
Distributionally Robust Feature Selection

Maitreyi Swaroop
Machine Learning Department
Carnegie Mellon University
mswaroop@andrew.cmu.edu

Tamar Krishnamurti
Division of General Internal Medicine
University of Pittsburgh
tamark@pitt.edu

Bryan Wilder
Machine Learning Department
Carnegie Mellon University
bwilder@andrew.cmu.edu

Abstract

We study the problem of selecting limited features to observe such that models trained on them can perform well simultaneously across multiple subpopulations. This problem has applications in settings where collecting each feature is costly, e.g. requiring adding survey questions or physical sensors, and we must be able to use the selected features to create high-quality downstream models for different populations. Our method frames the problem as a continuous relaxation of traditional variable selection using a noising mechanism, without requiring back-propagation through model training processes. By optimizing over the variance of a Bayes-optimal predictor, we develop a model-agnostic framework that balances overall performance of downstream prediction across populations. We validate our approach through experiments on both synthetic datasets and real-world data.¹

1 Introduction

Many real-world applications impose significant constraints on data collection for machine learning models. These constraints often stem from factors such as limited budgets, privacy concerns, or the operational costs associated with acquiring each data point. For instance, in healthcare, a hospital system aiming to implement a predictive screener across diverse patient populations must often contend with limitations on the number of questions permissible due to patient burden, time constraints, and regulatory considerations. In such scenarios, the ability to identify a minimal yet highly informative set of features—*feature selection*—becomes paramount. Oftentimes, we might collect pilot data on a large number of features in order to make the operational decision of which to collect in deployment. For example, medical screeners are often developed with a larger number of questions that are then cut down to a smaller number for general-population usage; e.g., the PHQ [Spitzer et al., 1999] is a 26-item mental health survey that is typically reduced to standard 9 or 2-question versions in practice [Kroenke et al., 2001, 2003].

The challenge intensifies when the selected features must ensure reliable model performance not just on average, but across varied and potentially shifting underlying data distributions. This necessitates a *distributionally robust* approach to feature selection, ensuring that no particular subpopulation is inadvertently neglected by models trained on the selected features. Furthermore, in many practical settings, the specific downstream inference model that will ultimately use the collected data may not be known at the time of feature selection, or multiple different models might be employed. Therefore,

¹Code for implementing our method is available here (linked).

an ideal feature selection methodology should be *model-agnostic*, providing a core set of features that are broadly useful without being tied to the idiosyncrasies of a particular predictive algorithm.

This paper addresses the problem of selecting a limited subset of features from a larger pool, using historical data, to facilitate accurate and robust predictions across diverse populations, all while adhering to a collection budget. We aim to identify features that are robust to distributional shifts, ensuring that the model performs well simultaneously across a variety of specified subpopulations (e.g., institutional settings, clinics, or demographic groups).

To our knowledge, this problem has not been previously studied in the literature: feature selection methods focus on a single distribution of interest, while distributionally robust optimization (DRO) methods attempt to find a single model that performs well on all populations, instead of selecting a limited number of features that can then be used to train high-performing models for each downstream population. The intersection of the two problems complicates both. On the one hand, we require an approach to feature selection that is model-agnostic and naturally extends across multiple populations. On the other hand, standard DRO methods do not apply because we have a discrete optimization problem of selecting features instead of directly optimizing over a single model. Our approach formulates this task by introducing a continuous relaxation of the discrete feature selection problem which injects synthetic noise into the observation of each covariate. We then develop an analytical simplification of the relaxation which eventually allows us to solve it with standard stochastic gradient descent-style methods.

Our main contributions are as follows:

1. We propose a novel, model-agnostic method for distributionally robust feature selection. This method identifies a subset of features that minimizes the maximum expected loss (or error) across potential data distributions, without requiring assumptions about the specific architecture or differentiability of the downstream predictive model.
2. We reframe the combinatorial problem of feature selection into a continuous optimization framework. This is achieved by introducing parameters that govern a noise injection process, effectively creating a differentiable measure of each feature's utility and permitting tractable, gradient-based optimization for selection.
3. We demonstrate the efficacy of our approach through experiments on both synthetic and real-world datasets, comparing its performance against naive selection strategies and existing distributionally robust optimization (DRO) baselines adapted for feature selection.

1.1 Related work

Our work lies in the intersection of distributionally robust optimization and feature selection. While there is a rich body of work that focuses on each topic individually, to our knowledge no previous work studies the intersection – selecting a set of *features* that will allow a high-performing model to be trained for each subpopulation, as opposed to training a single model that works well everywhere.

Feature selection: Classical feature selection methods primarily fall into two categories: combinatorial optimization approaches [Guyon and Elisseeff, 2003, Kohavi and John, 1997], and embedded methods [Breiman, 2001, Tibshirani, 1996]. In the combinatorial optimization vein, a long line of work develops greedy strategies or forward/backward selection heuristics for variable selection, historically focused on linear models. Even for linear models, selecting the best set of at most k features is known to be NP-hard [Natarajan, 1995]. Das and Kempe [2011], provided theoretical approximation guarantees for greedy selection in linear models via a connection to submodularity. Khanna et al. [2017] provide faster algorithms for the same problem. However, there is no direct path to extend these methods beyond linear models, or to incorporate robustness across multiple distributions as a goal. Lasso regression [Tibshirani, 1996] is a classical method frequently used for feature selection, which leverages an ℓ_1 penalty to induce sparsity. However, the Lasso formulation does not account for robustness to distributional shifts and again bakes in an assumption of linearity. Zhao and Yu [2006] highlight potential pitfalls of Lasso-based feature selection, showing that it may not consistently select the correct variables under certain correlation structures. Recent advances in embedded methods have explored heterogeneous feature selection, though these address different problem settings from our work. Yang et al. [2022] propose locally sparse neural networks (LSPIN) that learn sample-specific feature subsets, allowing different features for different samples. Similarly, Svirsky and Lindenbaum

[2024] introduce interpretable deep clustering (IDC) that performs cluster-level feature selection. While these methods advance feature selection for heterogeneous populations, they fundamentally differ from our approach: our method selects a single global feature subset that must perform well universally across all known population groups, a constraint motivated by practical deployment scenarios where systems require a fixed set of features for all users. Previous work has also proposed the use of random noise injection for feature selection in single population settings [Grandvalet, 2000], as well as through Bayesian relevance estimation methods [Neal, 1996, Tipping, 2001]. However, tunable-noise-based variable selection has seen limited adoption since it was originally proposed, in contrast to the large body of work on noise as a form of regularization [Bishop, 1995]. In this work, we revisit noise-based relaxations in the distributionally robust setting and show how the optimization problem can be reformulated in ways that create significant, previously-unrecognized advantages. Crucially, our approach separates variable selection from predictive model fitting, making it agnostic to the downstream model and eliminating the need to differentiate through model training – which may be infeasible for frequently used models like decision trees or random forests.

Distributionally Robust Optimization (DRO): DRO provides a principled framework to account for worst-case model performance under distribution shifts, contrasting with traditional ML methods that optimize only for average performance. Duchi and Namkoong [2020] formalize this approach by providing finite-sample minimax bounds for uniform model performance across test distributions. Group DRO, which focuses on worst-case performance across predefined population groups, is particularly relevant to our work. Sagawa et al. [2019] introduce a group DRO approach for neural networks that explicitly optimizes for the worst-case loss over known population groups, demonstrating improved performance on underrepresented groups compared to standard empirical risk minimization. Similarly, Hashimoto et al. [2018] demonstrate that minimizing the worst-case risk across groups can prevent representation disparity in sequential learning settings.

2 Problem formulation

We assume our input $(X, Y) \sim P$ to be covariates $X \in \mathbb{R}^m$ and outcomes Y . We wish to select a subset of $k < m$ variables from X that are most informative for predicting Y , while being robust to shifts in the underlying distribution P . Let $\mathcal{I}$ denote a set of indices corresponding to the selected variables, with $|\mathcal{I}| = k$. The selected sub-vector is $X_{\mathcal{I}} \in \mathbb{R}^{|\mathcal{I}|}$. Informally, we wish to solve the problem:

$$\min_{|\mathcal{I}|=k} \max_{P_i \in \mathcal{P}} \mathbb{E}[\text{Loss of a model trained on } (\tilde{X}_{\mathcal{I}}, Y) \sim P_i]$$

where $\mathcal{P}$ represents the set of distributions we wish to ensure robust performance on. The above can be equivalently expressed as the problem of choosing a binary mask $\alpha \in \{0, 1\}^m$, $\sum_i \alpha_i = k$ where the model observes $\tilde{X} = \alpha \odot X$. To formalize the problem, let $\mathcal{L}$ be a loss function (we will work with the mean squared error throughout), and

$$M_{i,\alpha} = \arg \min_{f \in \mathcal{F}} \mathbb{E}_{X,Y \sim P_i} [\mathcal{L}(Y, f(\alpha \odot X))]$$

be the risk minimizer for population i over some model class $\mathcal{F}$ when the covariates are masked by α . Accordingly, we can formalize our problem as

$$\min_{\alpha} \max_{P_i \in \mathcal{P}} \mathbb{E}_{X,Y \sim P_i} [\mathcal{L}(Y, M_{i,\alpha}(\alpha \odot X))] \text{ s.t. } \|\alpha\|_0 = k. \quad (1)$$

In order to solve this problem, we have access to samples $\{(X_i^j, Y_i^j)\}_{j=1}^{n_i}$ drawn iid for each population $P_i \in \mathcal{P}$.

3 Methods

There are two main challenges to solving this problem. First, the optimization over α is discrete, since α is binary – even for linear models and a single population, this is NP-hard [Natarajan, 1995]. We address this by introducing a continuous relaxation. Second, the population-level minimizer $M_{i,\alpha}$ depends on α in a nontrivial manner, as the solution to a risk minimization problem on the covariates included by α . In practice, we will only have finite data available to solve an empirical version of the risk minimization problem. Moreover, the model training process may induce complex

dependencies between the decision variable α and the objective function, which is unlikely to have a closed form for general model families $\mathcal{F}$. To circumvent these issues, our method targets the loss of the *Bayes-optimal* predictor for each distribution and show that this surprisingly allows us to arrive at a significantly more computationally tractable optimization problem.

3.1 Continuous relaxation

While ℓ_0 -constrained problems are typically hard, we draw inspiration from the Lasso and relax to an ℓ_1 constraint that restores convexity of the feasible set while still encouraging sparsity. In a continuous relaxation, $\alpha \in [0, 1]^m$ now gives the *degree* to which a variable is included instead of a binary decision. However, naively scaling inputs as $\alpha \odot X$ allows flexible predictors to trivially undo the scaling. For example, if the model involves a linear transformation, it could internally learn a coefficient w_i/α_i for the input $\alpha_i X_i$. A deterministic scaling does not actually remove any information about the covariate unless $\alpha_i = 0$ exactly.

