X^{(4)} = X^{(1)} + X^{(2)} + X^{(3)} + \epsilon_4.$$

One might expect $\mathscr{S} = \{1\}$ to suffice, since $X^{(2)}$ and $X^{(3)}$ are conditionally independent given $X^{(1)}$. However, $X^{(4)}$ includes contributions from ϵ_2 and ϵ_3, which also affect $X^{(2)}$ and $X^{(3)}$. As a result, conditioning only on $X^{(1)}$ does not render $X^{(2)}$ and $X^{(3)}$ independent of $X^{(4)}$:

$$X^{(2)} \not\perp X^{(4)} \mid X^{(1)}, \qquad X^{(3)} \not\perp X^{(4)} \mid X^{(1)}.$$

To restore conditional independence among noisy variables, $X^{(4)}$ must be included in the signal set. Thus, $\mathscr{S} = \{1, 4\}$.

Example 5. This example illustrates a variation of the redundancy structure from Example 3, involving a deeper causal chain. Let $\epsilon_1, \epsilon_2, \epsilon_3$ be continuous, mutually independent random variables, and define:

$$X^{(1)} = \epsilon_1, \quad X^{(2)} = (X^{(1)})^2, \quad X^{(3)} = X^{(1)} + X^{(2)} + \epsilon_2, \quad X^{(4)} = X^{(3)} + \epsilon_3.$$

Here, $X^{(2)}$ is a deterministic function of $X^{(1)}$ and is redundant under (3). Variables $X^{(3)}$ and $X^{(4)}$ are causally downstream. While conditional independencies like $X^{(1)} \perp X^{(4)} \mid \{X^{(2)}, X^{(3)}\}$ and $X^{(2)} \perp X^{(4)} \mid \{X^{(1)}, X^{(3)}\}$ hold, no proper subset of $\{1, 2, 3\}$ satisfies Definition 1. Thus, $\mathscr{S} = \{1, 2, 3\}$.

4. Methodology

The goal of unsupervised variable selection is to identify the most informative features without access to an outcome or class label. In supervised learning, importance is measured by the change in predictive accuracy when variables are perturbed or excluded. Without labels, importance must instead be defined through the intrinsic structure of the feature space. Our approach extends the Variable Priority (VarPro) framework to this setting. We begin by providing background on VarPro before describing the unsupervised extension.

4.1. Background on VarPro

VarPro [2,3] is a model-independent method for supervised feature selection. It is rule-based, using decision trees to generate candidate regions of the feature space, and evaluates variable importance by contrasting predictions across such regions. A rule ζ corresponds to a root-to-leaf path in a decision tree, defined by a sequence of variable-based splits. The associated region for ζ is

$$R(\zeta) = \bigcap_{j \in A(\zeta)} \{\mathbf{x} : x^{(j)} \in \mathcal{I}_j(\zeta)\},$$

where $A(\zeta)$ is the set of variables along the path and $\mathcal{I}_j(\zeta)$ the interval associated with coordinate j.

To evaluate the contribution of a set $S \subset \{1, \ldots, p\}$, VarPro defines the released rule ζ^S by removing all split constraints involving S:

$$R(\zeta^S) = \bigcap_{j \in A(\zeta) \setminus S} \{\mathbf{x} : x^{(j)} \in \mathcal{I}_j(\zeta)\}.$$

Importance is then measured by comparing outcome summaries on $R(\zeta)$ and $R(\zeta^S)$ using observed data. This construction generalizes the mean decrease impurity (MDI) score, which quantifies marginal improvement in split quality when a variable is introduced. VarPro extends this idea from single splits to entire regions, capturing the joint contribution of variables within a neighborhood. In doing so, it provides greater stability under correlation and interaction, while avoiding the synthetic inputs required by permutation-based methods [30,31].

In practice, a large and diverse collection of rules is obtained by growing a forest of trees and sampling rules across its terminal nodes. This makes the method computationally efficient: once a forest has been grown, $R(\zeta^S)$ is obtained by simply relaxing the splits involving S, and membership in $R(\zeta)$ or $R(\zeta^S)$ can be checked by traversing the trees without retraining. The same collection of rules can be reused to estimate a variety of outcome summaries, including conditional means, class probabilities, or restricted mean survival time. This efficiency and flexibility carry over directly to the unsupervised extension described next.

4.2. Extension to unsupervised data

When no response Y is available, the supervised contrast between $R(\zeta)$ and $R(\zeta^S)$ used in VarPro cannot be formed. We replace it with a local classification problem, where labels are assigned by rule membership. This creates implicit supervision for assessing the importance of S.

Two modifications are required to implement this idea. The first is to adapt the rule-construction process by using multivariate trees that treat the feature vector $\mathbf{X}$ as both input and pseudo-response. This self-regression is conceptually similar to an autoencoder and leverages multivariate impurity splitting strategies from [32]. Each tree produces terminal nodes that define candidate regions in the feature space. By growing a forest of such trees, we obtain many rules ζ, each corresponding to the sequence of splits along the path from the root to a sampled terminal node.

The second modification is to create surrogate labels through a non-intersecting near-miss set. For a rule ζ and coordinate $s \in A(\zeta)$, removing all constraints involving $X^{(s)}$ defines the release region

$$R(\zeta^s) = \bigcap_{j \in A(\zeta) \setminus \{s\}} \{\mathbf{x} : x^{(j)} \in \mathcal{I}_j(\zeta)\},$$

and the near-miss set is $\mathrm{rel}_s(\zeta) = R(\zeta^s) \setminus R(\zeta)$. The binary label is

$$Y_{s,\zeta}(\mathbf{x}) = \begin{cases} 0, & \mathbf{x} \in R(\zeta), \\ 1, & \mathbf{x} \in \mathrm{rel}_s(\zeta). \end{cases}$$

These labels depend only on $\mathbf{X}$ and do not require observed outcomes.

To evaluate the contribution of $X^{(s)}$, we predict $Y_{s,\zeta}$ from $\mathbf{X}^{\setminus(s)}$ using samples from $R(\zeta)$ and its near-miss set. This isolates the effect of relaxing the constraint on coordinate s while keeping all other rule conditions fixed. If $\mathbf{X}^{\setminus(s)}$ can reliably predict the contrast, then releasing $X^{(s)}$ changes the dependence structure, indicating that $X^{(s)}$ participates in the joint signal. Otherwise, $X^{(s)}$ is likely uninformative in that neighborhood. This construction is repeated across many rules drawn from the forest. For notational clarity, we index the sampled rules as $\zeta_1, \ldots, \zeta_B$, where B is the total number of rules across the forest. Outputs from these rules are later aggregated to form variable-level importance scores.

4.3. Justification for noisy features

To justify the validity of our approach, we first consider the case where $s \in \mathcal{N}$ is a noisy variable; the corresponding case of a signal variable is examined afterwards. Omitting dependence on the rule ζ, it follows from the joint factorization assumption (1) that

$$\mathbb{P}\{Y_s = 1 \mid \mathbf{X}^{\setminus(s)} = \mathbf{x}^{\setminus(s)}\} = \mathbb{P}\{Y_s = 1 \mid \mathbf{X}^{(\mathcal{S})} = \mathbf{x}^{(\mathcal{S})}\}, \tag{4}$$

showing that the conditional probability of $Y_s = 1$ given all variables except $X^{(s)}$ is equal to the probability given only the signal variables $\mathbf{X}^{(\mathcal{S})}$. This implies that noisy features outside the signal set do not alter this probability.

The critical step for (4) to hold is the condition

$$Y_s \perp X^{(j)} \mid \mathbf{X}^{(\mathcal{S})}, \forall j \in \mathcal{N} \setminus \{s\}, \tag{5}$$

which asserts that, conditional on the signal set, the rule membership label Y_s is independent of all other noisy variables.

This relationship holds under the assumption that s is a noisy variable. Decision trees construct rules as conjunctions of univariate split conditions, so each rule corresponds to a rectangular region defined by inequalities involving a subset of variables. When the rule is released along coordinate s, the original and released regions differ only in the constraint on $X^{(s)}$. Consequently, the event $Y_s = 1$ (indicating membership in the released region) depends only on $X^{(s)}$. By the factorization assumption (1), $X^{(j)} \perp X^{(s)} \mid \mathbf{X}^{(\mathcal{S})}$ for all $j \in \mathcal{N} \setminus \{s\}$, which implies that $X^{(j)} \perp Y_s \mid \mathbf{X}^{(\mathcal{S})}$. Thus (5) holds, validating (4).

We can strengthen the above analysis by partitioning the signal variables into s-dependent and s-independent groups. This allows us to refine our understanding of how signal variables influence the classifier.

Definition 2. For a given $s \in \mathcal{S}$, let $\mathcal{S}_s \subseteq \mathcal{S}$ denote the set of s-dependent signal variables. The complement $\mathcal{S} \setminus \mathcal{S}_s$ consists of those signal variables conditionally independent of $X^{(s)}$ given the remaining elements of $\mathcal{S}_s$, that is

$$X^{(j)} \perp X^{(s)} \mid \mathbf{X}^{(\mathcal{S}_s \setminus \{s\})}, \quad j \in \mathcal{S} \setminus \mathcal{S}_s.$$

If s is noisy, then $\mathcal{S}_s$ does not include s, so $X^{(j)} \perp X^{(s)} \mid \mathbf{X}^{(\mathcal{S}_s)}$ for all $j \in \mathcal{S} \setminus \mathcal{S}_s$.

The set $\mathcal{S}_s$ identifies the subset of signal variables that are conditionally dependent on s. These are the variables that share information with s beyond what is captured by the rest of the signal set. In contrast, the s-independent variables in $\mathcal{S} \setminus \mathcal{S}_s$ are those for which the association with s vanishes once the s-dependent variables are held fixed. This partitioning is useful because it isolates the subset of signal variables relevant for interpreting the effect of s in the local classifier. This leads to the following result.

Theorem 1. *Under the joint factorization assumption (1), if s is a noisy variable, then the conditional probability in the two-class formulation depends only on the s-dependent signal features, $\mathcal{S}_s$.*

4.4. Justification for signal features

Now consider a released variable $s \in \mathcal{S}$ that is a signal feature. The conditional probability can be decomposed as

$$\mathbb{P}\{Y_s = 1 \mid \mathbf{X}^{\setminus(s)} = \mathbf{x}^{\setminus(s)}\}$$
$$= \mathbb{P}\{Y_s = 1 \mid \mathbf{X}^{(\mathcal{S} \setminus \{s\})}, \mathbf{X}^{(\mathcal{N})} = \mathbf{x}^{(\mathcal{N})}\}$$

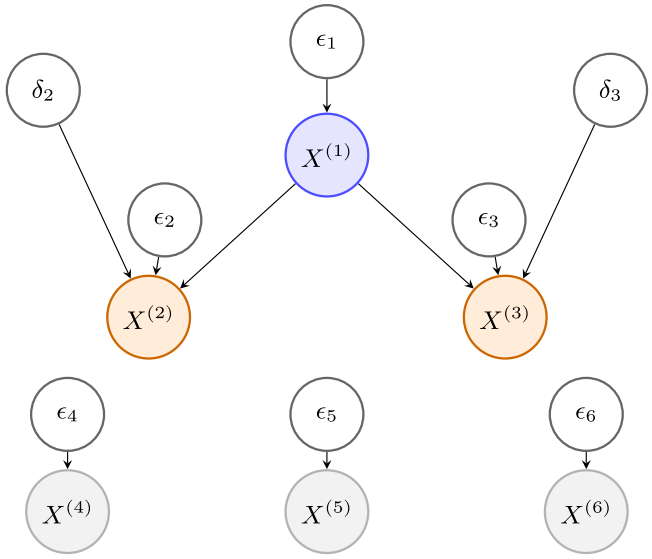

Fig. 3. Graphical model of Example 2. Releasing the signal variable $X^{(1)}$ (blue) alters the conditional distribution of the noisy variables $X^{(2)}$ and $X^{(3)}$ (orange) with respect to the proxy classification label Y_s, making them s-dependent noise ($s = 1$). The remaining noisy variables $X^{(4)}$, $X^{(5)}$, and $X^{(6)}$ (gray) however are unaffected and are classified as s-independent noise. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$$= \frac{\mathbb{P}\{Y_s = 1 \mid \mathbf{X}^{(\mathscr{S}\setminus\{s\})}, \mathbf{X}^{(\mathcal{N})} = \mathbf{x}^{(\mathcal{N})}\}}{\mathbb{P}\{Y_s = 1 \mid \mathbf{X}^{(\mathscr{S}\setminus\{s\})}\}} \times \mathbb{P}\{Y_s = 1 \mid \mathbf{X}^{(\mathscr{S}_s\setminus\{s\})} = \mathbf{x}^{(\mathscr{S}_s\setminus\{s\})}\}. \tag{6}$$

The last line follows because variables in $\mathscr{S} \setminus \mathscr{S}_s$ are s-independent: for any $j \in \mathscr{S} \setminus \mathscr{S}_s$, $X^{(j)} \perp X^{(s)} \mid \mathbf{X}^{(\mathscr{S}_s\setminus\{s\})}$. Since Y_s depends only on $X^{(s)}$, it is also conditionally independent of such j. Hence $\mathbb{P}\{Y_s = 1 \mid \mathbf{X}^{(\mathscr{S}\setminus\{s\})}\} = \mathbb{P}\{Y_s = 1 \mid \mathbf{X}^{(\mathscr{S}_s\setminus\{s\})}\}$, which yields the stated decomposition.

