# Evaluating robustness of tabular models under meta-features based shifts

**Irina Deeva**
AI Institute
ITMO University
Saint-Petersburg, Russia
iriny.deeva@gmail.com

**Nargiza Amerkhanova**
AI Institute
ITMO University
Saint-Petersburg, Russia
471673@edu.itmo.ru

**Alena Kropacheva**
AI Institute
ITMO University
Saint-Petersburg, Russia
al.kropach@gmail.com

## Abstract

Machine learning models for tabular data often encounter distribution shifts after deployment, yet target OOD samples are frequently unavailable at evaluation time. We propose a principled protocol that leverages aggregate dataset meta-features (MFs) to construct useful proxy OOD tests from in-distribution data. Our approach has two complementary branches: (1) an MFs based splitting procedure that searches for train/test partitions which maximize differences in selected meta-features, and (2) an MFs based synthetic data generator that uses multi-objective evolutionary optimization to produce datasets whose meta-characteristics match a (possibly unavailable) target. Evaluations on real-world source/target pairs of datasets and a diverse set of learners show that MFs based splits create substantially larger distributional differences than random splits and often yield more realistic stress tests; when splits fail to predict true OOD performance, targeted synthetic generation closes the gap. Our results indicate that selected meta-features - especially mutual information, class concentration, and joint entropy - are effective signals of concept shifts and can be used to construct practical pre-deployment OOD evaluations for tabular models.

## 1 Introduction

Machine learning systems deployed in the wild routinely face data that differ from the distributions seen during training. Measuring how model performance degrades under such distribution shift - and doing so before the shifted data are observed - is a central obstacle to building reliable ML systems. In many real-world settings the true out-of-distribution (OOD) target is unavailable at evaluation time; instead, practitioners often have only partial information about the target, for example aggregate statistics or meta-characteristics (meta-features) [24] of the target dataset. This paper asks a practical question that sits at the intersection of robustness and trustworthy evaluation: *can we use meta-features (MFs) to construct test data that meaningfully predict a model's performance on an unseen OOD target?*

Given the prevalence of tabular data in many high-stakes domains, models trained on such data require systematic and rigorous evaluation procedures. OOD evaluation for tabular models exhibits distinct challenges: shifts may affect heterogeneous feature types (continuous, ordinal, categorical) and are often difficult to interpret semantically compared to visual modalities. Consequently, widely used tabular repositories and benchmarks (e.g., UCI[13], OpenML[1], TabArena[4]) typically lack mechanisms for constructing or separating datasets according to well-defined distributional shifts. Specialized resources - such as Tableshift[6], Wild-Tab[14], and TabRed[25] - partially address this gap, but remain scarce and are often limited in the range of tasks they support or in the kinds of shift they represent. These limitations motivate a principled, reproducible OOD evaluation protocol

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Reliable ML from Unreliable Data.

that enables controlled manipulation of dataset characteristics. Meta-characteristics of tabular data offer an effective solution: they are compact descriptors capturing multiple distributional axes and are sensitive to shifts affecting model behavior [15], [3]. Building on these facts, we study two complementary strategies to turn meta-features information into practical OOD evaluations. The first strategy searches for train/test splits of the available in-distribution (ID) data that maximize differences in meta-features, yielding a meta-feature split that acts as a proxy OOD test. The second strategy goes beyond re-splitting and uses an evolutionary algorithm to synthesize datasets whose meta-characteristics match those reported for a target - a path we call OOD intrinsic property characterization.

**Contributions.** (1) We formalize a principled, reproducible protocol that uses meta-characteristics to construct proxy OOD evaluations from ID data, covering both split-based and synthesis-based approaches. (2) We provide a comprehensive empirical study across datasets, models, and competing splitting methods that identifies which meta-features and combinations are most predictive of OOD model performance. (3) We show that - when splits fail - targeted synthetic data generation can close the gap between proxy and true target performance, offering a practical tool for pre-deployment evaluation when only aggregate target information is available.

The code and data can be found in the repository `https://github.com/ITMO-NSS-team/OOD_Tab_Evaluation`.