To avoid this issue, we introduce an alternative continuous relaxation that incorporates a *stochastic* component controlled by α ; effectively, α_i will control the amount of noise added to the observation of X_i . Formally, let $\alpha = (\alpha_1, \dots, \alpha_m) \in \mathbb{R}_{\geq 0}^m$ be a vector of parameters controlling the degradation level for each covariate. We define the observed (noised) variable $S(\alpha)$ as a random variable whose i -th component $S_i(\alpha)$ distributed as

$$S_i(\alpha)|X \sim \mathcal{N}(X_i, \alpha_i),$$

with the random variables independent across i conditionally on X . Equivalently, $S(\alpha) = X + \epsilon(\alpha)$, where $\epsilon(\alpha) \sim \mathcal{N}(0, \Sigma_\alpha)$ with $\Sigma_\alpha = \text{diag}(\alpha_1, \dots, \alpha_m)$. Here, $\alpha_i = 0$ implies $S_i = X_i$ (no degradation), while $\alpha_i \rightarrow \infty$ implies S_i contains no information about X_i . Our goal is to find an α vector that has small values for relevant features and large values for irrelevant ones, while maintaining predictive performance under distribution shifts. Formally, our objective becomes

$$\min_{\alpha} \max_{P_i \in \mathcal{P}} \mathbb{E}_{S(\alpha), Y \sim P_i} [\mathcal{L}(Y, \tilde{M}_{i,\alpha}(S(\alpha)))] + \lambda \text{Reg}(\alpha) \quad (2)$$

$$\tilde{M}_{i,\alpha} = \arg \min_{f \in \mathcal{F}} \mathbb{E}_{S(\alpha), Y \sim P_i} [\mathcal{L}(Y, f(S(\alpha)))]. \quad (3)$$

so that we optimize the performance of the risk-minimizing models that observe the noisy version of the covariates. λ is the regularization parameter which controls the sparsity of the solution, and can be varied to obtain a solution with the desired cardinality. We note that the regularization term here is in contrast to standard ℓ_1 regularization, which typically promotes sparsity by shrinking irrelevant parameters toward zero, eg. we may choose $\text{Reg}(\alpha) = 1/\|\alpha\|_1$.

3.2 Solving the relaxation

Directly solving the relaxation in Equation (3) is nontrivial. Each choice of α leads to a different set of models $\tilde{M}_{i,\alpha}$. Moreover, the structure of this mapping may be highly complex, mediated as it is by the inner minimization problem defining $\tilde{M}_{i,\alpha}$. One strategy employed throughout the machine learning literature to solve problems with an inner optimization loop is to differentiate through the solution to the inner problem in order to optimize the outer objective via gradient descent [Amos and Kolter, 2017, Finn et al., 2017, Agrawal et al., 2019, Liu et al., 2018]. Intuitively, this corresponds to computing a gradient $\nabla_{\alpha} \tilde{M}_{i,\alpha}$ which captures how changes to α in turn change the fitted model. Perhaps the closest analogy to our setting is model-agnostic meta-learning (MAML) [Finn et al., 2017] which solves meta-learning problems by differentiating through the training loops of models for individual tasks.

While it might be possible to adapt such a strategy to our setting, it would incur three key disadvantages.

1. **Computational Expense:** Retraining a potentially complex model M for every adjustment to α and for every considered P_i during the optimization process is often prohibitively costly. Backpropagating through the model fitting process in every iteration is similarly costly.
2. **Optimization Instability:** Backpropagating gradients through the iterative training procedure of M with respect to α can be numerically unstable, suffer from vanishing/exploding gradients, or converge poorly, especially for deep or non-convex models.

3. **Specificity to a single model class:** Such a formulation would necessarily optimize for the performance of the particular class of models used to instantiate the inner loop. In order to maintain differentiability, this would likely have to be neural networks, even if other models (e.g., random forests) might be preferred for the tabular settings common in applications like medical risk prediction.

To circumvent these difficulties, we shift our focus from the performance of a specific, explicitly trained model M to the performance of a Bayes-optimal optimal predictor, which represents the best possible performance achievable given the selected features. We believe this to be a good target for optimization for multiple reasons. Firstly, it represents a model-family agnostic target which will be more closely approached if a practitioner makes good choices for the modeling approach within any specific setting and distribution. Secondly, in many settings, after feature selection, we might be able to collect a substantial amount of new data corresponding to the chosen covariates. E.g., consider a health system that decides on new survey items to gather based on a pilot study, and can then observe new data from the routine deployment of the selected questions. In such a setting, the Bayes-optimal loss may be a better proxy for the long-term performance of the system. Third, starting from the perspective of the Bayes-optimal predictor will allow us to further simplify the objective and arrive at a significantly more computationally tractable approach. In the following sections, we formalize this approach and show how it leads to a simple closed-form objective.

Population-level objective We now express the population-level formulation of this problem using the Bayes-optimal predictor for Y given $S(\alpha)$, denoted by $f^*(S(\alpha)) = \mathbb{E}[Y|S(\alpha)]$. With respect to the MSE, the expected loss of this optimal predictor is the conditional variance of Y given $S(\alpha)$ (since the bias term is 0):

$$\mathbb{E}_{(S(\alpha), Y) \sim P_i} [(Y - \mathbb{E}[Y|S(\alpha)])^2] = \mathbb{E}_{S(\alpha) \sim P_i} [\mathbb{V}[Y|S(\alpha)]]$$

Applying the law of total variance and dropping terms constant in α leads to an equivalent optimization problem that depends only on the conditional variance of $\mathbb{E}[Y | X]$ given $S(\alpha)$. This can be formalized as follows.

Theorem 1 (Population-Level Objective). *Under the noise-based relaxation $S(\alpha) = X + \epsilon(\alpha)$, $\epsilon(\alpha) \sim \mathcal{N}(0, \text{diag}(\alpha))$, the distributionally robust feature selection problem is equivalent to*

$$\min_{\alpha} \max_{P_i \in \mathcal{P}} -\mathbb{E}_{S(\alpha) \sim P_i} [\mathbb{E}_{X \sim P_i} [\mu_i(X) | S(\alpha)]^2] + \lambda \text{Reg}(\alpha),$$

where $\mu_i(X) = \mathbb{E}_{P_i}[Y | X]$.

The proof of Theorem 1 is provided in Appendix A.1.

3.3 Empirical estimation and kernel form of objective

So far, we have dealt only with population-level quantities. Next, we must develop a strategy to estimate these using the sampled data that we observe. Given samples $\{(X_i^j, Y_i^j)\}$ from each population P_i , let $\hat{\mu}_i(X)$ be an estimator of $\mu_i(X) = \mathbb{E}[Y|X]$ trained on these samples. The empirical form of the objective from Theorem 1 is

$$\min_{\alpha} \max_{P_i \in \mathcal{P}} -\hat{\mathbb{E}}_{S(\alpha) \sim P_i} [\hat{\mathbb{E}}[\hat{\mu}_i(X) | S(\alpha)]^2] + \lambda \text{Reg}(\alpha), \quad (4)$$

where the expectation $\hat{\mathbb{E}}[\hat{\mu}_i(X) | S(\alpha)]^2$ is taken with respect to the empirical distribution $\hat{P}_i(X) = \frac{1}{n_i} \sum_{j=1}^{n_i} \delta_{X_i^j}(X)$. First, we propose to estimate $\mu_i(X)$ simply by fitting a machine learning model to the samples $\{(X_i^j, Y_i^j)\}$ from population i . This can be done just once, using an arbitrary model well-suited to the application domain. There will be no need to differentiate through the training process, or refit the model during training. To make the empirical objective in Equation (4) computationally tractable, we express the conditional expectation $\hat{\mathbb{E}}[\hat{\mu}_i(X) | S(\alpha)]$ in closed form using Bayes' theorem under the Gaussian noise model. This yields a kernel-weighted representation of $\mu_i(X)$, where each observed sample contributes according to its likelihood under the noisy observation $S(\alpha)$.

Theorem 2 (Kernel Form Equivalence). *The empirical expectation*

$$\hat{\mathbb{E}}_{S(\alpha)} [\hat{\mathbb{E}}[\hat{\mu}_i(X) | S(\alpha)]^2]$$

is equivalent to

$$\widehat{\mathbb{E}}_{S(\alpha)} \left[\left(\sum_{j=1}^{n_i} w_i^j(S, \alpha) \hat{\mu}_i(X_i^j) \right)^2 \right]$$

where the weights are

$$w_i^j(S, \alpha) = \frac{\exp \left(-\frac{1}{2} \left(X_i^j - S(\alpha) \right)^T \text{diag}(\alpha)^{-1} \left(X_i^j - S(\alpha) \right) \right)}{\sum_{k=1}^{n_i} \exp \left(-\frac{1}{2} \left(X_i^k - S(\alpha) \right)^T \text{diag}(\alpha)^{-1} \left(X_i^k - S(\alpha) \right) \right)}$$

The proof for Theorem 2 is provided in Appendix A.2. This equivalence shows that the conditional expectation can be estimated via Gaussian kernel smoothing, where the bandwidth of the kernel is determined by α . In summary, this kernel-based formulation provides an explicit, differentiable link between the feature-noise parameters α and the resulting predictive uncertainty. It allows us to efficiently approximate the objective using empirical samples, enabling the end-to-end optimization procedure described in the following section.

3.4 Algorithmic approach

The final objective function takes an expectation of this over the random draw of S itself. Accordingly, the final statement of the optimization problem we wish to solve is

$$\min_{\alpha} \max_{P_i \in \mathcal{P}} -\mathbb{E}_{S(\alpha)} \left[\left(\sum_{j=1}^{n_i} w_i^j(S, \alpha) \mu_i(X_i^j) \right)^2 \right].$$

We implement gradient descent algorithm for this problem. We draw samples of S to approximate the outer objective function. Specifically, for b Monte Carlo samples $S^1 \dots S^b$, we obtain the objective

$$-\frac{1}{b} \sum_{\ell=1}^b \left(\sum_{j=1}^{n_i} w_i^j(S^\ell, \alpha) \mu_i(X_i^j) \right)^2.$$

We take a gradient descent step with respect to either the maximum loss over all the populations, or the softmax over the population losses, controlled by the temperature parameter β . Since the distribution of $S(\alpha)$ itself depends on α , we construct each sample via the reparameterization trick: $S = X + \sqrt{\alpha} \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$. This ensures that gradients can flow not only through the kernel weights $w_i^j(S, \alpha)$, but also through the sampled values S themselves, enabling end-to-end differentiation of the entire objective. This can now be implemented entirely with standard autodifferentiation software since w is a closed-form, differentiable function of α . All that is required is to fit a model $\mu_i(X)$ once at the start of the process for each population; it can then be reused at each iteration unchanged and we never need to differentiate through the model training process. In practice, we further improve computational efficiency by taking the inner sum only over the k -nearest neighbors of $S^{(\ell)}$ in the set $\{X_i^j\}_{j=1}^{n_i}$ since the conditional distribution of X given S typically places a negligible mass outside this set.

The algorithm concludes by selecting the k features with the smallest optimized noise parameters α^* , as these represent the most informative variables for maintaining prediction performance across all populations. We summarize our complete proposed method in Algorithm 1 in Appendix B, and provide its computational complexity.