The second term in (6) depends only on s-dependent signal variables, similar to the noisy case. However, the first term may also involve noisy variables. To describe this contribution, we introduce the following definition.

Definition 3. Define $\mathcal{N}_s \subseteq \mathcal{N}$ to be the s-dependent noisy variables and $\mathcal{N} \setminus \mathcal{N}_s$ the s-independent noisy variables, where $\mathcal{N} \setminus \mathcal{N}_s$ is the largest subset satisfying: $X^{(j)} \perp Y_s \mid \mathbf{X}^{(\mathscr{S}\setminus\{s\})}$ for all $j \in \mathcal{N} \setminus \mathcal{N}_s$.

Intuitively, s-dependent noisy variables are those whose distributions shift for the regions defined by $Y_s = 0, 1$, even after conditioning on the signal set. In contrast, s-independent noise variables remain conditionally invariant. Fig. 3 illustrates this distinction.

Using Definition 3, we can rewrite (6) as

$$\frac{\mathbb{P}\{Y_s = 1 \mid \mathbf{X}^{(\mathscr{S}\setminus\{s\})}, \mathbf{X}^{(\mathcal{N}_s)} = \mathbf{x}^{(\mathcal{N}_s)}\}}{\mathbb{P}\{Y_s = 1 \mid \mathbf{X}^{(\mathscr{S}\setminus\{s\})}\}} \times \mathbb{P}\{Y_s = 1 \mid \mathbf{X}^{(\mathscr{S}_s\setminus\{s\})} = \mathbf{x}^{(\mathscr{S}_s\setminus\{s\})}\}.$$

This alternate form provides useful intuition: the s-dependent noisy variables of the first term modulate the probability of entering the released region, even after conditioning on the remaining signal features. In contrast, s-independent noisy variables do not affect this likelihood. From this, we obtain the following result.

Theorem 2. *Under the factorization assumption (1), if s is a signal variable, then the conditional probability from the two-class formulation depends only on the s-dependent noisy variables $\mathcal{N}_s$ and the signal variables $\mathscr{S} \setminus \{s\}$. Moreover, if there are no s-dependent noisy variables (i.e., $\mathcal{N}_s = \emptyset$), then the conditional probability depends only on $\mathscr{S}_s \setminus \{s\}$.*

4.5. UVarPro procedure and algorithm

Theorems 1 and 2 provide the theoretical foundation for the UVarPro procedure. When $X^{(s)}$ is a noisy variable, releasing s yields a classification problem in which only the associated signal variables $\mathscr{S}_s$ should retain predictive power. Conversely, when $X^{(s)}$ is a signal variable, classification accuracy may also reflect contributions from s-dependent noisy variables in addition to the remaining signals $\mathscr{S} \setminus \{s\}$. This asymmetry allows UVarPro to distinguish signal from noise by aggregating evidence across multiple rules and released variables.

To operationalize this idea, UVarPro constructs a classification-based importance matrix that encodes how each variable responds to the exclusion of every other. The matrix is assembled from rule-based contrasts obtained from the forest as described below. For reference, the complete procedure is summarized in Algorithm 1.

Algorithm 1 UVarPro for Unsupervised Variable Selection.

Grow a multivariate unsupervised forest in autoencoder mode and collect terminal rules $\{\zeta_b\}_{b=1}^{B}$ with regions $R(\zeta_b)$.
for $b = 1, \dots, B$ **do**
  for $s = 1, \dots, p$ **do**
    Construct the s release region $R(\zeta_b^s)$ by dropping conditions on s; let $\mathrm{rel}_s(\zeta_b) = R(\zeta_b^s) \setminus R(\zeta_b)$ so that $R(\zeta_b^s) = R(\zeta_b) \cup \mathrm{rel}_s(\zeta_b)$.
    if $|\mathrm{rel}_s(\zeta_b)| = 0$ **then**
      Set $\boldsymbol{\iota}_{s,b} = \mathbf{0} \in \{0,1\}^p$ and continue to next s.
    end if
    Define $Y_{s,\zeta_b}(\mathbf{X}) = 0$ on $R(\zeta_b)$ and $Y_{s,\zeta_b}(\mathbf{X}) = 1$ on $\mathrm{rel}_s(\zeta_b)$.
    Fit lasso logistic regression of Y_{s,ζ_b} on $\mathbf{X}_{\setminus s}$ using samples in $R(\zeta_b^s)$.
    Record $\boldsymbol{\iota}_{s,b} \in \{0,1\}^p$ with $\boldsymbol{\iota}_{s,b}(s) = 0$ and, for $j \neq s$, $\boldsymbol{\iota}_{s,b}(j) = \mathbf{1}\{\hat{\beta}_{s,b}(j) \neq 0\}$.
  end for
end for
For each s, compute $\boldsymbol{\iota}_s = \frac{1}{B}\sum_{b=1}^{B} \boldsymbol{\iota}_{s,b}$ and stack $\mathbf{I} = \begin{bmatrix} \boldsymbol{\iota}_1 \\ \vdots \\ \boldsymbol{\iota}_p \end{bmatrix} \in [0,1]^{p\times p}$.
Return column scores $I^{(s)} = \frac{1}{p}\sum_{k=1}^{p} i_{ks}$ for $s = 1, \dots, p$.

4.5.1. Importance matrix construction

For each feature $X^{(s)}$ and each rule ζ_b generated by the unsupervised forest, UVarPro fits a localized lasso logistic regression to the corresponding release region, $R(\zeta_b^s)$. The pseudo-response is the binary indicator $Y_{s,\zeta_b}(\mathbf{X})$, while the predictors are the remaining features $\mathbf{X}^{\setminus(s)}$, ensuring that $X^{(s)}$ is excluded:

$$(\hat{\beta}_{0,s,b}, \hat{\boldsymbol{\beta}}_{s,b}) = \underset{\beta_0\in\mathbb{R},\ \boldsymbol{\beta}\in\mathbb{R}^{p-1}}{\operatorname{argmin}} \left\{ \sum_{\mathbf{x}_i \in R(\zeta_b^s)} \ell\Big(Y_{s,\zeta_b}(\mathbf{x}_i),\ \beta_0 + \mathbf{X}^{\setminus(s)}(\mathbf{x}_i)^\top \boldsymbol{\beta}\Big) + \lambda_{s,b} \|(\beta_0, \boldsymbol{\beta})\|_1 \right\},$$

with ℓ denoting logistic loss and $\lambda_{s,b}$ chosen by cross-validation.

The sparse solution $\hat{\boldsymbol{\beta}}_{s,b}$ is encoded as a binary selection vector $\boldsymbol{\iota}_{s,b} \in \{0,1\}^p$, with coordinate s set to zero by construction and $\boldsymbol{\iota}_{s,b}(j) = 1$ for $j \neq s$ whenever the corresponding coefficient is nonzero. These vectors capture which variables support classification when feature $X^{(s)}$ is released. Averaging across B rules yields the importance vector $\boldsymbol{\iota}_s = B^{-1}\sum_{b=1}^{B} \boldsymbol{\iota}_{s,b}$, a continuous measure of how often each variable is selected when $X^{(s)}$ is excluded. Stacking the p vectors produces the $p \times p$ importance matrix

$$\mathbf{I} = \begin{bmatrix} \boldsymbol{\iota}_1 \\ \boldsymbol{\iota}_2 \\ \vdots \\ \boldsymbol{\iota}_p \end{bmatrix} = \begin{bmatrix} i_{11} & \cdots & i_{1p} \\ \vdots & \ddots & \vdots \\ i_{p1} & \cdots & i_{pp} \end{bmatrix},$$

where each row $\boldsymbol{\iota}_s$ summarizes how other variables compensate when $X^{(s)}$ is released.

To obtain a single importance score for each variable, we sum its row entries,

$$I^{(s)} = \sum_{k=1}^{p} i_{ks}.$$

This measures how often $X^{(s)}$ contributes across all release-based comparisons. Theorems 1 and 2 establish that signal variables are repeatedly recovered in such tasks, even when they are not the variable being released. Summing therefore amplifies this recurrence, and elevates scores for true signals.

4.5.2. Refinement: Matrix shrinkage

Although our theory indicates that $\mathbf{I}$ separates signal from noise, incidental correlations and collinearity can inflate its entries in high dimensions. To address this, we apply a second-stage local lasso shrinkage that reweights and filters the row-wise selections.

Algorithm 2 Local Lasso Shrinkage of the Importance Matrix.

For each s, collect $\{\boldsymbol{\iota}_{s,b} : b = 1, \dots, B\}$ and $\{\mathrm{rel}_s(\zeta_b)\}$ from Algorithm 1.
for $s = 1$ to p **do**
  for $b = 1$ to B **do**
    if $\sum_{j \neq s} \boldsymbol{\iota}_{s,b}(j) > 0$ **then**
      Let $\mathcal{M} = \{j : \boldsymbol{\iota}_{s,b}(j) = 0\} \cup \{s\}$; predictors are *standardized* $\mathbf{X}_{\setminus \mathcal{M}}$.
      Using only samples in $\mathrm{rel}_s(\zeta_b)$, fit
      $$X^{(s)} \sim \mathbf{X}^{\setminus(\mathcal{M})} \quad \text{via lasso.}$$
      Store $\beta^{\mathrm{local}}_{s,b} \in \mathbb{R}^p$ with zeros on $\mathcal{M}$.
    else
      Set $\beta^{\mathrm{local}}_{s,b} = \mathbf{0}$.
    end if
  end for
end for
For each s, compute the averaged shrinkage vector $\beta^{\mathrm{local}}_s = \frac{1}{B}\sum_{b=1}^{B} \beta^{\mathrm{local}}_{s,b}$.
Form $\mathbf{I}^{\mathrm{local}}$ by the elementwise product
$$\boldsymbol{\iota}^{\mathrm{local}}_s = |\beta^{\mathrm{local}}_s| \odot \boldsymbol{\iota}_s,$$
and stack rows as in Algorithm 1.
Define the local column scores
$$I^{\mathrm{local}}(s) = \frac{1}{p}\sum_{k=1}^{p} i^{\mathrm{local}}_{ks}, \qquad s = 1, \dots, p.$$

Implementation note. Predictors are standardized before fitting, and intercept terms are excluded because they contain no variable-specific information. The coefficients $|\beta^{\mathrm{local}}_s|$ are applied elementwise to the corresponding row of $\mathbf{I}$ without clipping or additional normalization, and the column averages $I^{\mathrm{local}}(s)$ summarize these weighted entries.