## 2 Related works

Motivated by distribution shifts that degrade performance in high-stakes tabular applications, this section reviews methods that assess robustness through standardized evaluation, split design, and synthetic stress testing. Yu et al. [27] systematize the field into three strands: (i) testing OOD performance when labeled OOD data are available, (ii) predicting OOD performance from unlabeled data, and (iii) characterizing model-intrinsic properties without access to test data. Following this classification, in this work we focus on the third setting as the most realistic. In such cases, the option exists to either rely on existing ID data or to use synthetic data to generate OOD data.

For assessing robustness under OOD, several pillars have emerged. Initially, it is possible to emphasise the notion of employing non-random splittings as a methodology for the evaluation of the quality of machine learning models in conditions of data shift. For example, researchers partition tabular datasets along time or geography (e.g. house prices by year, taxi trips by state) or along demographic attributes to create pseudo-domains. DomainBed-style protocols also use leave-one-domain-out splits (training on all but one environment) [27], [9]. However, these approaches require prior knowledge of domain divisions or shift variables, which is often unattainable in real-world scenarios. A number of works formulate the task of data splitting as a clustering task [22], [26]; however, this formulation lack interpretability regarding why performance varies. A similar idea is considered in approaches where the subpopulation with the poorest model quality is sought in the test dataset [16].

When it comes to testing models under data shift conditions using synthetic data, the concept of held-out data augmentation is worth considering [21]. This has resulted in the emergence of a new field of research focused on the synthetic transformation of test data, encompassing visual corruptions and perturbations [10], stylization [8], the addition of spurious cues [17]. In semantic segmentation, Loiseau et al. [20] show that diffusion-based generation after fine-tuning on ID data and inpainting of OOD objects yield useful test scenarios; metrics on synthetic data correlate with those on real OOD inputs, supporting the validity of such “virtual testing”. Generative approaches (VAEs, GANs, diffusion models) create new data distributions or stress-test models on controlled shifts. For instance, the Bank Account Fraud (BAF) dataset was synthetically generated to introduce temporal shifts and class imbalance challenging for tabular models [11]. More recently, Puranik et al. [23] proposed TabOOD, which leverages a latent-diffusion model to generate synthetic tabular samples “at the boundaries” of the data manifold. TabOOD creates OOD-like examples (as well as minority/majority samples) to augment training, and reports large improvements in robustness under novel shifts.

It is evident from an analysis of the extant literature that the subject of evaluating the behaviour of machine learning (ML) models in the context of data shift conditions is a matter of considerable pertinence. However, it is noteworthy that there is a paucity of studies that focus specifically on the tabular domain or that seek to investigate this behaviour in an interpretable manner. The present study aims to address these two gaps.

## 3 Meta-features based shifts

### 3.1 Meta-features based splitting

Let a labeled dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$ is ID dataset. Fix test size $k = \lfloor test\_size \cdot N \rfloor$. The goal is to choose a subset $T \subset \{1, ..., N\}$ with $|T| = k$ (the test indices) so that the meta-characteristics of the training set $S = \{1, ..., N\} \setminus T$ and test set $T$ differ according to user-specified criteria. Define a meta-feature extractor $M(\cdot)$ that maps a sample $A$ to a vector of meta-features values:

$$M(A) = (m_1(A), m_2(A), ..., m_p(A)). \tag{1}$$

For each meta-feature $m_j$ we define a directed distance:

$$d_j(S,T) = \begin{cases} \dfrac{m_j(S)}{m_j(T)}, & \text{if } m_j(S) \neq 0 \text{ and } m_j(T) \neq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The selection of this ratio was driven by the objective of enhancing the manageability of the splitting process (for instance, it is desirable for the meta-characteristic to invariably exceed its value on the train in comparison to that on the test). We also define a scalar imbalance measure for a sample $A$ to obtain approximately balanced splits by class:

$$imb(A) = \frac{min_c n_c(A)}{max_c n_c(A)}, \tag{3}$$

where $n_c(A)$ is the class count of label $c$ in $A$. The imbalance objective will be then:

$$o_{imb}(S,T) = |imb(S) - imb(T) + \lambda(1 - min(imb(S), imb(T))), \tag{4}$$

with $\lambda > 0$ to penalize very imbalanced splits. We therefore obtain a multi-objective fitness vector:

$$S = \{1, \dots, N\} \setminus T, \qquad \mathbf{f}(T) = \big(d_1(S,T),\ d_2(S,T),\ \dots,\ d_p(S,T),\ o_{\text{imb}}(S,T)\big). \tag{5}$$