4 Experiments

We conduct experiments on both synthetic and real-world datasets. We evaluate our proposed method against baseline models trained on a pooled dataset comprising all populations. Specifically, we use Lasso regression (linear regression for real targets, and logistic regression for categorical) and XGBoost [Chen and Guestrin, 2016] regression/classification as baselines. For Lasso, feature

selection is based on the largest coefficients under the regularization path, while for XGBoost, we rely on internal feature importance scores. We also implement distributionally robust (DRO) variants of both models. Further, we include an embedded baseline (Embedded MLP), which uses a multi-layer-perceptron (MLP) with a learnable feature mask trained via DRO. A comprehensive description of all baseline methods can be found in Appendix C. For the downstream prediction task, we train a model on the selected features. The downstream model training is carried out independently for each feature selection method. We implemented our method using the `PyTorch` [Paszke et al., 2019] library, while for the downstream models, we use the `scikit-learn` [Pedregosa et al., 2011] library. All baselines plus our method share the pipeline for downstream models, isolating the impact of the feature selections they output as opposed to predictive performance of models that they use en route.

Data processing We split each dataset into three parts – a *feature-selection-dataset*, *downstream-model-training-dataset* and a *downstream-model-test-dataset*. We first do a 60 : 40 split of each population to obtain the *feature-selection-dataset* and the downstream model training and evaluation datasets. The latter is split 80 : 20 for downstream-model training and evaluation respectively.

Evaluation Metrics In keeping with our motivation to improve downstream prediction accuracy, we first perform feature selection using our method and the baseline methods. We then train the downstream model independently on the *downstream-model-training-dataset*, using the features selected by each method. We then evaluate performance of the fitted downstream models (random forest and MLP) on the *downstream-model-test-dataset* using the mean-squared-error (MSE) as the primary metric for regression datasets, and Log Loss on the classification dataset. For the real datasets, we also include R^2 -score for regression and prediction accuracy for classification tasks. The tabulated metrics are provided in Appendix D.

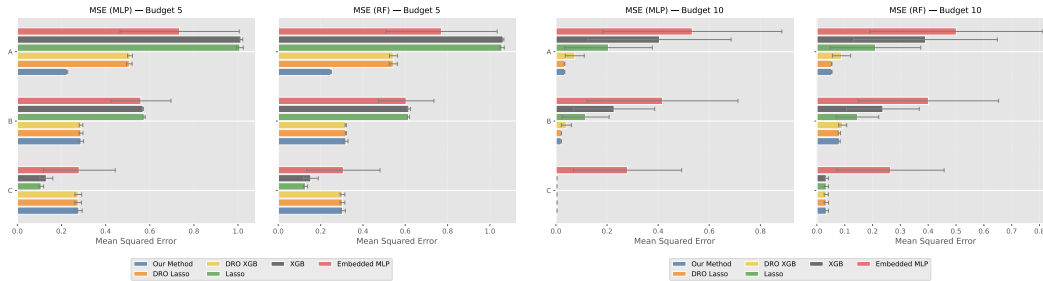
4.1 Synthetic Data Experiments

To systematically evaluate the behavior of our method under controlled settings, we conduct a series of synthetic experiments that vary in functional form, dimensionality, and population heterogeneity. The first experiment considers a purely linear data-generating process with population-specific coefficients to study the effect of structured covariate shifts. A higher-dimensional variant of this experiment is provided in Appendix E.1. The second experiment introduces nonlinear relationships and heterogeneous noise distributions across populations, allowing us to assess robustness to more complex, mixed effects. Finally, an additional experiment using a sparse linear prediction model for targets is included in Appendix E.2. For the downstream prediction models, we use a random forest and an MLP– both models use the standard `scikit-learn` implementations for fitting to the output. The random forest is an ensemble of 100 decision trees, while the MLP has a single hidden layer with 100 neurons, trained for a maximum of 1000 epochs with early stopping based on validation loss convergence (handled internally by `scikit-learn`).

Synthetic dataset 1: Linear model This synthetic dataset comprises three populations A, B, C , with 15 features each. The populations have the following proportions in the dataset, and (linear) outcome functions:

$$\begin{aligned} A \text{ (40\%)} \quad Y &= 8X_0 + 6X_1 - 4X_2 + 3X_3 + 2X_4 + \epsilon \\ B \text{ (35\%)} \quad Y &= -8X_0 - 6X_1 + 4X_2 - 3X_3 - 2X_4 + 8X_5 + 6X_6 + \epsilon \\ C \text{ (25\%)} \quad Y &= 10X_7 + 8X_8 + 6X_9 - 5X_{10} + \epsilon \\ \text{Noise} \quad \epsilon &\sim \mathcal{N}(0, 0.1^2) \end{aligned}$$

We set the budget for feature selection to 5 and, for comparison, also include the results for an increased budget of 10. Figure 1 shows the performance of the downstream prediction models on the task of predicting Y for each population, using the features selected using our method and the baselines. The complete metrics table can be found in Appendix D, Table 1. More details on the implementation and hyperparameters used are provided in Appendix D.1. We also include an additional experiment with this data set in which the number of features increases to 50, the results of which can be found in Appendix E.1.



(a) **Budget = 5:** Our method consistently achieves low error across all populations, performing on par with DRO-XGBoost and DRO-Lasso, even outperforming both methods in population *A*.

(b) **Budget = 10:** Overall performance improves predictably upon increasing the feature budget. Our method maintains lower MSE compared to other methods, including DRO-Lasso.

Figure 1: Performance comparison across populations on synthetic dataset 1 using different feature selection methods. In each subplot, left: Mean Squared Error of downstream MLP model; right: Mean Squared Error of downstream random forest model. The relative performance of feature selection methods is consistent across the choice of downstream prediction model.

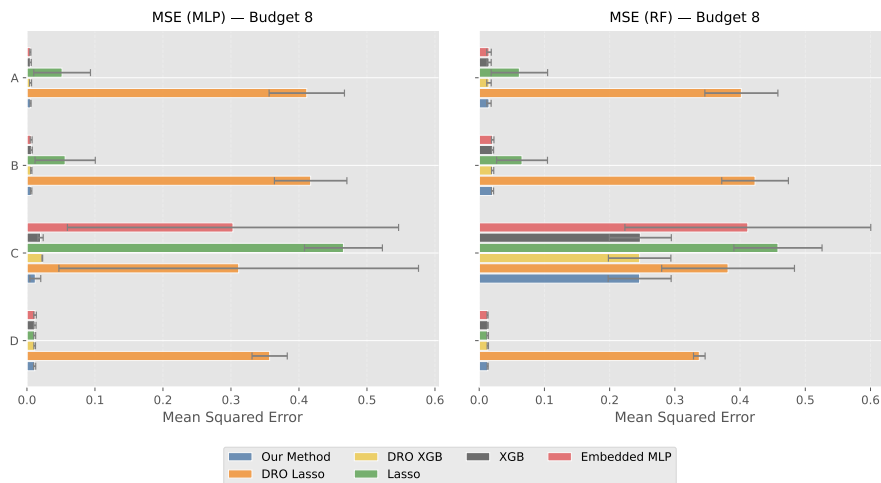


Figure 2: Performance comparison across populations on synthetic dataset 2. Left: Mean Squared Error of downstream MLP model; right: Mean Squared Error of downstream random forest (RF) model. Our method consistently achieves low error across populations and budgets, outperforming or matching DRO-based baselines.

Synthetic experiment 2: Linear model Here we have four populations, *A*, *B*, *C*, *D*, with 50 features each. The populations have the following proportions in the dataset, and outcome functions:

$$\begin{aligned}
 \text{A (40\%)} \quad Y &= 4X_0 + 3X_1 + X_2^2 + \epsilon_A \\
 \text{B (35\%)} \quad Y &= 4X_0 + 3X_1 + X_2^2 + \epsilon_B \\
 \text{C (25\%)} \quad Y &= 2X_0 + 3X_5X_6 + 4\sin(2X_7) + \epsilon_C \\
 \text{D (15\%)} \quad Y &= 3X_0 + 2X_1 + \epsilon_D
 \end{aligned}$$

with heterogeneous noise across populations: Population *A* experiences reduced noise with $\epsilon_A \sim \mathcal{N}(0, 0.05^2)$, while Population *B* exhibits heteroscedasticity where $\epsilon_B = \sigma(X_3, X_4) \cdot \eta \cdot 0.1$, with $\sigma = \exp(0.5X_3 + 0.3X_4)$ and $\eta \sim \mathcal{N}(0, 1)$. Population *C* follows a standard noise model $\epsilon_C \sim \mathcal{N}(0, 0.1^2)$, and Population *D* is characterized by heavy-tailed noise drawn from a scaled *t*-distribution: $\epsilon_D \sim t_3 \cdot 0.2$.

We set the budget for feature selection to 8. Figure 2 shows the performance of the downstream prediction models on the task of predicting *Y* for each population, using the features selected using

our method and the baselines. The complete metrics table can be found in Appendix D, Table 2. More details on the implementation and hyperparameters used are provided in Appendix D.2.

Results In *synthetic experiment 1*, although the generative process is linear and variables X_0 to X_4 have strong effects in both populations A and B , the signs of their coefficients are reversed between populations. This reduces the effectiveness of LASSO, which tends to select features based on average effects across all data. As a result, vanilla LASSO achieves its best performance on population C , as features relevant for C have no conflicting coefficients, but performs poorly on A and B , in spite of the linear setting. Vanilla XGBoost shows a similar trend. Our method outperforms most baselines, and has a balanced performance across populations, not performing inordinately well in a certain population at the cost of others, unlike vanilla Lasso and XGBoost. For budget=10, our method is comparable with the best performing baselines, namely DRO XGB and DRO Lasso. The Embedded MLP baseline performs poorly even for the higher budget.

In *synthetic experiment 2*, the nonlinear data setting puts both vanilla Lasso and its DRO variant at a disadvantage, as seen in Figure 2, with DRO Lasso having the worst performance. The Embedded MLP baseline also performs poorly, being imbalanced in favor of other populations at the cost of population C , similar to Lasso. Our method is again comparable with the best performing baselines of XGB (DRO and vanilla), for both budgets.

It should also be noted that our method has consistently low variance across both the synthetic experiments, the exact values of which may be found in Tables 1 and 2. The relative performance of feature selection methods is consistent across the choice of downstream prediction model.