For each rule ζ_b, once candidate signal variables have been identified, we run a second lasso regression within the release region, regressing $X^{(s)}$ on the selected features. This step removes weak or spurious predictors that survive the first pass. The resulting coefficients are then averaged across rules to form a local coefficient vector β^{local}_s for each feature s. Finally, we refine the importance matrix by scaling each row of $\mathbf{I}$ according to the absolute values of these local coefficients. A full description of the procedure is given in Algorithm 2.

5. Synthetic experiments

We evaluate UVarPro on a suite of synthetic benchmarks designed to probe distinct properties of unsupervised feature selection. This encompasses four scenario families: latent factor dependence (normal and uniform latent variables with both correlated and uncorrelated noise); structured clustering geometries (V-cluster, X-cluster, ellipse, and non-coexistence), each considered in original and noise-correlated versions; redundancy and conditional independence (causal chains and functionally redundant signals); and higher-dimensional experiments (multivariate normal with correlated noise and a normal mixture). Table 1 summarizes the designs with full data-generation details provided in Supplementary Materials. Results and comparative summaries are presented in the subsections that follow.

Table 1
Summary of synthetic experiments.

Simulation	Dimension	Dimension of Signal Features	Signal Feature Set
Normal Latent Variables	18	4	{w, x, y, z}
Normal Latent Variables Correlated	27	4	{w, x, y, z}
Uniform Latent Variables Correlated	27	4	{w, x, y, z}
Normal Latent Correlated High Dim.	105	12	**{W, X, Y, Z}**
Example 4	4	2	$\{X^{(1)}, X^{(4)}\}$
Example 5	4	3	$\{X^{(1)}, X^{(2)}, X^{(3)}\}$
Multivariate Normal	10	3	$\{X^{(1)}, X^{(2)}, X^{(3)}\}$
Normal Mixture Model	5	2	$\{X^{(1)}, X^{(2)}\}$
V-cluster	11	2	$\{X^{(1)}, X^{(2)}\}$
V-cluster Correlated	12	2	$\{X^{(1)}, X^{(2)}\}$
X-cluster	11	2	$\{X^{(1)}, X^{(2)}\}$
X-cluster Correlated	12	2	$\{X^{(1)}, X^{(2)}\}$
Non-coexistence	12	2	$\{X^{(1)}, X^{(2)}\}$
Non-coexistence Correlated	12	2	$\{X^{(1)}, X^{(2)}\}$
Ellipse	16	6	$\{X^{(1)}, \dots, X^{(6)}\}$
Ellipse Correlated	10	4	$\{X^{(1)}, \dots, X^{(4)}\}$

5.1. Monte carlo replications and performance evaluation

Each simulation scenario is defined by a known data-generating model with an oracle signal set, the subset of features that truly determine dependence in the joint distribution. Because the oracle model is known, independent datasets can be generated where the identity of signal and noise variables is specified exactly, enabling objective evaluation of each method's ability to recover the true signal set.

We performed 100 independent Monte Carlo replications, each using a fresh dataset drawn from the oracle model. All methods, including UVarPro, were fit entirely within the generated sample. No cross-validation, tuning splits, or data reuse occurred, as each replication represents a fully independent, out-of-sample evaluation with known ground truth, eliminating any possibility of information leakage. Performance was measured by the Area Under the Receiver Operating Characteristic Curve (AUC-ROC), quantifying each method's ability to distinguish oracle signal from noise features. Results are reported over the 100 independent replications.

5.2. Comparison procedures

The following competing methods were included for comparison: FSCA [10], L-FSCA [11], and FSFP [13], implemented in R based on the original algorithm descriptions. PC-simple [33] was run using the `pcalg` package in R. MIC was implemented using the `minerva` package, and the HHG test via the `HHG` package. For MCFS [16], UDFS [17], SPEC [15], and Laplacian score (LapScore) [14], we used the Python-based `scikit-feature` repository. UVarPro and the UVarPro-lasso were run in autoencoder mode, with other parameters set at defaults (see Section 6.2). Comparison methods were run using recommended default configurations.

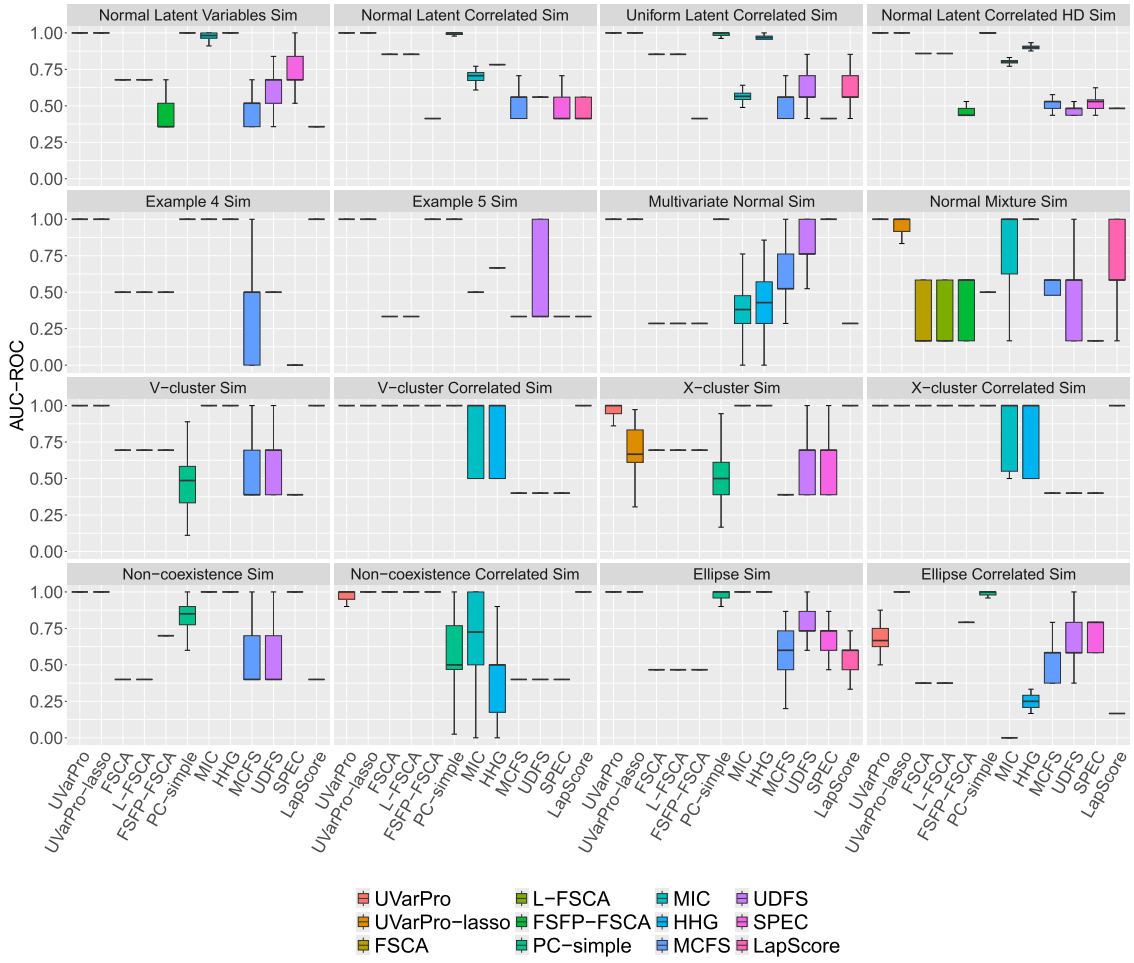

Fig. 4. AUC-ROC performance values over synthetic simulations for UVarPro, UVarPro-lasso, and competing unsupervised feature selection methods. Higher values indicate better identification of the ground truth signal set.

5.3. *Empirical results*

Fig. 4 summarizes performance across the simulations. In the Latent Variable Simulation series (first row), which includes the normal, correlated, uniform, and high-dimensional designs, both UVarPro methods consistently achieve perfect AUC-ROC scores. PC-simple also performs well, but its accuracy is affected by correlation. Greedy search procedures, similarity-based approaches, and sparse learning methods are generally ineffective across this group, while dependency-based techniques such as MIC and HHG show early promise but degrade with correlation.

The second row considers more structurally complex settings. Examples 4 and 5, which involve redundancy and nonlinear interactions, are difficult for most baseline methods, yet the proposed method achieves perfect or near-perfect performance. In the Multivariate Normal Simulation, where noise variables are correlated with all signals, most methods exhibit a sharp drop in accuracy, but UVarPro remains highly effective. In the Normal Mixture Simulation, even though noise is independent, marginal structure alone is not enough for many competitors; only UVarPro, HHG, and MIC maintain reliable accuracy.

Several additional designs are organized as paired structures: V-cluster, X-cluster, Non-coexistence, and Ellipse, each with independent and correlated noise versions. These settings probe stability under correlation among noise variables. UVarPro continues to deliver near-perfect accuracy throughout, whereas competing approaches degrade substantially when noise correlation is present, with HHG, MIC, and PC-simple most affected.

To consolidate these findings across all designs, we used critical difference (CD) plots [34] at the 5 % significance level, ranking methods by average AUC and applying the Nemenyi test (see Fig. 5). UVarPro and UVarPro-lasso rank at the extreme left, well separated from all competitors. These rankings are consistent with the patterns seen in individual simulations: PC-simple frequently identifies only part of the signal set, MIC and HHG are sensitive to correlated noise, greedy search struggles when noise variables have variances comparable to those of the signals, and similarity or sparsity-based methods designed for high-dimensional clustering tasks perform poorly in structured low-dimensional problems.

Although UVarPro and its lasso extension perform similarly overall, some differences emerge in specific settings. In designs such as the Normal Mixture and X-cluster, UVarPro's direct elevation of group-level dependencies provides a natural advantage, with little added benefit from shrinkage. On the other hand, in the Non-coexistence Correlated and Ellipse Correlated simulations, where signals are weak and collinearity is strong, the extra regularization helps filter surrogates and gives UVarPro-lasso a slight edge. These differences are modest, however, and both procedures are highly effective across experiments.

6. Real-World data experiments

This section evaluates UVarPro across a diverse collection of real-world datasets from multiple application domains, supplemented by two challenging synthetic benchmarks (Cobra8 and Friedman1) used to assess runtime, scalability and robustness. The analysis is organized

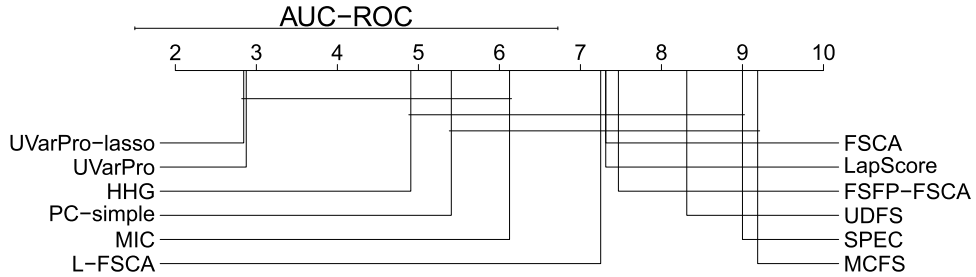

Fig. 5. Average rank of each method across the synthetic simulations, based on AUC-ROC for each dataset. Lower ranks indicate stronger overall performance. Critical difference (CD) plots are shown at the 5 % significance level; methods not connected by a horizontal bar differ significantly.

into four parts: (i) a parameter sensitivity study on real data to evaluate robustness under different configurations; (ii) experiments on Cobra8 and Friedman1 examining performance relative to CPU runtime, dimensionality scaling, memory usage, and robustness to feature correlation, as well as the effect of the subtask learner; (iii) assessment of feature-selection accuracy on labeled real datasets by comparing variables identified by UVarPro with those selected by supervised methods; and (iv) clustering benchmarks on six publicly available high-dimensional datasets, followed by a detailed biological case study with external validation.