The search seeks Pareto-optimal $T$ that maximize the first $p$ objectives and minimize the last objective. The decision was taken to employ evolutionary algorithms as the optimisation algorithm, primarily due to the fact that not all meta-characteristics can be differentiated. The evolutionary algorithm considers the indices of the test dataset as a population individual. We employ NSGA-II for selection, which maintains a Pareto-optimal front of solutions [2]. The algorithm's overall structure corresponds to the standard steps of an evolutionary algorithm (see pseudocode 1). The mutation function replaces selected indices with unused indices from the dataset, while the crossover function exchanges unique indices between parents while preserving duplicates (see details in Appendix A, pseudocode 3). In addition to the utilisation of random generation algorithms, algorithms based on an ordered dataset were incorporated as population generation algorithms (see details in Appendix A, pseudocode 2). This modification resulted in enhanced outcomes for specific meta-characteristics. For instance, it is evident that in order to divide the sample into two parts that differ significantly in terms of their mean, it is necessary to first sort the data and then divide it. This is the reason why sorting exerts a beneficial effect on the performance of the algorithm.

### 3.2 Meta-features based generation

The second strategy involves the generation of a synthetic OOD dataset, in which the meta-features will approximate the actual values of the meta-features in the real OOD dataset. Define target meta-values $m_j^* \in \mathbb{R}^{q_j}$ (some meta-features are vectors). For a candidate synthetic dataset $S'$:

$$M(S') = (m_1(S'), m_2(S'), ..., m_p(S')), l_j(S') = ||m_j(S') - m_j^*||_2. \tag{6}$$

Multi-objective fitness (to minimize) will be then:

$$\mathbf{f}(S') = (l_1(S'), l_2(S'), ..., l_p(S')). \tag{7}$$

Evolutionary algorithms were also selected for the purpose of generating synthetic data that would minimise such fitness (Eq. 7). In the context of literature, evolutionary algorithms have not been widely employed for the generation of synthetic tabular data. However, there are notable exceptions, particularly within the domain of obtaining synthteic data with a specified level of differential privacy [19]. In this particular instance, the individual population will constitute the dataset of size $N$ itself. The general generation algorithm can be delineated as follows:

**Algorithm 1** Main: MFs based splitting

```
Require: Dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, test size $k$, meta-features $\mathcal{M}$, population size $P$, generations $G$
Ensure: Pareto front of candidate splits
 1: Population ← $\emptyset$
 2: for $i \leftarrow 1$ to $P$ do
 3:     Population[$i$] ← CreateIndividualByOrdering($D, k$)
 4:     Population[$i$].fitness ← Evaluate(Population[$i$], $D, \mathcal{M}$)
 5: end for
 6: HallOfFame ← $\emptyset$
 7: for $g \leftarrow 1$ to $G$ do
 8:     Offspring ← VarAnd(Population)                    ▷ apply crossover+mutation
 9:     for all child ∈ Offspring do
10:         if not child.fitness.valid then
11:             child.fitness ← Evaluate(child, $D, \mathcal{M}$)
12:         end if
13:     end for
14:     Population ← selNSGA2(Population ∪ Offspring, $P$) [5]
15:     HallOfFame.update(Population)
16:     record statistics (avg/min/max per objective)
17: end for
18: ParetoFront ← extract non-dominated solutions from Population
        return {ParetoFront, HallOfFame, Population}
```