4.2 Real-datasets

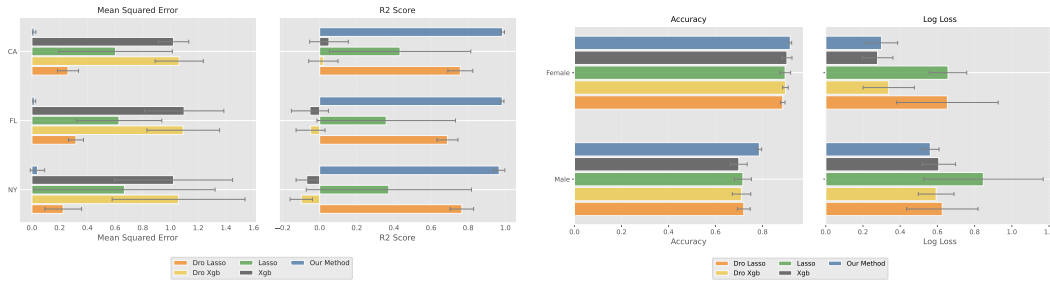
UCI Adult Income Dataset [Becker and Kohavi, 1996] We use the UCI Adult Income dataset to predict income across different demographic groups, where each age group represents a distinct population. The dataset contains census income data with 14 features including age, education, occupation, and demographic information. The target variable is binary, indicating whether an individual's income exceeds \$50K per year. After combining the original train/test splits ($\approx 48,000$ samples), we subsample 10% of the data for faster computation. Populations are defined by sex (Male/Female). We drop any features used to define the population group. After one-hot-encoding, we have 44 features, of which we select 5 (i.e. we set our budget to 5). Since the task is classification, a Random Forest Classifier is trained on selected features. Training is performed on each population separately, using each set of selected features. For further details about hyperparameter choices, please see Appendix D.4.

American Community Survey (ACS) Dataset [U.S. Census Bureau, 2018] We use person-level ACS Public Use Microdata Sample (PUMS) data for the year 2018 to predict household income across state populations. We focus on three populations, namely California (CA), FLorida (FL) and New York (NY). We subsample 5% of each state's population for faster computation. To further ease computation load, we also subsample features by fitting a random forest regressor to the entire pooled dataset of all the populations, and picking the top 18 most important features. We then perform feature selection using our method, and the baselines. For the downstream evaluation, since the target is real valued (similar to the synthetic datasets), we fit a Random Forest Regressor using the selected features. For further details about hyperparameter choices, please see Appendix D.3.

Data pre-processing Categorical features are one-hot encoded to create numerical representations. Missing values are imputed using median imputation. Any instances with missing target values are dropped. The numerical features are standardized to ensure consistent scale across features. From the ACS data, we drop all non-numeric or weight/geoid features, and retain only the numeric features.

4.3 Results

Our method consistently outperforms all baseline approaches across both the ACS and UCI datasets. On the ACS regression task, we observe (Figure 3a) an order-of-magnitude reduction in MSE and substantial gains in R^2 across all populations. This indicates that our feature selection procedure identifies highly informative variables. Notably, even strong baselines such as DRO-XGBoost suffer from significantly higher error and variance, suggesting sensitivity to spurious or less transferable



(a) **ACS dataset results:** Comparison of mean squared error (MSE) and R^2 score (with standard deviations) for each method across California (CA), Florida (FL), and New York (NY). Our method achieves an order-of-magnitude lower MSE and substantially higher R^2 than all baselines, with consistently low run-to-run variance.

(b) **UCI dataset results:** Downstream classification performance (mean accuracy and log loss with standard deviations) for each method on Female and Male populations. Our method achieves the highest accuracy in both groups while maintaining a comparative log loss and generally lower run-to-run variance.

Figure 3: Performance comparison across populations on real datasets using different feature selection methods. In each subplot, left: Mean Squared Error; right: R^2 Score.

features across groups. We also note that on the ACS dataset, the Lasso variants outperform XGBoost. For exact metric values, please see Table 3 in Appendix D.3.

On the UCI classification task, our method generally outperforms the baselines across both populations, with the exception of the Log Loss over the Female population, where XGBoost outperforms it (Figure 3b). However, our method has a more balanced performance across the two population. Furthermore, the consistently low standard deviation across all reported metrics demonstrates the stability of our method across random initializations and training splits. For exact metric values, please see Table 4 in Appendix D.4.

5 Discussion and future work

We have presented a novel noise-based continuous relaxation framework for distributionally robust feature selection in a group-DRO setting. In contrast to existing methods that optimize for average-case performance or train a single robust model, our approach directly targets the selection of features that enable high-quality models across multiple subpopulations. By injecting feature-wise noise and optimizing the Bayes-optimal predictor’s variance, we derive a model-agnostic and computationally tractable objective that avoids differentiating through model training. This formulation allows us to identify features with stable predictive utility under distribution shifts.

Empirical results highlight the limitations of standard selection techniques—such as Lasso—in capturing non-linear or population-specific signal. Our method achieves substantially improved performance, including an order-of-magnitude reduction in MSE on the ACS dataset, underscoring its practical value in real-world settings.

In future work, we aim to apply our method to a broader range of real-world datasets to further assess its generalizability and practical utility. Replacing our current plug-in estimators with influence function-based approaches could reduce estimation bias, especially when dealing with limited samples from minority populations. This would improve the robustness of our feature selection when population sizes are imbalanced.

Additionally, extending our framework beyond MSE loss presents an interesting theoretical challenge. Our derivation leverages the bias–variance decomposition specific to mean squared error, though generalized variance decompositions have been proposed for broader classes of loss functions, including proper scoring rules such as cross-entropy. Incorporating these generalized forms could allow our method to directly optimize alternative objectives, though this would require nontrivial extensions of the current theoretical framework. Empirically, our results with cross-entropy loss in the UCI experiment suggest that optimizing MSE within our framework transfers well to other proper scoring losses, indicating that our bias–variance-based reasoning may extend beyond MSE in practice. Developing a theoretical foundation for such extensions is an important direction for future work.

Acknowledgments

Research reported in this publication was supported by the National Institute of Mental Health of the National Institutes of Health under award number R01MH139097, and the AI Research Institutes Program funded by the National Science Foundation under AI Institute for Societal Decision Making (AI-SDM), Award No. 2229881.

References

- Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.
- Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pages 136–145. PMLR, 2017.
- Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: <https://doi.org/10.24432/C5XW20>.
- Chris M Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7 (1):108–116, 1995.
- L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. URL <https://link.springer.com/article/10.1023/A:1010933404324>.
- Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794. ACM, August 2016. doi: 10.1145/2939672.2939785. URL <http://dx.doi.org/10.1145/2939672.2939785>.
- Abhimanyu Das and David Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection, 2011. URL <https://arxiv.org/abs/1102.3975>.
- John Duchi and Hongseok Namkoong. Learning models with uniform performance via distributionally robust optimization, 2020. URL <https://arxiv.org/abs/1810.08750>.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.
- Yves Grandvalet. Anisotropic noise injection for input variables relevance determination. *IEEE transactions on neural networks*, 11 6:1201–12, 2000. URL <https://ieeexplore.ieee.org/document/883393>.
- Isabelle M Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003. URL <https://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf>.
- Tatsunori B. Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. Fairness without demographics in repeated loss minimization. *ArXiv*, abs/1806.08010, 2018. URL <https://arxiv.org/abs/1806.08010>.
- Rajiv Khanna, Ethan Elenberg, Alex Dimakis, Sahand Negahban, and Joydeep Ghosh. Scalable greedy feature selection via weak submodularity. In *Artificial Intelligence and Statistics*, pages 1560–1568. PMLR, 2017.
- Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97 (1-2):273–324, 1997.

- Kurt Kroenke, Robert L Spitzer, and Janet BW Williams. The phq-9: validity of a brief depression severity measure. *Journal of general internal medicine*, 16(9):606–613, 2001.
- Kurt Kroenke, Robert L Spitzer, and Janet BW Williams. The patient health questionnaire-2: validity of a two-item depression screener. *Medical care*, 41(11):1284–1292, 2003.
- Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.
- Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995.
- Radford M. Neal. Bayesian learning for neural networks. 1996. URL <https://link.springer.com/book/10.1007/978-1-4612-0745-0>.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL <https://arxiv.org/abs/1912.01703>.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.
- Robert L Spitzer, Kurt Kroenke, Janet BW Williams, Patient Health Questionnaire Primary Care Study Group, Patient Health Questionnaire Primary Care Study Group, et al. Validation and utility of a self-report version of prime-md: the phq primary care study. *Jama*, 282(18):1737–1744, 1999.
- Jonathan Svirsky and Ofir Lindenbaum. Interpretable deep clustering for tabular data. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 47314–47330. PMLR, 21–27 Jul 2024. URL <https://proceedings.mlr.press/v235/svirsky24a.html>.
- Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.
- Michael E Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of machine learning research*, 1(Jun):211–244, 2001.
- U.S. Census Bureau. American community survey [1-year] [1-year/5-year] estimates. Dataset, U.S. Census Bureau, 2018. Retrieved from [URL], [Access date].
- Junchen Yang, Ofir Lindenbaum, and Yuval Kluger. Locally sparse neural networks for tabular biomedical data. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 25123–25153. PMLR, 17–23 Jul 2022. URL <https://proceedings.mlr.press/v162/yang22i.html>.
- Peng Zhao and Bin Yu. On model selection consistency of lasso. *Journal of Machine learning research*, 7(Nov):2541–2563, 2006.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: Section 3, Section 4

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: Section 5

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: Section 3

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: Section 4Appendix D

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [\[Yes\]](#)

Justification: Abstract - see footnote. Section 4

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: Section 4, Appendix D

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[Yes\]](#)

Justification: Section 4, Appendix D

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: Appendix D

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines?>

Answer: [\[Yes\]](#)

Justification: Yes

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [\[Yes\]](#)

Justification: Section 5

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: See citations and Section 4

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

A Theoretical Results: Derivations and Intuition

In this appendix, we provide detailed derivations and additional intuition for the theoretical results presented in the main text. We begin by expanding the population-level formulation of the objective, clarifying its interpretation in terms of conditional variances. We then derive the equivalent kernel-based form used in the algorithmic implementation.

A.1 Population-level objective

In this section, we derive the population-level objective in Theorem 1. Recall that the Bayes-optimal predictor for Y given $S(\alpha)$, is given by $f^*(S(\alpha)) = \mathbb{E}[Y|S(\alpha)]$. With respect to the MSE, the expected loss of this optimal predictor is the conditional variance of Y given $S(\alpha)$ (since bias is 0):

$$\mathbb{E}_{S(\alpha), Y \sim P_i} [(Y - \mathbb{E}[Y|S(\alpha)])^2] = \mathbb{E}_{S(\alpha) \sim P_i} [\mathbb{V}[Y|S(\alpha)]]$$

We can expand this using the law of total variance to rewrite

$$\mathbb{V}[Y|S(\alpha)] = \mathbb{E}_X [\mathbb{V}[Y|S(\alpha), X]|S(\alpha)] + \mathbb{V}[\mathbb{E}[Y|S(\alpha), X]|S(\alpha)]. \quad (5)$$

Given the generative process $S(\alpha) = X + \alpha \sqrt{\epsilon}$, where the noise ϵ is independently sampled, we have $Y \perp\!\!\!\perp S|X$ and thus can drop $S(\alpha)$ from the inner conditioning terms. We obtain:

$$\mathbb{V}[Y|S(\alpha)] = \mathbb{E}_X [\mathbb{V}[Y|X]|S(\alpha)] + \mathbb{V}[\mathbb{E}[Y|X]|S(\alpha)]. \quad (6)$$

This expression has the appealing property that α only enters through the outer conditioning: it changes the distribution over which we take the expectation/variance, but not the variable we are taking the expectation/variance of. Thus we may estimate $\mathbb{V}[Y|X]$ and $\mathbb{E}[Y|X]$ just once per test distribution P_i , instead of having to fit a new model for every value α . However, we can pursue further simplifications to arrive at an even more streamlined objective with closed-form dependence on α .