6.1. Tuning parameters

This analysis examines how key parameters in unsupervised forests affect the accuracy of selected features, focusing on the size of terminal nodes, the number of rules, and the choice of response coordinates at each split. In the autoencoder configuration of Algorithm 1, the feature vector serves as both input and response, so all p features are used simultaneously when evaluating splits, and multivariate impurity is computed over the full set of coordinates. In the more general framework of [32], which we investigate here, only a random subset of size *ytry* is drawn from the p coordinates at each node. Impurity is then computed using this subset, and the predictor-side split variable is chosen to maximize impurity reduction. The splitting rule itself is unchanged, but the number of response coordinates influencing it varies with *ytry*.

Two other parameters refine the structure of trees and the stability of importance scores. The *nodesize* parameter sets the minimum number of observations in a terminal node, with smaller values producing deeper trees and finer partitions, and larger values yielding broader release regions with more observations for local contrasts, which can reduce variance. The number of sampled rules, B, controls the precision of the aggregated importance measure: larger B values reduce variability in importance scores, but at the cost of increased computation time. The trade-off between stability and efficiency is examined further in Section 6.2.

We evaluated performance over a grid of B, *ytry*, and *nodesize* values, comparing full response sampling (*ytry* $= p$, autoencoder mode) with reduced pseudo-response sampling (*ytry* $< p$), and examining how terminal node size, the number of rules, and the choice of response coordinates affect feature selection accuracy.

6.1.1. PeakVO2 data

The analysis is applied to the PeakVO2 survival dataset, which includes 2231 observations and 41 variables. The data were collected from patients with systolic heart failure who underwent cardiopulmonary stress testing. The endpoint of interest is all-cause mortality. Additional details on the dataset and study design are provided in [2].

6.1.2. Tuning parameter results

In earlier synthetic simulations we measured performance using AUC-ROC. For the PeakVO2 study and the higher-dimensional settings considered later, we instead report AUC-PR, which is more reliable when informative variables are sparse and represent only a small fraction of the features. To define a reference set of signals, we use the supervised VarPro method [2], introduced in Section 4, and take the variables it selects as the ground truth.

Fig. 6 reports performance relative to this supervised standard. Each panel corresponds to a different *ytry* value. The horizontal axis shows the minimum terminal node size (*nodesize*), varied from 10 to 200, and lines within each panel indicate the number of rules per tree (1, 5, 10, 20). With the forest size fixed at 200 trees, the total number of sampled rules is $B = 200 \times$ (rules per tree); larger values provide more aggregation and typically improve stability, though at additional computational cost. Curves display mean AUC-PR over 100 repetitions where each repetition used a fresh 75 % data subsample (without replacement).

The results highlight the influence of both *nodesize* and *ytry* on feature selection accuracy. Increasing *nodesize* stabilizes local contrasts and improves performance, while higher *ytry* values consistently raise AUC-PR by providing more response coordinates at each split. The number of rules per tree also matters: performance is poor when only a single rule is used but improves markedly with two or more, after which gains are minimal.

6.2. Performance versus CPU runtimes

We next examined how hyperparameter choices affect both feature selection accuracy and computational cost, with the aim of identifying practical settings that balance performance and efficiency. CPU times were recorded in seconds, and accuracy was measured by AUC-PR with the true set of signal variables defined by the simulation models. To make results comparable across problems, we summarize them in terms of the complexity ratio p/n (number of features relative to sample size).

We considered two well-known regression simulations. The first is the Friedman1 model,

$$Y = 10\sin(\pi X^{(1)}X^{(2)}) + 20(X^{(3)} - 0.5)^2 + 10X^{(4)} + 5X^{(5)} + \varepsilon,$$
$$X^{(j)} \sim U(0,1),\ \varepsilon \sim N(0,1),$$

a nonlinear function with five signal variables. The second is Cobra8,

$$Y = \frac{1}{1 + \exp\left(-30\left(X^{(1)} + (X^{(4)})^3 + X^{(9)} + \sin(X^{(2)}X^{(8)}) + \varepsilon - 0.38\right)\right)},$$
$$X^{(j)} \sim \mathrm{U}(-0.25, 1),\ \varepsilon \sim \mathrm{N}(0, 0.1^2),$$

which is more challenging due to strong nonlinear interactions and higher-order terms. More details regarding these models can be found in [2].

A default configuration was defined as

$$\textit{nodesize} = \max\{\, n/10,\ \min(n, 20)\,\},$$
$$\textit{ytry} = \sqrt{p},$$
$$\textit{ntree} = 200,$$
$$\textit{rules per tree} = 10.$$

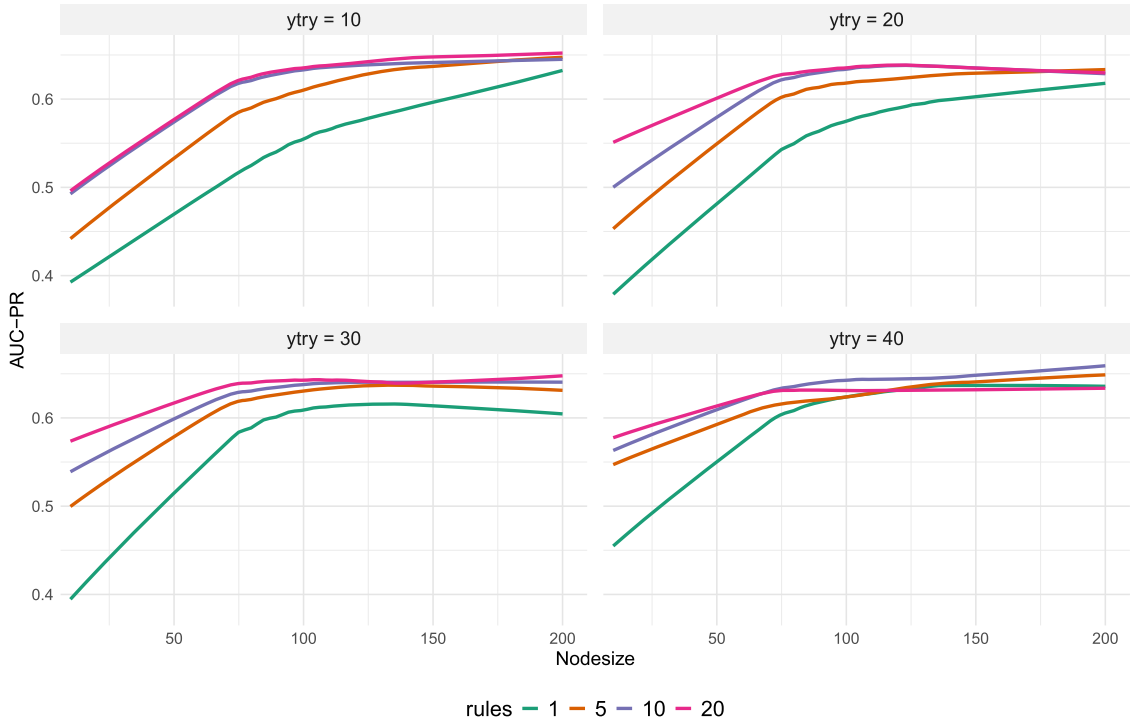

Fig. 6. Feature selection performance on the PeakVO2 dataset measured by AUC-PR. Each panel corresponds to a distinct *ytry* value. The horizontal axis shows the minimum terminal node size (*nodesize*), and lines trace AUC-PR as the number of rules per tree varies (1, 5, 10, 20).

Variations of this baseline were also considered: (i) *autoencoder* mode, with $ytry = p$; (ii) *double*, with *nodesize* doubled relative to the default; and (iii) the combination of autoencoder and double. Thus, four conditions were evaluated in total.

Complexity values (p/n) were obtained by varying both sample size and dimension across a grid of values, with n ranging from 200 to 2000 and p ranging from 20 to 500.

6.2.1. A note about independence

Because the simulated features are generated independently, there is no intrinsic dependence structure to reveal signal variables. To make the unsupervised analysis meaningful, we therefore treat the augmented collection $(Y, \mathbf{X})$ as the feature vector and include the outcome Y as an observed coordinate. Let $\mathscr{S} \subset \{1, \ldots, p\}$ index the coordinates on which the outcome depends, where

$$Y = g\left(\mathbf{X}^{(\mathscr{S})}, \varepsilon\right), \quad X^{(1)}, \ldots, X^{(p)} \text{ are mutually independent,}$$
$$\varepsilon \text{ is independent of } \mathbf{X}.$$

Under these conditions $Y \perp \mathbf{X}^{\setminus(\mathscr{S})} \mid \mathbf{X}^{(\mathscr{S})}$, and the remaining coordinates become mutually independent when conditioning on the pair $(Y, \mathbf{X}^{(\mathscr{S})})$. Equivalently,

$$f_{\mathbf{X}^{\setminus(\mathscr{S})} \mid Y, \mathbf{X}^{(\mathscr{S})}}\left(\mathbf{x}^{\setminus(\mathscr{S})} \mid y, \mathbf{x}^{(\mathscr{S})}\right) = f_{\mathbf{X}^{\setminus(\mathscr{S})} \mid \mathbf{X}^{(\mathscr{S})}}\left(\mathbf{x}^{\setminus(\mathscr{S})} \mid \mathbf{x}^{(\mathscr{S})}\right)$$
$$= f_{\mathbf{X}^{\setminus(\mathscr{S})}}(\mathbf{x}^{\setminus(\mathscr{S})}) = \prod_{j \notin \mathscr{S}} f_{X^{(j)}}(x^{(j)}),$$

so $(Y, \mathbf{X}^{(\mathscr{S})})$ is a valid signal set for the augmented feature vector $(Y, \mathbf{X})$ in the sense of Definition 1. In reporting performance, we record recovery of the $\mathbf{X}$ coordinates in $\mathscr{S}$.

6.2.2. CPU and performance results

Results for the unsupervised analysis are shown in Fig. 7. The top row displays AUC-PR, with Cobra8 on the left and Friedman1 on the right. AUC-PR decreases with increasing complexity (p/n), with a sharper decline for Cobra8. The four hyperparameter conditions yield clearly separated curves. For Cobra8, autoencoder + double has the largest AUC-PR. For Friedman1, autoencoder + double again performs best, with autoencoder a close second.

The bottom row of Fig. 7 shows CPU runtimes. Runtimes initially increase with complexity, then decrease and stabilize. This pattern is due to sparsity induced by lasso regularization. As dimensionality grows, many coefficients are shrunk to zero, reducing the set of predictors that enter the second local lasso step, thereby lowering computation. Overall, the default setting produces the lowest runtimes, while double produces the longest runtimes due to the cost of fitting larger terminal nodes. Autoencoder increases runtimes further because each split considers all p variables. Interestingly, the autoencoder + double setting is often faster than autoencoder alone. The likely explanation is that the larger nodes in the double configuration reduce the number of candidate rules available for random selection, which partially offsets the added cost of autoencoder splitting.

In summary, if runtimes are not an issue we recommend autoencoder or even using autoencoder + double. For large-scale applications, the default setting is a sensible and practical starting point. If accuracy under the default is inadequate, performance could be improved by increasing *ytry* towards autoencoder if computations allow.

6.3. Computation and memory costs

The stabilization of runtimes observed in Fig. 7 raises a natural question about dimension scalability. In those plots the horizontal axis is expressed as p/n, so both p and n vary. This mixing of sample size and dimensionality effects makes it difficult to isolate the dependence of runtime on p.