First, recall that our objective is to minimize the expected variance $\mathbb{E}_{S(\alpha)} [\mathbb{V}[Y|S(\alpha)]]$. When wrapped in this outer expectation, the first term in Equation 6 above becomes constant with respect to α : $\mathbb{E}_{S(\alpha)} [\mathbb{E}_X [\mathbb{V}[Y|X]|S(\alpha)]] = \mathbb{E}_X [\mathbb{V}[Y|X]]$. Accordingly, we can drop it from the optimization objective. We are left with the problem

$$\min_{\alpha} \max_{P_i \in \mathcal{P}} \mathbb{E}_{S(\alpha) \sim P_i} [\mathbb{V}[\mathbb{E}[Y|X]|S(\alpha)]] \quad (7)$$

Intuitively the inner variance term $\mathbb{V}[\mathbb{E}[Y|X]|S(\alpha)]$ represents the variance of the true conditional expectation $\mathbb{E}[Y|X]$ conditioned on us observing $S(\alpha)$. Specifically, given $S(\alpha) = s$, there is a distribution of possible true covariate (X) values that could have generated the corresponding $S(\alpha)$ through the noising process. Each of these potential X values has an associated true conditional mean $\mathbb{E}[Y|X]$. The term $\mathbb{V}[\mathbb{E}[Y|X]|S(\alpha) = s]$ measures the variability or *spread* of these true $\mathbb{E}[Y|X]$ values, given the observed s . A high value of this conditional variance indicates that a single noisy observation is consistent with a wide range of possible values for X , and correspondingly, $\mathbb{E}[Y|X]$ values. In this scenario, the noise introduced by α has significantly obscured the relationship between the observed $S(\alpha)$ and the underlying true predictive signal $\mathbb{E}[Y|X]$. Conversely, a low value for this variance implies that $S(\alpha)$ is highly informative about $\mathbb{E}[Y|X]$. Our objective, therefore, seeks to choose noise levels α such that, even under the worst-case test distribution, the average uncertainty (variance) about the true predictive function $\mathbb{E}[Y|X]$ given the noised observation $S(\alpha)$ is minimized. In order to further simplify the objective we expand the variance term above. Let $\mu_i(X) = \mathbb{E}_{P_i}[Y|X]$ denote the conditional mean function on the i th population. We have

$$\mathbb{E}_{S(\alpha) \sim P_i} [\mathbb{V}[\mu_i(X)|S(\alpha)]] = \mathbb{E}_{S(\alpha) \sim P_i} [E_X [\mu_i(X)^2|S(\alpha)]] - \mathbb{E}_{S(\alpha) \sim P_i} [\mathbb{E}[\mu_i(X)|S(\alpha)]^2]$$

where the first term again collapses to $\mathbb{E}[\mu_i(X)^2]$, which is constant with respect to α and can be dropped from the optimization. We are left with the second term, $-\mathbb{E}_{S(\alpha) \sim P_i} [\mathbb{E}[\mu_i(X)|S(\alpha)]^2]$. This term requires us to average $\mu(X_i)$ over the conditional distribution of X given S .

A.2 Kernel Form Equivalence Proof

In this section, we derive the kernel-form of the objective in Theorem 2. We need to estimate the conditional expectation over $\mu_i(X)$. By applying Bayes theorem, we can arrive at an estimate for

this expression with a closed form in terms of α . Specifically, we can rewrite

$$\mathbb{E}[\mu_i(X)|S] = \int \mu_i(X) dP_i(X|S(\alpha))$$

where μ is averaged over the distribution of X implied by the noise model, $P_i(X|S(\alpha))$. Via Bayes theorem, we can rewrite

$$P_i(X|S(\alpha)) = \frac{P_i(X)P_i(S(\alpha)|X)}{P_i(S(\alpha))}.$$

We propose to estimate this quantity, and hence the objective function, by setting the “prior” $P_i(X)$ in this expression to be the uniform distribution over the observed samples of X from population P_i , $X_i^1 \dots X_i^{n_i}$. The likelihood $P_i(S(\alpha)|X)$ under our Gaussian noise model for S is given simply by

$$P_i(S(\alpha)|X) \propto \exp \left\{ -\frac{1}{2}(X - S)^T \text{diag}(\alpha)^{-1}(X - S) \right\}$$

and we can then calculate the denominator as $\frac{1}{n_i} \sum_{j=1}^{n_i} P_i(S(\alpha)|X_i^j)$. Putting this all together, let

$$w_i^j(S, \alpha) = \frac{P_i(S(\alpha)|X_i^j)}{\sum_{j=1}^{n_i} P_i(S(\alpha)|X_i^j)}$$

denote the estimate of the probability of observed data point X_i^j . We arrive at the estimator

$$\widehat{\mathbb{E}}[\mu_i(X)|S]^2 = \left(\sum_{j=1}^{n_i} w_i^j(S, \alpha) \mu_i(X_i^j) \right)^2$$

which can be interpreted as kernel smoothing of $\mu_i(X)$ under the Gaussian kernel implied by the noise model, measuring the amount of information lost due to the smoothing.

B DRO Feature Selection Algorithm and complexity

In this section, we summarize the full procedure for optimizing the noise-based relaxation and obtaining the final set of robust features described in section 3. We present the algorithmic implementation of our method, followed by an analysis of its computational complexity. The proposed procedure, is outlined in algorithm 1.

Computational complexity per iteration Our method requires $O(P \cdot b \cdot n \cdot K \cdot d)$ operations per iteration, where P is the number of populations, b is the number of Monte Carlo samples, $n = \max_p n_p$ is the maximum population size, K is the number of nearest neighbors used for kernel weight computation, and feature dimensionality d .

C Baseline Methods

Data Pooling & Standardization: Data points $(X_i^{(p)}, Y_i^{(p)})$ from all populations $p = 1, \dots, P$ are pooled into a single dataset $\{(X_j, Y_j)\}_{j=1}^N$. Both features X and outcomes Y are then standardized to have zero mean and unit variance.

We summarize the baseline methods below:

1. **Vanilla Lasso regression**
2. **Vanilla XGBoost regression**
3. **DRO Lasso regression**
4. **DRO Lasso regression**
5. **Embedded MLP**

Algorithm 1 Distributionally Robust Feature Selection

Input: Dataset $\{(X_j^{(p)}, Y_j^{(p)})\}_{j=1}^{n_p}$ for populations $p = 1, \dots, P$; budget k ; regularization parameter λ ; learning rate η ; Monte Carlo samples b ; number of nearest neighbors K

Output: Feature noise parameters $\alpha^* \in \mathbb{R}_{\geq 0}^m$

- 1: **Precompute:** For each population p :
- 2: Fit model $\hat{\mu}_p(X) \approx \mathbb{E}[Y|X]$ using $\{(X_j^{(p)}, Y_j^{(p)})\}_{j=1}^{n_p}$
- 3: Build k-NN index for $\{X_j^{(p)}\}_{j=1}^{n_p}$ ▷ Optional: for efficiency
- 4: **Initialize:** $\alpha^{(0)} \leftarrow \mathbf{1} + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$
- 5: **for** epoch = 1 **to** $T_{\max}$ **do**
- 6: $\mathcal{L} \leftarrow 0$
- 7: **for** $p = 1$ **to** P **do**
- 8: $\mathcal{L}_p \leftarrow 0$
- 9: **for** $\ell = 1$ **to** b **do**
- 10: Sample noise $\xi \sim \mathcal{N}(0, I)$
- 11: Generate noisy observation $S^{(\ell)} \leftarrow X + \sqrt{\alpha^{(\text{epoch}-1)}} \odot \xi$
- 12: **if** using k-NN **then**
- 13: $\mathcal{N} \leftarrow$ k-nearest neighbors of $S^{(\ell)}$ in $\{X_j^{(p)}\}_{j=1}^{n_p}$
- 14: **else**
- 15: $\mathcal{N} \leftarrow \{1, \dots, n_p\}$
- 16: **for** $j = 1$ **to** n_p **do**
- 17: Compute weights:
- 18: $w_j^p(S^{(\ell)}, \alpha) \leftarrow \frac{\exp\left(-\frac{1}{2}(X_j^{(p)} - S^{(\ell)})^T \text{diag}(\alpha)^{-1}(X_j^{(p)} - S^{(\ell)})\right)}{\sum_{j' \in \mathcal{N}} \exp\left(-\frac{1}{2}(X_{j'}^{(p)} - S^{(\ell)})^T \text{diag}(\alpha)^{-1}(X_{j'}^{(p)} - S^{(\ell)})\right)}$
- 19: $\mathcal{L}_p \leftarrow \mathcal{L}_p - \frac{1}{b} \left(\sum_{j=1}^{n_p} w_j^p(S^{(\ell)}, \alpha) \hat{\mu}_p(X_j^{(p)}) \right)^2$
- 20: $\mathcal{L} \leftarrow \max_{p \in \{1, \dots, P\}} \mathcal{L}_p + \lambda \cdot \text{Reg}(\alpha)$ ▷ or use softmax
- 21: $\alpha^{(\text{epoch})} \leftarrow \alpha^{(\text{epoch}-1)} - \eta \nabla_{\alpha} \mathcal{L}$
- 22: Project $\alpha^{(\text{epoch})}$ to $\mathbb{R}_{\geq 0}^m$ ▷ Ensure non-negativity
- 23: **Feature Selection:** Select k features with smallest α^* values
- 24: **return** Selected feature indices $\mathcal{I} = \{\text{indices of } k \text{ smallest } \alpha_i^*\}$

Vanilla Lasso This method applies Lasso regression to the combined dataset from all populations. The Lasso model is trained by solving:

$$\min_{\beta \in \mathbb{R}^d} \frac{1}{2N} \sum_{j=1}^N (Y_j - X_j^T \beta)^2 + \lambda_L \|\beta\|_1$$

We set λ_L value to 0.01. This selection is performed by retraining the model for each λ_L and evaluating its performance on the full training set. Once the coefficients β^* are determined, the k features with the largest absolute coefficient values ($|\beta_i^*|$) are selected.

Vanilla XGBoost This method utilizes feature importance scores from an XGBoost model trained on the pooled dataset. Hyperparameters such as tree depth, learning rate, etc., are set to default values provided by the XGBoost library [Chen and Guestrin, 2016]. Feature importance scores are extracted from the trained XGBoost model using the default `feature_importances_` attribute). The k features with the highest importance scores are selected.

Distributionally Robust Optimization (DRO) Lasso This method adapts Lasso to be robust by iteratively re-weighting populations to focus on worst-case performance. Features $X^{(p)}$ and outcomes $Y^{(p)}$ are standardized separately for each population p . We then perform **DRO Lasso**, summarized in Algorithm 2.

The k features with the largest absolute coefficient values from the final Lasso model $\beta^{(T_{\max})}$ are selected.