To disentangle these factors, we conducted additional large-p experiments using Cobra8 and Friedman1 with sample size fixed at $n = 100$ and feature dimension ranging from $p = 100$ to 10,000 (Supplementary Figure SF1). A full derivation of runtime and memory requirements is provided in the Supplementary Materials; here we summarize the main results. Under the most aggressive autoencoder mode with $ytry = p$, CPU time increases linearly in p, dominated by the $O(ntree \cdot n \log n \cdot p)$ cost of

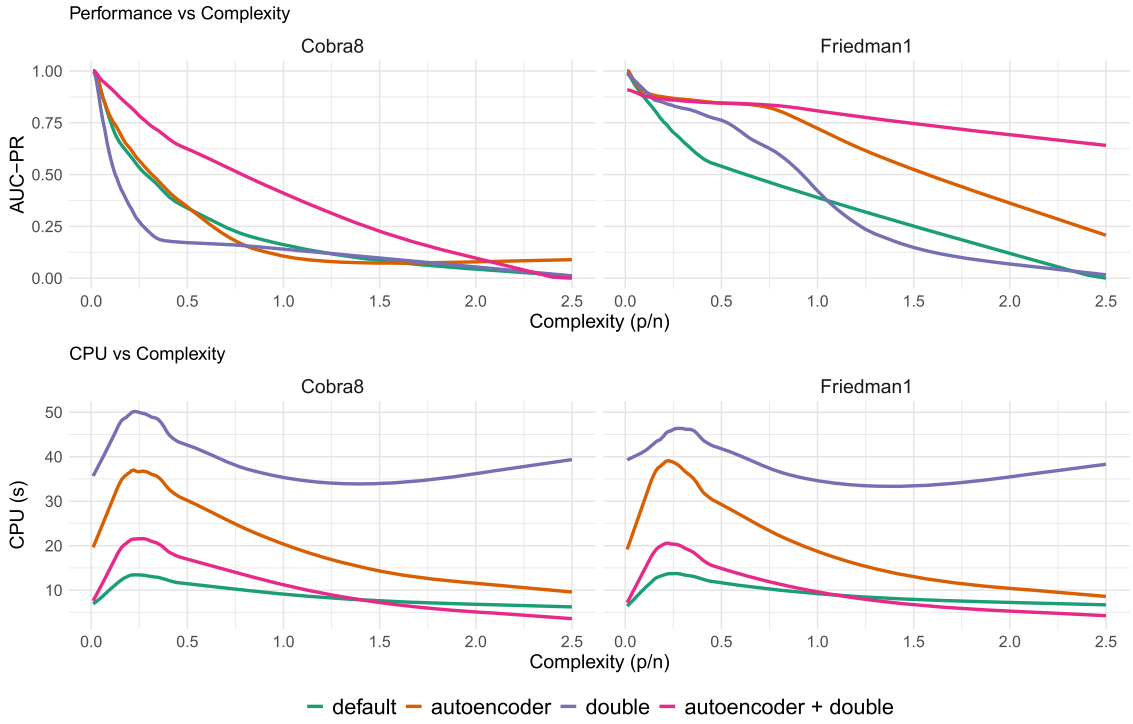

Fig. 7. Performance and CPU runtimes as a function of complexity (p/n) for Cobra8 (left) and Friedman1 (right) simulations. Top row: AUC-PR decreases with increasing complexity, with autoencoder + double yielding the strongest performance and autoencoder a close second on Friedman1. Bottom row: CPU runtimes rise with complexity and then decline due to lasso-induced sparsity.

forest construction. Local tasks remain bounded so long as $n/nodesize \leq C$, for some fixed constant, as is the case here.

Memory requirements are modest. Data storage is $O(np)$, while forest storage is $O(ntree \cdot n/nodesize)$, which remains constant in $ntree$ if $n/nodesize \leq C$. The importance matrix requires $O(Kp)$ in sparse form, with at most K nonzeros per column ($K < p$). Peak memory during the local lasso steps is determined primarily by the size of the working set rather than by p itself.

6.4. Performance in correlated feature spaces

As noted in Section 3, when variables have overlapping roles the signal set is not unique, so recovery should be understood at the group rather than the individual level. To explore this empirically, we revisited the Friedman1 and Cobra8 simulations under a dense correlated design using autoencoder + double mode. The sample size was fixed at $n = 1000$, p was varied, and performance was measured by AUC-PR.

The experimental setup was the same as in the original benchmarks, except that signal variables were replaced with correlated versions. Specifically, the uniform signals were resampled from a Gaussian copula with exchangeable correlation 0.9 and uniform margins. This created strong dependence among the signals while preserving their marginal distributions and their functional relationship with Y.

Results in Fig. 8 show that strong correlation among signals reduces performance. AUC-PR is consistently lower than in the independent design. For Cobra8 the decline is smaller, and even with one or two hundred variables the method remains reasonably accurate, likely because the larger pool of signals increases the chance that some remain distinguishable. Thus, UVarPro can still identify informative variables in correlated spaces, but reliable recovery requires a sufficiently large signal set.

6.5. Effect of the surrogate learner

Our default surrogate for the release-region classifier is logistic regression with an ℓ_1 penalty (lasso). To examine sensitivity to this choice, we compared three alternative learners: an elastic net (logistic regression with mixed ℓ_1/ℓ_2 penalty, $\alpha = 0.5$), an unpenalized logistic regression (GLM), and a linear support vector machine (SVM) with regularization parameter $C = 1$, which controls the trade-off between margin width and misclassification error.

For each rule-release task, the pseudo-labels distinguishing a region from its near-miss set were classified using the same collection of rules, ensuring that differences reflect only the surrogate learner. Variable scores from each task were aggregated to form global importance rankings. We used the *Friedman1* and *Cobra8* simulations with $n = 1000$ and p varying from 20 to 100; each experiment repeated for 100 independent Monte Carlo replications.

Fig. 9 summarizes AUC-PR results versus p. Two consistent patterns emerge. (1) Logistic lasso and elastic net achieve the highest AUC-PR values across all dimensions, indicating that embedded regularization stabilizes small-sample fits and suppresses noisy predictors. (2) The unpenalized GLM is the weakest baseline, particularly as dimension increases, while the linear SVM shows a clear degradation as p grows and does not provide a systematic improvement over the penalized logistic models. Because the release-region contrasts are locally simple and often nearly linearly separable, explicit margin optimization offers little benefit in this setting, and the absence of embedded sparsity in the linear SVM becomes increasingly limiting with higher p.

Overall, logistic lasso provides the best trade-off between accuracy, stability, and computational efficiency.

6.6. Performance across data modalities

We next assess how reliably UVarPro recovers informative variables across different data modalities, using supervised methods as gold-standard references. FSCA is included as a benchmark, and UVarPro is run in autoencoder + double mode. The datasets considered are: (i) Glioma, a genomic classification problem for predicting disease subtypes; (ii) Iowa Housing, a regression task predicting sale prices from demographic and structural characteristics; and (iii) PeakVO2,

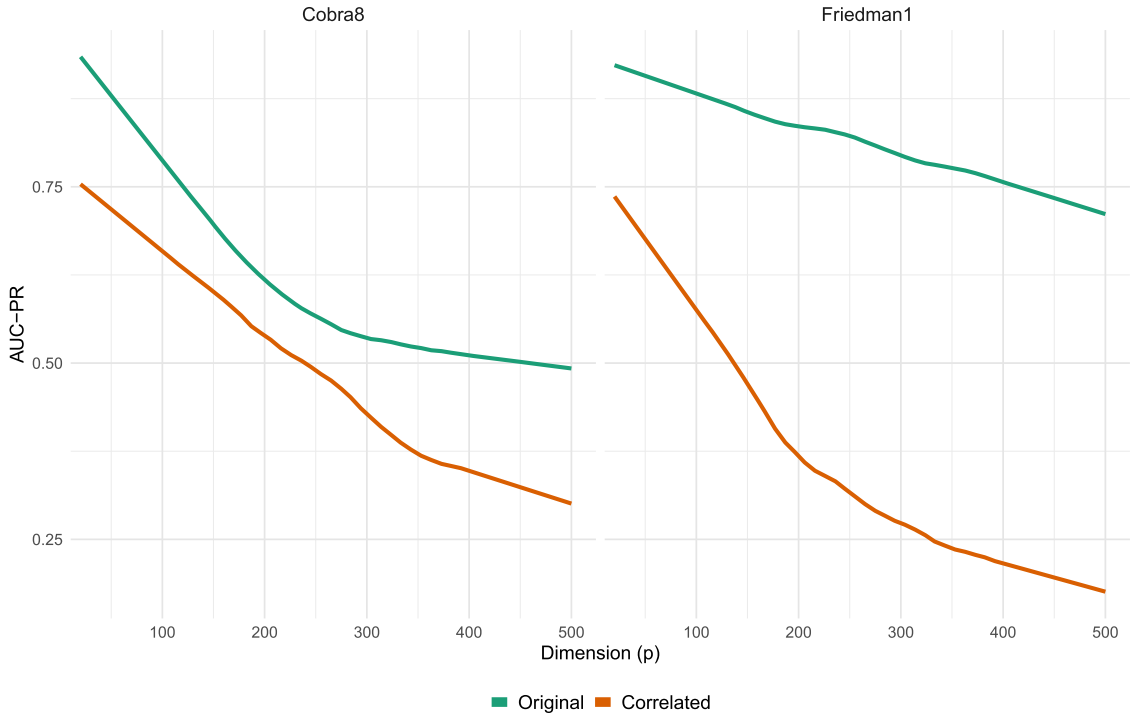

Fig. 8. Performance under correlated signal settings. Results for Cobra8 (left) and Friedman1 (right) show that AUC-PR decreases with increasing p, with consistently lower values when the signal variables are correlated. The gap is smaller for Cobra8 due to its larger number of signal features.

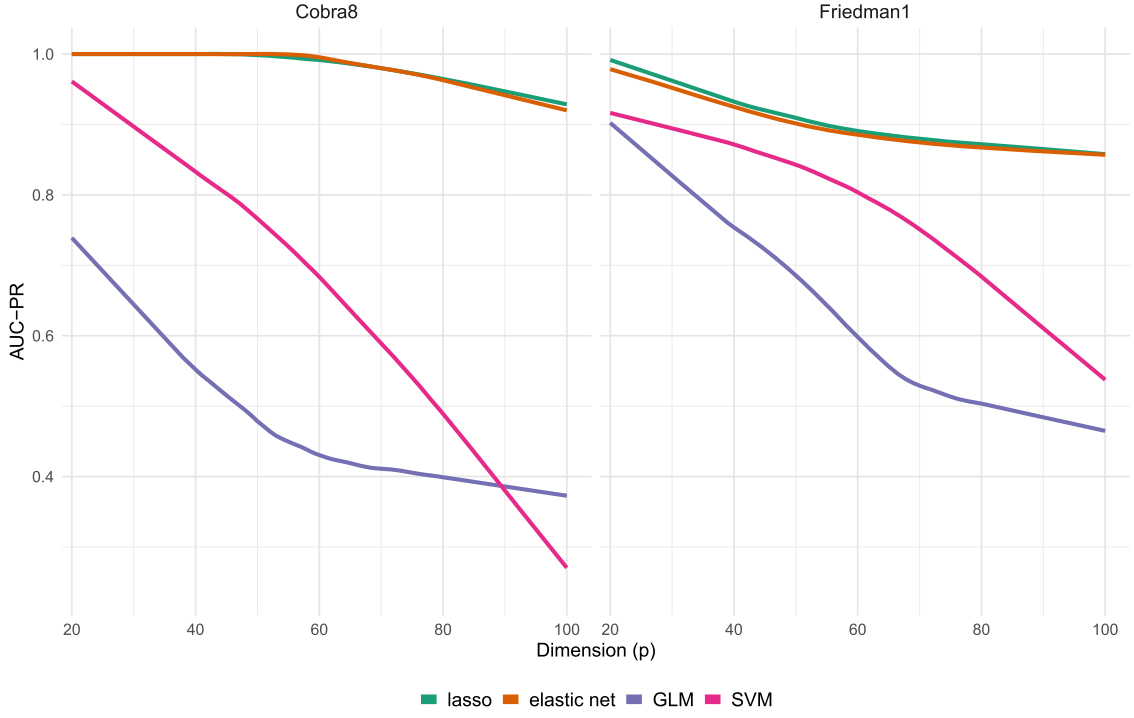

Fig. 9. AUC-PR of variable selection versus dimension p for four surrogate learners (logistic lasso, elastic net, GLM, and linear SVM), averaged over 100 replications. Panels: Cobra8 (left) and Friedman1 (right).

a survival dataset described in Section 6.1. Further dataset details are provided in [2].

To avoid bias from any single importance metric, we employ three supervised reference standards: VarPro; Gradient Boosting, which ranks variables by cumulative loss reductions across tree splits; and permutation-based importance (VIMP) from random forests.

Fig. 10 summarizes AUC-PR over 100 runs per dataset using 75 % subsampling. The results from this analysis show that across all datasets and reference standards, UVarPro consistently outperforms FSCA. Performance varies with data complexity, with the lowest AUC-PR observed for the Glioma dataset, an expected result given its high dimensionality and multi-class structure, both of which present intrinsic challenges for unsupervised feature selection.

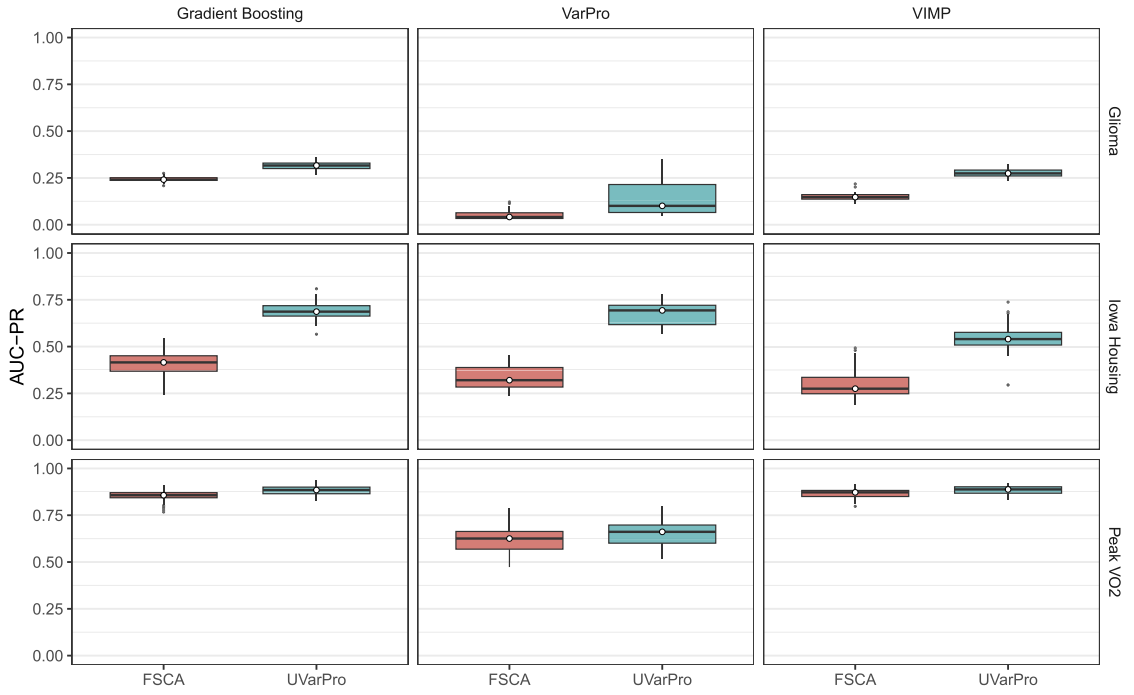

Fig. 10. Comparison of feature selection precision between UVarPro and FSCA across three datasets: *Glioma* (classification), *Iowa Housing* (regression), and *PeakVO2* (time-to-event). The boxplots display the precision values achieved relative to the three supervised gold standard methods: VarPro, Gradient Boosting, and VIMP.

Table 2
Summary of high-dimensional datasets used in the benchmark analysis.

Dataset	Instances (n)	Features (p)	Clusters (l)	Type
LUNG	203	3312	5	Biological
TOX_171	171	5748	4	Biological
GLIOMA	50	4434	4	Biological
ALLAML	72	7129	2	Biological
COIL20	1440	1024	20	Face Image
USPS	9298	256	10	Handwritten Image

6.7. Clustering performance in high-dimensional settings

To evaluate performance in high-dimensional contexts, we use six well-known datasets from biology and image analysis: LUNG, TOX_171, GLIOMA, ALLAML, COIL20, and USPS. All are drawn from the ASU Feature Selection Repository and provide a diverse testbed for unsupervised feature selection under challenging conditions. Summary statistics and dataset characteristics are given in Table 2.

6.7.1. Evaluation framework via clustering

Because ground truth labels are available for these datasets, we assess feature selection quality by testing how well the selected features recover the underlying cluster structure, without using labels during selection. Clustering performance is evaluated in two steps. First, we compare UVarPro and UVarPro-lasso with established unsupervised methods (MCFS, UDFS, LapScore, SPEC, FSCA, and MIC). Second, we test robustness by varying both the clustering backend and the evaluation metric, fixing UVarPro-lasso as the feature selection procedure.

Backends for Comparison Across Procedures

(i) *K-means.* We set the number of clusters to the true value l and run K-means, reporting the average NMI over 20 random initializations.
(ii) *sidClustering.* We also use sidClustering [35], evaluated with normalized entropy scores that penalize small, fragmented clusters while rewarding both purity and size. Lower values indicate better clustering performance.

Backends for Robustness Across Clustering Pipelines

(iii) *Spectral clustering.* We build a k-nearest neighbor graph with Gaussian affinities $w_{ij} = \exp\big(-\|x_i - x_j\|^2/(\sigma_i\sigma_j)\big)$ using self-tuning local scales, form the symmetric normalized Laplacian, embed into the first l eigenvectors with smallest eigenvalues, row normalize, and run K-means in the spectral space (20 restarts). We sweep $k \in \{5, 10, 15\}$ and report the median result.
(iv) *Gaussian mixture model (GMM).* We fit an l-component mixture by EM, initialized with K-means. Both diagonal and full covariance families are considered with a small ridge for stability, and the better model is selected by BIC. For these backends we also report ARI and Silhouette scores (the latter computed in the scaled feature space with Euclidean distance).

Protocol

For the comparison benchmarking, we evaluate clustering on reduced feature sets obtained from each procedure where number of selected features m was varied as $m \in \{20, 40, 60, 80, 100, 120, 140, 160, 180, 200\}$. Each competing procedure is tuned over its internal parameters, including UVarPro, and we report its best performance. For the across-backend robustness study we fix the procedure to UVarPro-lasso, where for computational speed we set tuning parameters to default values. Results for the comparison study are provided in Figs. 11 and 12. Results from the cross-backend study are provided in Table ST1 of the Supplementary Materials.

6.7.2. Findings

Comparison Across Procedures. NMI performance reported in Fig. 11 show that over LUNG and TOX_171, UVarPro and UVarPro-lasso dominate competing methods across all values of m. On GLIOMA and ALLAML, UVarPro-lasso attains the highest overall performance, with UVarPro close behind and ahead of most baselines. On COIL20, where n and p are comparable, FSCA is competitive while UVarPro-lasso remains among the top methods. On USPS, where $n > p$, the unpenalized UVarPro outperforms UVarPro-lasso. Normalized entropy scores from

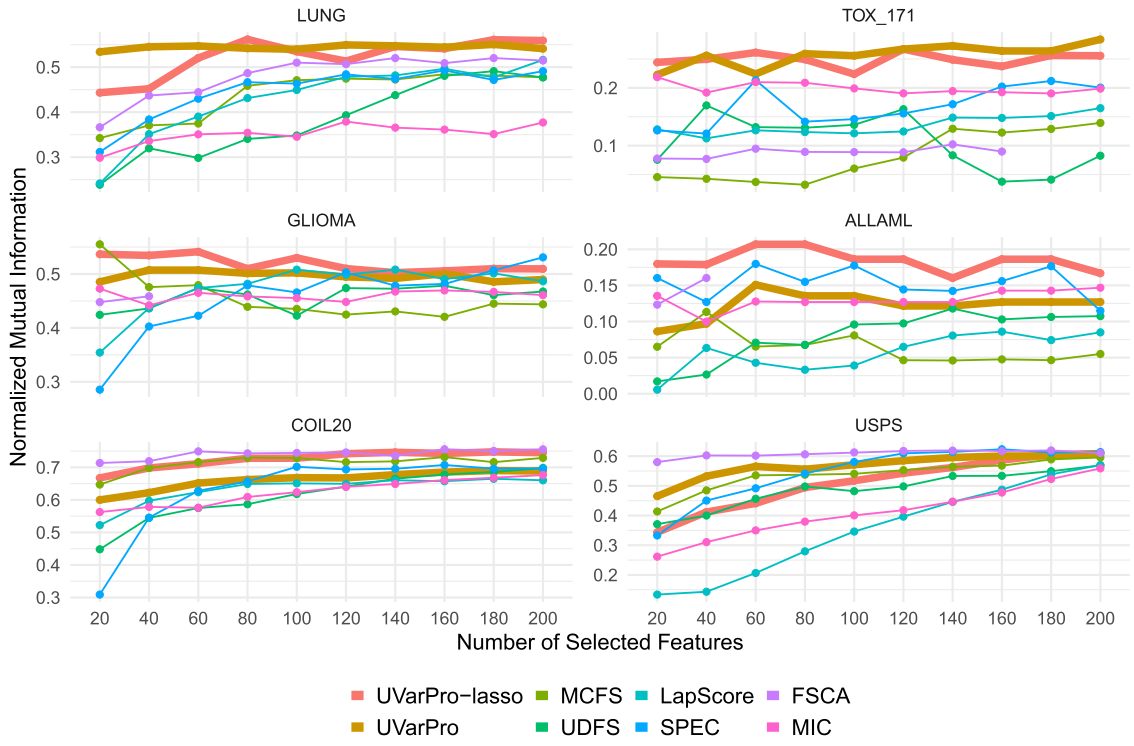

Fig. 11. Clustering performance based on K-means measured by Normalized Mutual Information (NMI) across six high dimensional datasets. Higher NMI indicates better agreement.

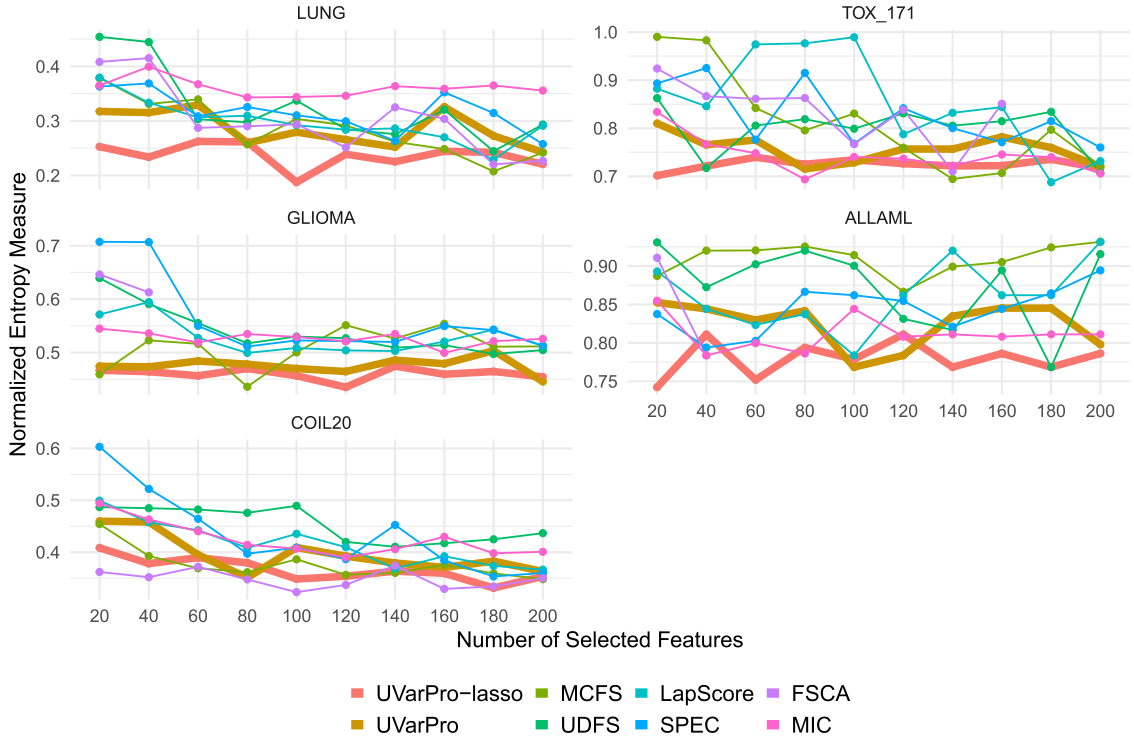

Fig. 12. Clustering performance based on sidClustering with normalized entropy scores. Lower values correspond to better clustering.

sidClustering reported in Fig. 12 are consistent with the NMI based results (note that the USPS dataset is omitted due to computational costs).