Algorithm 2 DRO Lasso

- 1: **Initialize** uniform weights $w_p^{(0)} = \frac{1}{P}$ for each population $p = 1, \dots, P$.
- 2: **Standardize** each population's data $(X^{(p)}, Y^{(p)})$ independently.
- 3: **Select** L1 regularization parameter λ_L
- 4: **for** $t = 1 \dots T_{\max}$ **do**
- 5: **Construct Pooled Dataset with Sample Weights:**
 Form $X_{\text{all}}, Y_{\text{all}}$ by concatenating all $X^{(p)}, Y^{(p)}$, and assign weight $w_p^{(t)}$ to each sample from population p .
 Fit the Lasso model:

$$\min_{\beta \in \mathbb{R}^d} \sum_{p=1}^P \sum_{i=1}^{N_p} w_p^{(t)} (Y_i^{(p)} - X_i^{(p)T} \beta)^2 + \lambda_L \|\beta\|_1$$

- 6: **Compute Population Losses:** Calculate the MSE for the current model $\beta^{(t)}$ on each original, unweighted standardized population p :

$$\text{loss}_p^{(t)} = \text{MSE}(\beta^{(t)}; X^{(p)}, Y^{(p)})$$

- 7: **Weight Update:** Update population weights for the next iteration:

$$w_p^{(t+1)'} = w_p^{(t)} \exp(\eta \cdot \text{loss}_p^{(t)})$$

 Normalize the weights:

$$w_p^{(t+1)} = \frac{w_p^{(t+1)'}}{\sum_{j=1}^P w_j^{(t+1)'}}$$

Distributionally Robust Optimization (DRO) XGBoost This method adapts XGBoost to a distributionally robust setting by iteratively re-weighting populations based on worst-case performance. Each population's data is standardized separately. The procedure aims to select the k most important features by accounting for heterogeneity in population-level performance. The method is summarized in Algorithm 3.

Algorithm 3 DRO XGBoost

- Initialize** uniform weights $w_p^{(0)} = \frac{1}{P}$ for each population $p = 1, \dots, P$.
Standardize each population's data $(X^{(p)}, Y^{(p)})$ independently.
for $t = 1 \dots T_{\max}$ **do**
 Construct Pooled Dataset with Sample Weights:
 Form $X_{\text{all}}, Y_{\text{all}}$ by concatenating all $X^{(p)}, Y^{(p)}$, and assign weight $w_p^{(t)}$ to each sample from population p .
 Train XGBoost Model:
 Fit an XGBoost regressor or classifier (depending on the task) on the pooled data using sample weights.
 Compute Population Losses:
 For each population p , compute: $\text{loss}_p^{(t)} = \begin{cases} \frac{1}{N_p} \sum_{i=1}^{N_p} (Y_i^{(p)} - \hat{Y}_i^{(p)})^2 & \text{(Regression)} \\ \frac{1}{N_p} \sum_{i=1}^{N_p} \log \text{loss}(Y_i^{(p)}, \hat{Y}_i^{(p)}) & \text{(Classification)} \end{cases}$
 Update Population Weights: $w_p^{(t+1)'} = w_p^{(t)} \exp(\eta \cdot \text{loss}_p^{(t)})$, $w_p^{(t+1)} = \frac{w_p^{(t+1)'}}{\sum_{j=1}^P w_j^{(t+1)'}}$
-

The k features with the largest feature importance scores from the final XGBoost model are selected.

Embedded baseline – Embedded MLP We include an embedded baseline, namely Embedded MLP which uses an MLP with a learnable feature mask trained via DRO. The MLP has a single hidden layer of size 100. We train the model with the joint objective of MSE minimization (for the

regression task), and L-1 regularization (weighted by hyperparameter $\lambda = 0.01$; we found this value to work best out of $[0.1, 0.01, 0.001]$). The training procedure alternates between neural network training and population weight updates over multiple iterations (with the DRO weights being updated once every 10 epochs of the model training). The learnable mask parameters are constrained to $[0, 1]$ and the top- k features are selected based on the final mask values after training convergence. We train for a maximum of 200 epochs, with early stopping with patience of 10 epochs when the average loss improvement falls below 10^{-4} .

D Results (Continued)

D.1 Synthetic dataset 1: Linear model

Table 1: Results for synthetic experiment 1 (Section 4.1). Performance comparison across methods and populations for different feature budgets. Total number of features is 15.

Method	Pop.	Budget 5		Budget 10	
		MLP MSE ↓	RF MSE ↓	MLP MSE ↓	RF MSE ↓
Our Method	A	0.2251 ± 0.0029	0.2486 ± 0.0018	0.0342 ± 0.0004	0.0535 ± 0.0010
	B	0.2888 ± 0.0113	0.3194 ± 0.0098	0.0207 ± 0.0005	0.0804 ± 0.0036
	C	0.2787 ± 0.0150	0.3039 ± 0.0130	0.0039 ± 0.0004	0.0339 ± 0.0079
DRO Lasso	A	0.5071 ± 0.0129	0.5434 ± 0.0191	0.0338 ± 0.0009	0.0534 ± 0.0009
	B	0.2886 ± 0.0086	0.3194 ± 0.0026	0.0211 ± 0.0001	0.0807 ± 0.0041
	C	0.2755 ± 0.0150	0.3031 ± 0.0113	0.0039 ± 0.0004	0.0340 ± 0.0078
DRO XGB	A	0.5107 ± 0.0113	0.5441 ± 0.0188	0.0738 ± 0.0370	0.0887 ± 0.0327
	B	0.2888 ± 0.0081	0.3187 ± 0.0030	0.0408 ± 0.0198	0.0928 ± 0.0145
	C	0.2762 ± 0.0151	0.3032 ± 0.0112	0.0036 ± 0.0003	0.0337 ± 0.0076
Lasso	A	1.0087 ± 0.0153	1.0571 ± 0.0107	0.2056 ± 0.1714	0.2106 ± 0.1625
	B	0.5744 ± 0.0051	0.6150 ± 0.0052	0.1160 ± 0.0921	0.1460 ± 0.0763
	C	0.1080 ± 0.0107	0.1283 ± 0.0102	0.0036 ± 0.0003	0.0340 ± 0.0076
XGB	A	1.0136 ± 0.0076	1.0623 ± 0.0039	0.4051 ± 0.2802	0.3905 ± 0.2579
	B	0.5702 ± 0.0008	0.6170 ± 0.0076	0.2279 ± 0.1586	0.2373 ± 0.1307
	C	0.1314 ± 0.0291	0.1538 ± 0.0352	0.0037 ± 0.0002	0.0338 ± 0.0077
Embedded MLP	A	0.7355 ± 0.2702	0.7709 ± 0.2636	0.5336 ± 0.3503	0.5011 ± 0.3107
	B	0.5599 ± 0.1363	0.6043 ± 0.1310	0.4170 ± 0.2951	0.4009 ± 0.2521
	C	0.2805 ± 0.1630	0.3077 ± 0.1728	0.2801 ± 0.2117	0.2649 ± 0.1923

¹ Values reported as mean ± standard deviation.

² ↓ indicates lower values are better.

³ Best results per population metric are highlighted in **bold**.

Implementation details α is initialized values to near 1 by adding random noise to a vector of ones. We use Adam optimizer Kingma [2014] with a learning rate of 0.1. We also use a CosineAnnealing Scheduler for the learning rate, and train the model for 200 epochs. For the kernel estimation, we set the number of nearest neighbours $k = 1000$. We take 10 Monte Carlo samples for estimating the objective. At each epoch, we do a full-batch gradient descent. For the objective, we use the hard-max formulation (setting the SoftMax parameter to `inf`). The penalty term is a reciprocal of the L_1 norm of α . The entire dataset (combining all splits) is of size 36000. We ran the experiment for 3 different seeds and reported the average over all runs as seen in Figure 1. Experiments were conducted on an Apple MacBook Pro equipped with an Apple M3. We set the budget to 5 and, for comparison, also include the results for an increased budget of 10. The metrics are summarized in Table 1.

Consistency across seeds Our method consistently selects a similar core ordered set of features across (3) different seeds (in decreasing order of importance) $[0, 7, 1, 5, 8, 6, 9, 2, 10, 3]$, $[0, 7, 1, 8, 5, 9, 6, 2, 10, 3]$, $[0, 7, 5, 1, 8, 6, 9, 2, 10, 3]$. In contrast, baseline methods, especially non-DRO versions, show significant variability, often selecting irrelevant noisy features (e.g., 13, 14) and failing to consistently identify the true signal variables.

D.2 Synthetic dataset 2: Nonlinear

Table 2: Results for synthetic experiment 1 (Section 4.1). Performance comparison across methods and populations for different feature budgets. Total number of features is 50, and budget is set to 8.

Method	Pop.	MLP MSE ↓	RF MSE ↓
Our Method	A	0.0054 ± 0.0007	0.0150 ± 0.0033
	B	0.0062 ± 0.0010	0.0205 ± 0.0016
	C	0.0123 ± 0.0080	0.2460 ± 0.0483
	D	0.0114 ± 0.0015	0.0130 ± 0.0008
DRO Lasso	A	0.4115 ± 0.0554	0.4020 ± 0.0560
	B	0.4172 ± 0.0533	0.4230 ± 0.0511
	C	0.3114 ± 0.2644	0.3816 ± 0.1018
	D	0.3569 ± 0.0259	0.3375 ± 0.0089
DRO XGB	A	0.0054 ± 0.0014	0.0150 ± 0.0034
	B	0.0064 ± 0.0009	0.0206 ± 0.0017
	C	0.0226 ± 0.0005	0.2460 ± 0.0479
	D	0.0114 ± 0.0012	0.0132 ± 0.0010
Lasso	A	0.0517 ± 0.0417	0.0616 ± 0.0433
	B	0.0562 ± 0.0443	0.0657 ± 0.0390
	C	0.4654 ± 0.0573	0.4581 ± 0.0676
	D	0.0114 ± 0.0014	0.0130 ± 0.0012
XGB	A	0.0051 ± 0.0013	0.0151 ± 0.0033
	B	0.0067 ± 0.0012	0.0206 ± 0.0015
	C	0.0200 ± 0.0039	0.2471 ± 0.0475
	D	0.0117 ± 0.0017	0.0131 ± 0.0007
Embedded MLP	A	0.0050 ± 0.0008	0.0150 ± 0.0035
	B	0.0062 ± 0.0014	0.0207 ± 0.0019
	C	0.3030 ± 0.2435	0.4119 ± 0.1885
	D	0.0119 ± 0.0019	0.0128 ± 0.0008

¹ Values reported as mean ± standard deviation for Budget 8.