Robustness Across Clustering Backends and Metrics. Using UVarPro-lasso and varying the clustering backend and metric, three patterns are observed: (i) on TOX_171 and ALLAML, NMI and ARI remain low for K-means and spectral; GMM improves these metrics but they remain modest, and spectral Silhouette is near or below zero on TOX_171 and near zero on ALLAML, indicating weak separation; (ii) on LUNG and GLIOMA, K-means and GMM exhibit similar performance with higher NMI and ARI than spectral and small positive Silhouette, while spectral

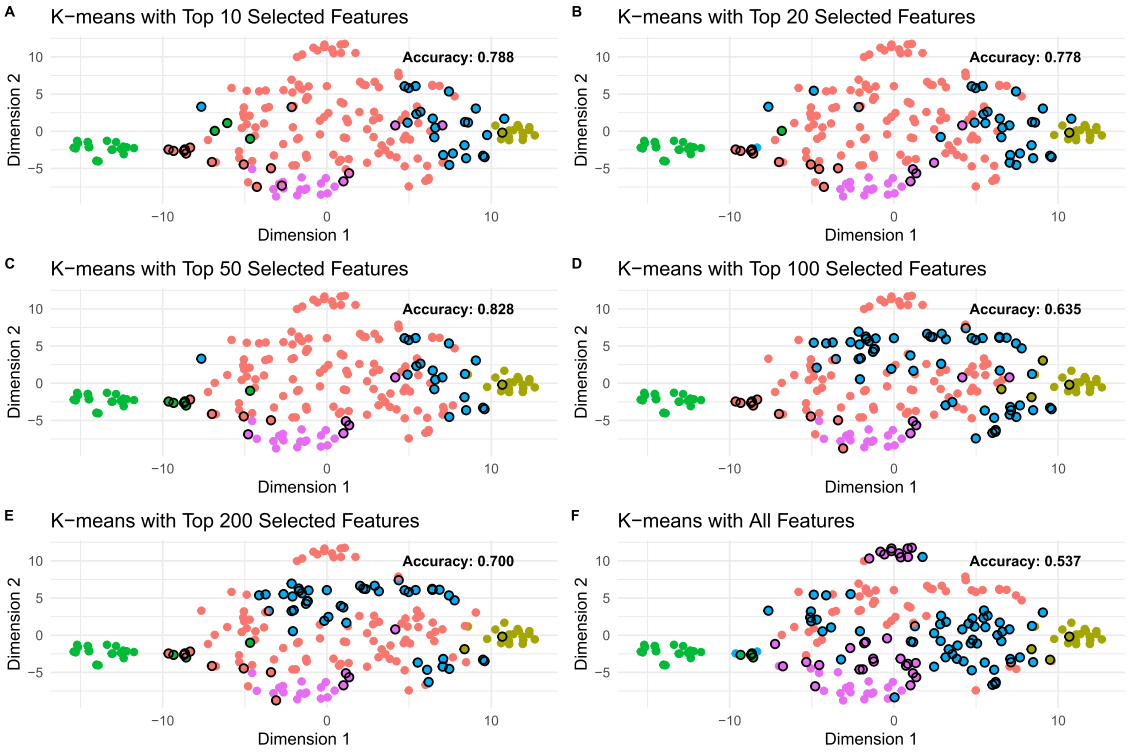

Fig. 13. K-means clustering results across different numbers of selected features by UVarPro. Points highlighted in black identify misclassified cases.

is lower on all three metrics; (iii) on COIL20 and USPS, NMI generally increases with the number of retained features for K-means and GMM. Silhouette is near or below zero for K-means and spectral, consistent with non spherical clusters; GMM yields positive Silhouette at smaller m (for example, USPS 0.511 at Top 10) with diminishing returns as m grows.

Overall Assessment. The across backend results confirm the comparison analysis: the same trends and relative performance observed with K-means (NMI) reappear with spectral and GMM and also when evaluating sidClustering by NMI. Where metrics are low (for example, NMI ceilings and near zero or negative Silhouette), the cause is intrinsic data geometry (weak between cluster separation, overlap among classes, and non spherical structure) and, as m increases, added marginal or redundant features inflate within cluster dispersion and blur boundaries.

6.8. Biological validation: Lung cancer case study

To evaluate performance on real-world, high-dimensional data, we apply our unsupervised feature selection method to gene expression profiles from a widely cited lung cancer study [36]. The goal is to determine whether the method recovers biologically meaningful genes that distinguish known lung cancer subtypes. We assess this in two ways: (i) by measuring the proportion of selected genes previously reported in the biomedical literature as subtype indicators, and (ii) by examining how well the selected genes, used without outcome labels, organize samples into distinct groups. The latter is quantified by subtype separation and by K-means clustering accuracy, compared with results obtained using all genes or principal components.

The dataset contains mRNA expression measurements from 203 lung tissue samples, comprising 186 tumors and 17 normal controls. Tumors span four major subtypes: 139 adenocarcinomas, 21 squamous cell carcinomas, 20 pulmonary carcinoids, and 6 small cell lung carcinomas (SCLC). Following the preprocessing of the original study, a standard deviation filter reduces the initial 12,600 probe sets to the 3,312 most variable genes.

6.8.1. Selected features

Table ST2 (Supplementary Materials) lists the top 20 features selected by the proposed method. Twelve of these genes have been previously reported in the biomedical literature as relevant to lung cancer, including transforming growth factor beta receptor 3, four and a half LIM domains 1, glypican 3, tetranectin, keratin 5, IQ motif-containing GTPase-activating protein 1, AHNAK nucleoprotein, metallothionein 1A, clone 24,651 mRNA, small inducible cytokine A5, glutathione peroxidase 3, and keratin 6 isoform K6E mRNA. Notably, the top five features identified here match those from the original supervised analysis of this dataset [36].

6.8.2. Visualization of clustering results

To evaluate the impact of our unsupervised feature selection method on clustering performance, we visualize the structure of the high-dimensional lung cancer dataset under dimensionality reduction strategies. Given the multi-class complexity of this dataset, visualization offers intuitive insights into the quality of the resulting clusters.

6.8.3. Visualization of clustering results

We next examine how unsupervised feature selection affects clustering by visualizing the lung cancer dataset after dimensionality reduction. Because this dataset contains multiple subtypes, visualization helps convey how well groups are separated. To do so, we use t-distributed Stochastic Neighbor Embedding (t-SNE) [37] to project the data into two dimensions, which highlights local neighborhood structure and is widely used for biological data. Fig. SF2 (Supplementary Materials) shows the t-SNE map with true subtype labels for reference.

To evaluate clustering, we apply K-means to features selected by our method and display the results in Fig. 13 for different numbers of selected variables. For comparison, Fig. 14 shows K-means applied to principal components from PCA. In each plot, cluster colors are overlaid on the baseline t-SNE map, accuracies are reported in the panel titles, and misclassified cases are marked in black.

UVarPro achieves its best accuracy (0.828) with the top 50 features, and consistently outperforms the baseline of using all features. Even

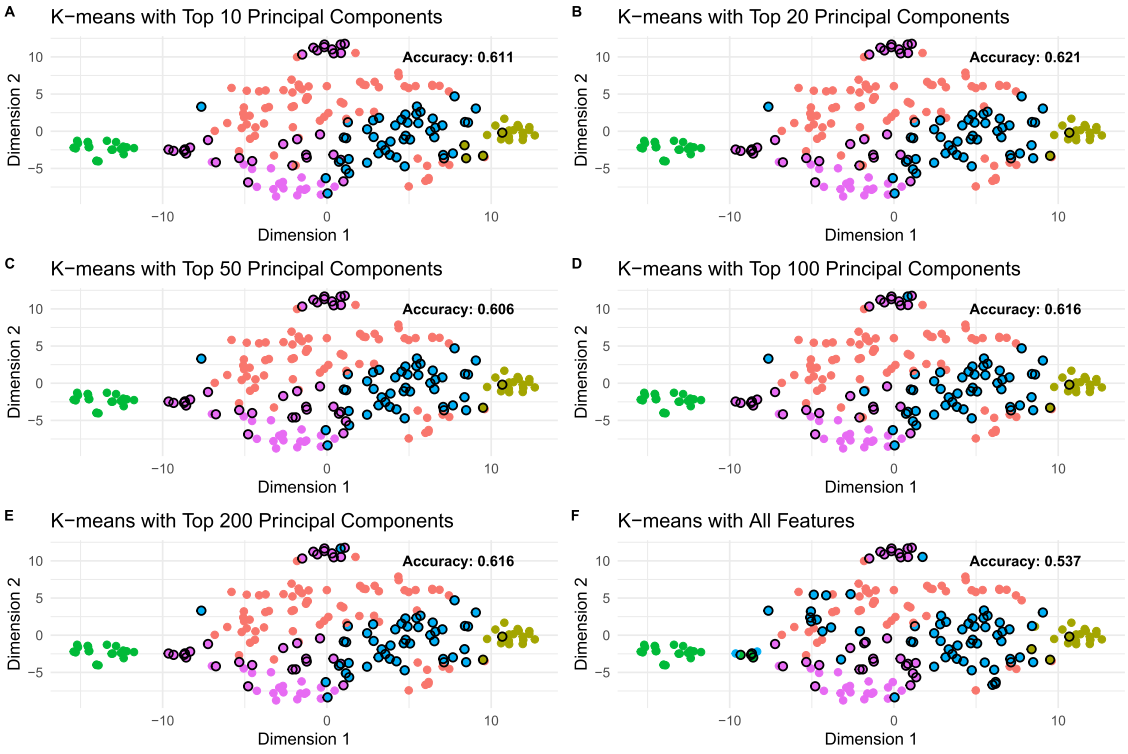

Fig. 14. K-means clustering results across different numbers of selected features by PCA. Points highlighted in black identify misclassified cases.

with only 10 features, the method separates several subtypes, including normal lung, squamous cell carcinomas, and pulmonary carcinoids. In comparison, clustering with all genes or with PCA components fails to clearly distinguish these groups. Accuracy declines for larger sets (e.g., 50 or 200 genes), as weaker or redundant variables dilute the signal and increase K-means sensitivity to initialization. A practical rule is to choose the smallest feature set within 1 % of the best score. For this dataset, that cutoff favors about 20 genes.

7. Discussion

7.1. Summary of contributions

This work introduces a framework for unsupervised variable selection that reformulates the problem as a sequence of proxy classification tasks defined by tree-based rules. Each task is constructed by contrasting membership in a tree region with its released counterpart, producing synthetic two-class labels. Although not tied to an outcome, these labels capture structural properties of the joint feature distribution and allow supervised tools, such as lasso logistic regression, to be used for quantifying variable influence.

A second contribution is the use of multivariate unsupervised trees in an autoencoder-style forest, where the same feature matrix serves as both input and pseudo-response. This design yields high-quality partitions, and the resulting rules define localized problems that reveal how individual features preserve or alter dependency structure. Aggregating selection patterns across many such rules produces a stable, interpretable importance matrix that reflects both marginal and conditional relevance.