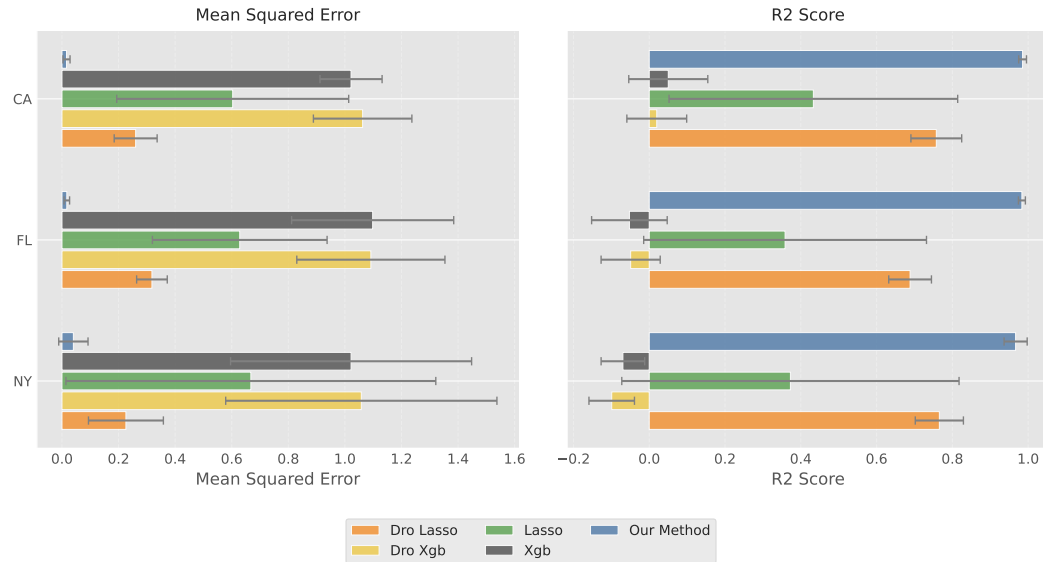
² ↓ indicates lower values are better.

³ Budget refers to the number of features selected.

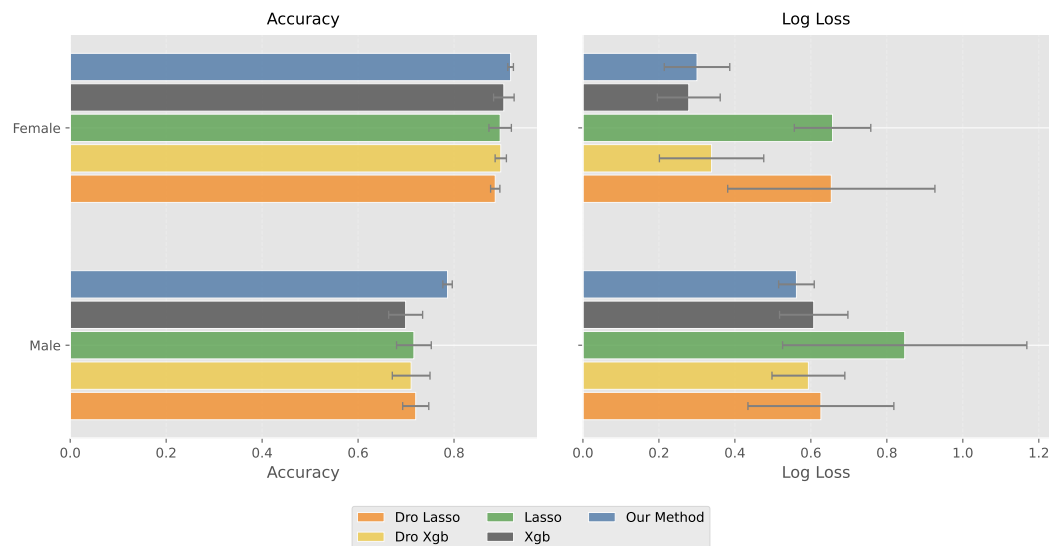
Implementation details α is initialized values to near 2 by adding random noise to a vector of twos. We use Adam optimizer Kingma [2014] with a learning rate of 0.1. We also use a CosineAnnealing Scheduler for the learning rate, and train the model for 150 epochs. For the kernel estimation, we set the number of nearest neighbours $k = 1000$. We take 50 Monte Carlo samples for estimating the objective. At each epoch, we do a full-batch gradient descent. For the objective, we use the hard-max formulation (setting the SoftMax parameter to inf). The penalty term is a reciprocal of the L_1 norm of α . The entire dataset (combining all splits) is of size 44000. We ran the experiment for 3 different seeds and reported the average over all runs as seen in Figure 2. Experiments were conducted on an Apple MacBook Pro equipped with an Apple M3. We set the budget to 8. The metrics are summarized in Table 2.

D.3 ACS Dataset

Implementation details α is initialized values to near 5 by adding random noise to a vector of 5s. We use an Adam optimizer, set initial learning rates of 0.01 with a CosineAnnealing Scheduler for the learning rate, and run the optimization for 120 epochs. For the kernel estimation, we set the number of nearest neighbours $k = 500$. We take 10 Monte Carlo samples for estimating the objective. At each epoch, we do a full-batch gradient descent. For the objective, we use the SoftMax formulation (setting the SoftMax parameter to 10). The penalty term is a reciprocal of the L_1 norm of α , with $\lambda = 1e - 4$. We ran the experiment for 5 different seeds and reported the average over all runs as seen in Figure 3a. For the ACS dataset, we set the budget to 7 out of 18 features. The metrics are summarized in Table 3. For larger visualizations for greater readability, see Figure 4a.



(a) **ACS dataset results:** Comparison of mean squared error (MSE) and R^2 score (with standard deviations) for each method across California (CA), Florida (FL), and New York (NY). Our method achieves an order-of-magnitude lower MSE and substantially higher R^2 than all baselines, with consistently low run-to-run variance.



(b) **UCI dataset results:** Downstream classification performance (mean accuracy and log loss with standard deviations) for each method on Female and Male populations. Our method achieves the highest accuracy in both groups while maintaining a comparative log loss and lower run-to-run variance.

Figure 4: **Performance comparison across populations on real datasets using different feature selection methods.**

D.4 UCI Dataset

α is initialized values to near 5 by adding random noise to a vector of 5s. We use an Adam optimizer, and set initial learning rates to 0.02 with a CosineAnnealing Scheduler for the learning rate, and run the optimization for 120 epochs. For the kernel estimation, we set the number of nearest neighbours $k = 500$. We take 10 Monte Carlo samples for estimating the objective. At each epoch, we do a full-batch gradient descent. For the objective, we use the SoftMax formulation (setting the SoftMax parameter to 10). The penalty term is a reciprocal of the L_1 norm of α , with $\lambda = 5e - 6$. Due to computational instability of the Logistic Regression baseline, we ran the experiment for 3 different

Table 3: ACS dataset: Performance comparison across methods and populations

Method	Population	MSE ↓	R^2 ↑
Baseline DRO Lasso	CA	0.260 ± 0.076	0.758 ± 0.067
	FL	0.318 ± 0.054	0.688 ± 0.056
	NY	0.226 ± 0.132	0.766 ± 0.064
Baseline DRO XGBoost	CA	1.063 ± 0.174	0.020 ± 0.079
	FL	1.092 ± 0.262	-0.049 ± 0.078
	NY	1.058 ± 0.479	-0.099 ± 0.060
Baseline Lasso	CA	0.604 ± 0.410	0.433 ± 0.381
	FL	0.628 ± 0.309	0.359 ± 0.373
	NY	0.668 ± 0.654	0.373 ± 0.445
Baseline XGBoost	CA	1.022 ± 0.110	0.050 ± 0.104
	FL	1.098 ± 0.286	-0.052 ± 0.100
	NY	1.022 ± 0.426	-0.069 ± 0.057
Our Method	CA	0.016 ± 0.012	0.985 ± 0.010
	FL	0.017 ± 0.010	0.983 ± 0.009
	NY	0.040 ± 0.052	0.967 ± 0.030

¹ Values reported as mean $\pm$ standard deviation, rounded to three decimal places.

² ↓ indicates lower values are better; ↑ indicates higher values are better.

³ Best results per population metric are highlighted in **bold**.

Table 4: UCI dataset: Performance comparison across methods and populations

Method	Population	Accuracy ↑	Log Loss ↓
Baseline DRO Lasso	Female	0.886 ± 0.010	0.654 ± 0.273
	Male	0.720 ± 0.027	0.626 ± 0.192
Baseline DRO XGBoost	Female	0.897 ± 0.012	0.339 ± 0.137
	Male	0.711 ± 0.039	0.594 ± 0.096
Baseline Lasso	Female	0.896 ± 0.023	0.657 ± 0.101
	Male	0.716 ± 0.036	0.847 ± 0.321
Baseline XGBoost	Female	0.904 ± 0.021	0.279 ± 0.083
	Male	0.699 ± 0.035	0.608 ± 0.090
Our Method	Female	0.918 ± 0.006	0.300 ± 0.086
	Male	0.786 ± 0.010	0.562 ± 0.047

¹ Values reported as mean $\pm$ standard deviation, rounded to three decimal places.

² ↑ indicates higher values are better; ↓ indicates lower values are better.

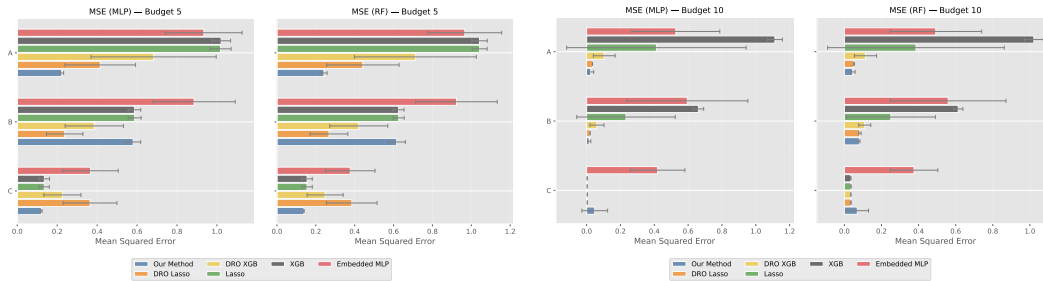
³ Best results per population metric are highlighted in **bold**.

seeds and reported the average over all runs as seen in Figure 3b. We set the budget to 5 out of 44 (real and one-hot-encoded) features. The metrics are summarized in Table 4. For larger visualizations for greater readability, see Figure 4b

E Additional synthetic experiments

E.1 Synthetic experiment 1: Linear model with increased dimension

In this section we include additional experiments for the data setting from Section 4.1, **synthetic experiment 1** (linear data setting), when the total number of features is set to 50. Here, the increase in dimension only contributes in additional noise variables, while the meaningful variables are the same as for dimension= 15. We set the budget for feature selection to 5 and, for comparison, also include the results for an increased budget of 10. Figure 5 shows the performance of the downstream



(a) **Budget = 5**: Our method consistently among the top-3 feature selection methods along with DRO-XGBoost and DRO-Lasso, even outperforming both methods in population *C*. (b) **Budget = 10**: Overall performance improves predictably upon reducing the budget. Our method maintains lower MSE compared to other methods, including DRO-Lasso, and has a balanced performance across populations.

Figure 5: **Performance comparison across populations on synthetic dataset 1 using different feature selection methods.** In each subplot, left: Mean Squared Error of downstream MLP model; right: Mean Squared Error of downstream random forest model. The relative performance of feature selection methods is consistent across the choice of downstream prediction model, with vanilla XGBoost consistently having the worst performance. Here the total number of features is set to 50.

Table 5: Additional results for synthetic experiment 1 (Section 4.1). Performance comparison across methods and populations for different feature budgets. Total number of features is 50.