The theoretical basis for the approach is a factorization of the joint feature density, which guarantees that contrasts between regions and their released versions correspond to meaningful dependencies. This ensures recovery of signal variables without observing a response. The framework also accommodates non-uniqueness in the signal set, including settings with redundancy or interchangeable features.

7.2. Relation to existing methods

Most existing unsupervised feature selection methods rely on clustering or manifold preservation, evaluating variables by their contribution to partition recovery. UVarPro departs from this paradigm by targeting conditional dependence. The central question is whether a variable improves reconstruction of local structure, even in settings where clusters overlap or are absent. This distinction yields several advantages relative to clustering-based approaches: (i) it provides directional attribution, since the importance matrix records, for each excluded coordinate s, the variables that best account for its local structure; (ii) it enables the use of supervised tools well-suited to high-dimensional data; and (iii) it remains lightweight and parallelizable, supporting efficient computation in practice.

7.3. Empirical performance

Empirical results confirm the method's effectiveness across a broad range of synthetic simulations, including latent variable models, redundant causal structures, and cluster-based designs with correlated noise. Across these settings, the method generally outperforms alternatives based on greedy search, mutual information, graph sparsification, or sparse projections. Additional benchmarking on real high-dimensional datasets supports its strong performance, particularly in low-sample, high-feature settings. Evaluation against supervised gold standards further reinforces these findings. Across classification, regression, and survival contexts, the method identifies more relevant features than strong baselines. In a lung cancer case study, it successfully recovers known subtype indicators and improves clustering quality over standard models.

7.4. Limitations and scalability

Despite its advantages, the current approach faces practical challenges in high-dimensional settings. The autoencoder-style forest is computationally expensive, since the full feature matrix is used on both

sides of the split, making multivariate partitioning costly as p increases. One potential direction is to replace full-response splitting with a `ytry`-like mechanism that samples candidate response variables, potentially guided by adaptive weights that are iteratively updated during training. Another bottleneck is the repeated fitting of lasso models across rules and coordinates. However, these proxy problems are fully independent and thus highly parallelizable. Future work could explore GPU acceleration, multi-threaded solvers, or warm-start strategies to further improve scalability.

7.5. Future extensions

The method produces a full $p \times p$ matrix of pairwise selection scores, yet current implementations use only column-wise summaries to rank variables. The unused structure in the matrix may carry valuable information about higher-order dependencies, which could support structured variable selection or unsupervised interaction discovery.

The framework is also adaptable to more complex data types. For time series, rules could be grown on lag-augmented feature vectors, with releases defined by relaxing the present-time coordinate while incorporating safeguards against temporal leakage. For multimodal data, contributions could be aggregated at the modality level, with penalties that respect block structure. For hierarchical or temporal settings, theoretical results such as Theorem 2 would need to be extended to account for more intricate conditional independence relationships, including lagged or nested dependencies. These extensions will require additional design choices and validation, and are left for future work.

CRediT authorship contribution statement

Lili Zhou: Writing – original draft, Visualization, Validation, Methodology, Formal analysis, Data curation, Conceptualization; **Min Lu:** Supervision, Methodology, Conceptualization; **Hemant Ishwaran:** Writing – review & editing, Writing – original draft, Supervision, Software, Project administration, Methodology, Funding acquisition, Conceptualization.

Data and Code Availability

The code used in this study is publicly available as the R package `varPro`, hosted at https://github.com/kogalur/varPro. All datasets used in the analysis are publicly accessible. The synthetic experiments are described in sufficient detail to allow full reproducibility.

Data Availability

All data is publicly available. Synthetic experiments are reproducible.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Research for the authors was supported by the National Institute Of General Medical Sciences of the National Institutes of Health, Award Number R35 GM139659 and the National Heart, Lung, and Blood Institute of the National Institutes of Health, Award Number R01 HL164405.

Supplementary material

Supplementary material associated with this article can be found in the online version at 10.1016/j.patcog.2025.112727.

References

[1] D. Koller, M. Sahami, Toward optimal feature selection, in: Proceedings of the Thirteenth International Conference on Machine Learning, Morgan Kaufmann, 1996, pp. 284–292.
[2] M. Lu, H. Ishwaran, Model-independent variable selection via the rule-based variable priority, (2024). arXiv:2409.09003
[3] H. Ishwaran, Multivariate Statistics: Classical Foundations and Modern Machine Learning, Chapman and Hall/CRC, Boca Raton, FL, 1st ed., 2025.
[4] X. Zuo, W. Zhang, X. Wang, L. Dang, B. Qiao, Y. Wang, Unsupervised feature selection via maximum relevance and minimum global redundancy, Pattern Recognit. 164 (2025) 111483.
[5] D. Li, H. Chen, Y. Mi, C. Luo, S.-J. Horng, T. Li, Dual space-based fuzzy graphs and orthogonal basis clustering for unsupervised feature selection, Pattern Recognit. 155 (2024) 110683.
[6] C. Wang, J. Wang, Z. Gu, J.-M. Wei, J. Liu, Unsupervised feature selection by learning exponential weights, Pattern Recognit. 148 (2024) 110183. https://doi.org/10.1016/j.patcog.2023.110183
[7] P. Huang, Z. Kong, M. Xie, X. Yang, Robust unsupervised feature selection via data relationship learning, Pattern Recognit. 142 (2023) 109676.
[8] Z. Wang, Q. Li, H. Zhao, F. Nie, Simultaneous local clustering and unsupervised feature selection via strong space constraint, Pattern Recognit. 142 (2023) 109718.
[9] C. Liang, L. Wang, Y. Yang, Y. Wang, Multi-view unsupervised feature selection with tensor robust principal component analysis and consensus graph learning, Pattern Recognit. 141 (2023) 109632. https://doi.org/10.1016/j.patcog.2023.109632
[10] L. Puggini, S. McLoone, Forward selection component analysis: algorithms and applications, IEEE Trans. Pattern Anal. Mach. Intell. 39 (12) (2017) 2395–2408.
[11] F. Zocco, M. Maggipinto, G.A. Susto, S. McLoone, Lazy FSCA for unsupervised variable selection, Eng. Appl. Artif. Intell. 124 (2023) 106624.
[12] M. Minoux, Accelerated greedy algorithms for maximizing submodular set functions, in: Optimization Techniques: Proceedings of the 8th IFIP Conference on Optimization Techniques Würzburg, Springer, 1978, pp. 234–243.
[13] F. Zocco, S. McLoone, Mean squared error vs. frame potential for unsupervised variable selection, in: International Conference on Intelligent Computing for Sustainable Energy and Environment, Springer, 2017, pp. 353–362.
[14] X. He, D. Cai, P. Niyogi, Laplacian score for feature selection, in: Advances in Neural Information Processing Systems, 2005, pp. 507–514.
[15] Z. Zhao, H. Liu, Spectral feature selection for supervised and unsupervised learning, in: Proceedings of the 24th International Conference on Machine Learning, 2007, pp. 1151–1157.
[16] D. Cai, C. Zhang, X. He, Unsupervised feature selection for multi-cluster data, Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2010) 333–342.
[17] Y. Yang, H.T. Shen, Z. Ma, Z. Huang, X. Zhou, L2,1-norm regularized discriminative feature selection for unsupervised learning, IJCAI 11 (2011) 1589–1594.
[18] Z. Li, Y. Yang, J. Liu, X. Zhou, H. Lu, Unsupervised feature selection using nonnegative spectral analysis, in: Proceedings of the AAAI Conference on Artificial Intelligence, 26, 2012, pp. 1026–1032.
[19] J.-S. Wu, J.-T. Yu, J.-Y. Wu, W. Min, W.-S. Zheng, High-order aligned deep complementary and view-Specific similarity graphs for unsupervised multi-View feature selection, Pattern Recognit. 171 (2026) 112047. https://doi.org/10.1016/j.patcog.2025.112047
[20] P. Bühlmann, M. Kalisch, M.H. Maathuis, Variable selection in high-dimensional linear models: partially faithful distributions and the PC-simple algorithm, Biometrika 97 (2) (2010) 261–278.
[21] D.N. Reshef, Y.A. Reshef, H.K. Finucane, S.R. Grossman, G. McVean, P.J. Turnbaugh, E.S. Lander, M. Mitzenmacher, P.C. Sabeti, Detecting novel associations in large data sets, Science 334 (6062) (2011) 1518–1524.
[22] R. Heller, Y. Heller, S. Kaufman, B. Brill, M. Gorfine, Consistent distribution-free K-sample and independence tests for univariate random variables, J. Mach. Learn. Res. 17 (29) (2016) 1–54.
[23] J.-S. Wu, M.-X. Song, W. Min, J.-H. Lai, W.-S. Zheng, Joint adaptive manifold and embedding learning for unsupervised feature selection, Pattern Recognit. 112 (2021) 107742. https://doi.org/10.1016/j.patcog.2020.107742
[24] Y. Zhang, Q. Wang, D.-w. Gong, X.-f. Song, Nonnegative Laplacian embedding guided subspace learning for unsupervised feature selection, Pattern Recognit. 93 (2019) 337–352. https://doi.org/10.1016/j.patcog.2019.04.020
[25] P. Zhu, W. Zhu, Q. Hu, C. Zhang, W. Zuo, Subspace clustering guided unsupervised feature selection, Pattern Recognit. 66 (2017) 364–374. https://doi.org/10.1016/j.patcog.2017.01.016
[26] A.K. Shakya, A. Ramola, Exploration of pixel based and object based change detection techniques by analyzing ALOS PALSAR and LANDSAT data, in: Smart and Sustainable Intelligent Systems, Wiley, 2021, pp. 229–244. https://doi.org/10.1002/9781119752134.ch17
[27] A.K. Shakya, A. Ramola, A. Kandwal, R. Prakash, Comparison of supervised classification techniques with ALOS PALSAR sensor for Roorkee region of Uttarakhand, India, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLII-5 (2018) 693–701. https://doi.org/10.5194/isprs-archives-XLII-5-693-2018
[28] A.K. Shakya, A. Ramola, A. Vidyarthi, Modeling of texture quantification and image classification for change prediction due to COVID lockdown using SkySat and PlanetScope imagery, Model. Earth Syst. Environ. 8 (2) (2022) 2767–2792. https://doi.org/10.1007/s40808-021-01258-6
[29] A.K. Shakya, A. Ramola, S. Singh, A. Vidyarthi, Optimum supervised classification algorithm identification by investigating PlanetScope and SkySat

multispectral satellite data of Covid lockdown, Geosyst. Geoenviron. 2 (2) (2023) 100163. https://doi.org/10.1016/j.geogeo.2022.100163
[30] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[31] H. Ishwaran, M. Lu, Standard errors and confidence intervals for variable importance in random forest regression, classification, and survival, Stat. Med. 38 (4) (2019) 558–582.
[32] F. Tang, H. Ishwaran, Random forest missing data algorithms, Stat. Anal. Data Min. 10 (6) (2017) 363–377.
[33] P. Bühlmann, M. Kalisch, M.H. Maathuis, Variable selection in high-dimensional additive models: the PC-simple algorithm, J. R. Stat. Soc. Ser. B 72 (3) (2010) 349–372.
[34] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learni Res. 7 (2006) 1–30.
[35] A. Mantero, H. Ishwaran, Unsupervised random forests, Stat. Anal. Data Min. ASA Data Sci. J. 14 (2) (2021) 144–167.
[36] A. Bhattacharjee, W.G. Richards, J. Staunton, C. Li, S. Monti, P. Vasa, C. Ladd, J. Beheshti, R. Bueno, M. Gillette, et al., Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses, Proc. Natl. Acad. Sci. 98 (24) (2001) 13790–13795.
[37] G.E. Hinton, S. Roweis, Stochastic neighbor embedding, Adv. Neural Inf. Process. Syst. 15 (2002).