Method	Pop.	Budget 5		Budget 10	
		MLP MSE ↓	RF MSE ↓	MLP MSE ↓	RF MSE ↓
Our Method	A	0.2227 ± 0.0094	0.2420 ± 0.0161	0.0241 ± 0.0178	0.0450 ± 0.0129
	B	0.5805 ± 0.0375	0.6158 ± 0.0448	0.0150 ± 0.0096	0.0809 ± 0.0035
	C	0.1208 ± 0.0034	0.1388 ± 0.0011	0.0480 ± 0.0761	0.0697 ± 0.0605
DRO Lasso	A	0.4155 ± 0.1769	0.4418 ± 0.1866	0.0345 ± 0.0004	0.0511 ± 0.0029
	B	0.2377 ± 0.0909	0.2677 ± 0.0972	0.0206 ± 0.0006	0.0835 ± 0.0067
	C	0.3642 ± 0.1350	0.3849 ± 0.1300	0.0040 ± 0.0007	0.0354 ± 0.0012
DRO XGB	A	0.6827 ± 0.3136	0.7120 ± 0.3138	0.1041 ± 0.0645	0.1139 ± 0.0597
	B	0.3857 ± 0.1461	0.4209 ± 0.1495	0.0613 ± 0.0416	0.1098 ± 0.0326
	C	0.2259 ± 0.0928	0.2479 ± 0.0926	0.0040 ± 0.0005	0.0350 ± 0.0010
Lasso	A	1.0181 ± 0.0526	1.0429 ± 0.0387	0.4126 ± 0.5303	0.3859 ± 0.4760
	B	0.5880 ± 0.0323	0.6264 ± 0.0284	0.2327 ± 0.2912	0.2504 ± 0.2401
	C	0.1351 ± 0.0252	0.1550 ± 0.0267	0.0037 ± 0.0001	0.0354 ± 0.0009
XGB	A	1.0211 ± 0.0474	1.0421 ± 0.0400	1.1132 ± 0.0439	1.0216 ± 0.0507
	B	0.5867 ± 0.0319	0.6258 ± 0.0282	0.6607 ± 0.0314	0.6135 ± 0.0252
	C	0.1356 ± 0.0261	0.1551 ± 0.0267	0.0036 ± 0.0004	0.0347 ± 0.0012
Embedded MLP	A	0.9346 ± 0.1914	0.9670 ± 0.1896	0.5275 ± 0.2598	0.4928 ± 0.2475
	B	0.8857 ± 0.2055	0.9238 ± 0.2096	0.5952 ± 0.3581	0.5597 ± 0.3116
	C	0.3666 ± 0.1390	0.3772 ± 0.1274	0.4200 ± 0.1610	0.3756 ± 0.1282

¹ Values reported as mean ± standard deviation.

² ↓ indicates lower values are better.

³ Best results per population metric are highlighted in **bold**.

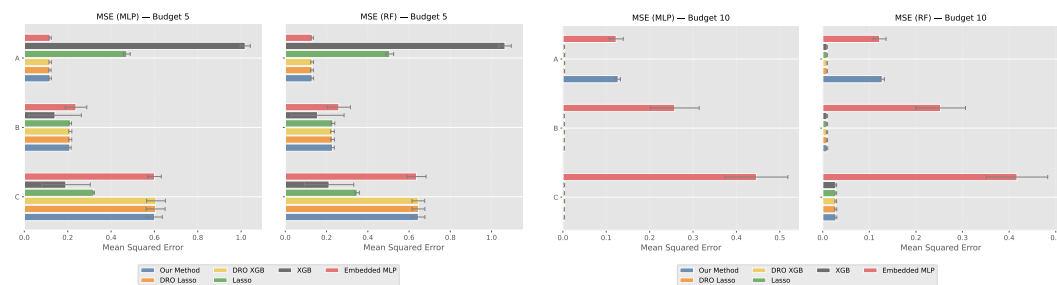
prediction models on the task of predicting *Y* for each population, using the features selected using our method and the baselines. The complete metrics can be found in Table 5.

Results Even in the higher dimensional setting, our method’s performance is comparable with the best performing baselines (DRO XGBoost and DRO Lasso), while maintaining a balanced performance across populations. Vanilla XGBoost shows the greatest degradation in performance when compared to the lower dimensional setting. Vanilla Lasso and Embedded MLP show the highest

variance across different seeds, while our method, along with DRO Lasso and DROXGBoost has the lowest variance across seeds.

Implementation details α is initialized values to near 1 by adding random noise to a vector of ones. We use Adam optimizer Kingma [2014] with a learning rate of 0.1. We also use a CosineAnnealing Scheduler for the learning rate, and train the model for 150 epochs. For the kernel estimation, we set the number of nearest neighbours $k = 1000$. We take 10 Monte Carlo samples for estimating the objective. At each epoch, we do a full-batch gradient descent. For the objective, we use the hard-max formulation (setting the SoftMax parameter to inf). The penalty term is a reciprocal of the L_1 norm of α . Our entire dataset (combining all splits) is of size 36000. We ran the experiment for 3 different seeds and reported the average over all runs as seen in Figure 5. Experiments were conducted on an Apple MacBook Pro equipped with an Apple M3. We set the budget to 5 and, for comparison, also include the results for an increased budget of 10.

E.2 Synthetic experiment 3: Sparse Linear



(a) **Budget = 5:** Our method consistently achieves low error across all populations, performing on par with DRO-XGBoost and DRO-Lasso, even outperforming both methods in population *A*. (b) **Budget = 10:** Overall performance improves predictably upon reducing the budget.

Figure 6: **Performance comparison across populations on synthetic dataset 3 using different feature selection methods.** In each subplot, left: Mean Squared Error of downstream MLP model; right: Mean Squared Error of downstream random forest model. The relative performance of feature selection methods is consistent across the choice of downstream prediction model.

This synthetic dataset comprises three populations *A*, *B*, *C*, with 50 features each. The populations have the following proportions in the dataset, and outcome functions:

$$\begin{aligned} \text{A (35\%)} \quad Y &= 5X_0 + 4X_{15} + 3X_{30} + \epsilon \\ \text{B (35\%)} \quad Y &= 6X_5 + 5X_{20} + 4X_{35} + \epsilon \\ \text{C (30\%)} \quad Y &= 7X_{10} + 6X_{25} + 5X_{40} + 4X_{45} + \epsilon \end{aligned}$$

The noise term follows a base distribution $\epsilon \sim \mathcal{N}(0, 0.1^2)$. Feature correlations are introduced via a generative process where $X_{i+1} = 0.3X_i + 0.7\eta$, with $\eta \sim \mathcal{N}(0, 1)$, inducing structured dependencies across dimensions. The dataset contains 50 features in total, the majority of which are noise. We set the budget for feature selection to 5 and, for comparison, also include the results for an increased budget of 10. Figure 6 shows the performance of the downstream prediction models on the task of predicting Y for each population, using the features selected using our method and the baselines. The complete metrics table can be found in Table 6.

Results We observe that our method performs similarly to the remaining baselines. For budget= 5, all models perform similarly, with the exception of vanilla XGBoost, which has the best performance on populations *B* and *C*, but seems to do so at the cost of its performance on population *A*. In a group-DRO setting, we would prefer to have a more balanced performance across all populations, as seen in the other methods. For budget= 10, the Embedded MLP performs the worst on all populations. Our method is consistent with the other baselines on populations *B* and *C*, however, in this setting it performs poorly on population *A*, potentially due to *A* having the weakest signals.

Table 6: Results for synthetic experiment 3. Performance comparison across methods and populations for different budgets. Total number of features is 50.

Method	Pop.	Budget 5		Budget 10	
		MLP MSE ↓	RF MSE ↓	MLP MSE ↓	RF MSE ↓
Our Method	A	0.1186 ± 0.0055	0.1299 ± 0.0064	0.1274 ± 0.0054	0.1277 ± 0.0047
	B	0.2099 ± 0.0061	0.2290 ± 0.0086	0.0034 ± 0.0001	0.0093 ± 0.0014
	C	0.6009 ± 0.0356	0.6453 ± 0.0318	0.0036 ± 0.0002	0.0282 ± 0.0021
DRO Lasso	A	0.1179 ± 0.0053	0.1296 ± 0.0066	0.0038 ± 0.0003	0.0095 ± 0.0003
	B	0.2115 ± 0.0070	0.2295 ± 0.0088	0.0033 ± 0.0004	0.0092 ± 0.0013
	C	0.6054 ± 0.0438	0.6456 ± 0.0310	0.0034 ± 0.0001	0.0281 ± 0.0021
DRO XGB	A	0.1189 ± 0.0057	0.1298 ± 0.0066	0.0038 ± 0.0000	0.0095 ± 0.0004
	B	0.2118 ± 0.0071	0.2286 ± 0.0092	0.0035 ± 0.0003	0.0092 ± 0.0014
	C	0.6073 ± 0.0432	0.6447 ± 0.0311	0.0037 ± 0.0004	0.0280 ± 0.0018
Lasso	A	0.4729 ± 0.0153	0.5066 ± 0.0185	0.0039 ± 0.0004	0.0095 ± 0.0004
	B	0.2115 ± 0.0057	0.2309 ± 0.0090	0.0036 ± 0.0004	0.0092 ± 0.0013
	C	0.3202 ± 0.0029	0.3485 ± 0.0106	0.0037 ± 0.0001	0.0281 ± 0.0018
XGB	A	1.0188 ± 0.0235	1.0651 ± 0.0303	0.0036 ± 0.0003	0.0095 ± 0.0004
	B	0.1414 ± 0.1214	0.1564 ± 0.1285	0.0034 ± 0.0002	0.0091 ± 0.0013
	C	0.1915 ± 0.1125	0.2121 ± 0.1196	0.0037 ± 0.0004	0.0280 ± 0.0019
Embedded MLP	A	0.1188 ± 0.0051	0.1306 ± 0.0054	0.1229 ± 0.0163	0.1222 ± 0.0138
	B	0.2381 ± 0.0499	0.2595 ± 0.0562	0.2579 ± 0.0562	0.2530 ± 0.0534
	C	0.6004 ± 0.0314	0.6364 ± 0.0459	0.4461 ± 0.0724	0.4168 ± 0.0659

¹ Values reported as mean ± standard deviation.

² ↓ indicates lower values are better.

³ Budget refers to the number of features selected.

Implementation details α is initialized values to near 2 by adding random noise to a vector of twos. We use Adam optimizer Kingma [2014] with a learning rate of 0.1. We also use a CosineAnnealing Scheduler for the learning rate, and train the model for 150 epochs. For kernel estimation, we set the number of nearest neighbors $k = 1000$. We take 50 Monte Carlo samples for estimating the objective. At each epoch, we do a full-batch gradient descent. For the objective, we use the hard-max formulation (setting the SoftMax parameter to inf). The penalty term is a reciprocal of the L_1 norm of α . Our entire dataset (combining all splits) is of size 36000. We ran the experiment for 3 different seeds and reported the average over all runs as seen in Figure 1. Experiments were conducted on an Apple MacBook Pro equipped with an Apple M3. We set the budget to 5 and, for comparison, also include the results for an increased budget of 10.