1. **Population generation / initialization**: sample a batch from a generative prior $G$, where $S' \leftarrow G(batch = N)$. The SOTA model for tabular data, the Forest Diffusion model [12], was selected as the generative model.
2. **Crossover**: with probability *row_mode_prob* choose row-mode or column-mode; exchange $\approx 30\%$ randomly chosen rows (row-mode) or columns (column-mode) between parents (see details in Appendix A, pseudocode 7).
3. **Mutations**:
   - Add Gaussian noise to selected rows and continuous columns; scale noise per-feature (see details in Appendix A, pseudocode 4).
   - Replace selected rows by sampling continuous values from a fitted Gaussian Mixture Model and categorical by empirical category probabilities (see details in Appendix A, pseudocode 5).
   - Draw new continuous rows from multivariate normal with empirical covariance; preserve categorical by resampling (see details in Appendix A, pseudocode 6).
4. **Selection**: multi-objective selection `selNSGA3WithMemory` [5].

### 3.3 Proposed meta-features

The present study puts forward a series of five meta-features, selected via systematic sensitivity analysis to distributional shifts (see details in Appendix B), validated across synthetic scenarios (see Appendix B, Figure 3) showing consistent sensitivity to covariate and concept shifts. The Python meta-feature extractor (PyMFE) library is utilized [24], which provides standardized implementations of meta-learning features. It is important to note that ordinal characteristics (minimum, maximum) were not included in the final list of meta-characteristics, despite demonstrating a higher degree of responsiveness to shifts in the data compared to other characteristics. These characteristics were found to be too sensitive to changes in the data, which rendered optimization a challenging process.

**Information-theoretic meta-features:**

- **Mutual Information (mut_inf)** measures statistical dependence between features and the target variable. The mutual information vector is computed as

$$\mathbf{I} = [I(X_1; Y), I(X_2; Y), \ldots, I(X_d; Y)],$$

$$\text{where } I(X_i;Y) = \sum_{x_i \in X_i} \sum_{y \in Y} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)},$$

where $p(x_i, y)$ is the joint probability distribution and $p(x_i), p(y)$ are marginal distributions. For continuous variables, kernel density estimation is used to estimate probability densities.

- **Joint Entropy (joint_ent)** measures the joint entropy between features and the target variable. The joint entropy vector is computed as

$$\mathbf{H}_{joint} = [H(X_1, Y), H(X_2, Y), \ldots, H(X_d, Y)],$$
$$\text{where } H(X_i, Y) = -\sum_{x_i, y} p(x_i, y) \log p(x_i, y),$$

  where $p(x_i, y)$ is the joint probability distribution of feature $X_i$ and target $Y$.

- **Attribute Entropy (attr_ent)** measures the uncertainty of individual features. The entropy vector is computed as

$$\mathbf{H}_{attr} = [H(X_1), H(X_2), \ldots, H(X_d)],$$
$$\text{where } H(X_i) = -\sum_{x_i} p(x_i) \log p(x_i),$$

  where $p(x_i)$ is the probability of feature $X_i$ taking value $x_i$.

- **Class Concentration (class_conc)** measures the concentration coefficient for features with respect to class separation. The concentration vector is computed as

$$\mathbf{C} = [\text{conc}(X_1, Y), \text{conc}(X_2, Y), \ldots, \text{conc}(X_d, Y)],$$
$$\text{where } \text{conc}(X_i, Y) = \frac{\sum_j \frac{p_{ij}^2}{p_{\cdot j}} - \sum_i p_{i\cdot}^2}{1 - \sum_i p_{i\cdot}^2},$$

  where $p_{ij}$ is the joint probability of feature value $i$ and class $j$, $p_{\cdot j} = \sum_i p_{ij}$ is the marginal probability of class $j$, and $p_{i\cdot} = \sum_j p_{ij}$ is the marginal probability of feature value $i$.

**Statistical meta-features:**

- **Interquartile Range (iq_range)** measures distributional spread robustness for features. The IQR vector is computed as

$$\mathbf{IQR} = [Q_{75}^1 - Q_{25}^1, Q_{75}^2 - Q_{25}^2, \ldots, Q_{75}^d - Q_{25}^d],$$

  where $Q_{25}^i$ and $Q_{75}^i$ are the 25th and 75th percentiles of feature $X_i$, respectively.

Summarized meta-features are employed in this work, computed as the **means** of the individual feature vectors described above. This summarization approach reduces dimensionality while preserving the essential distributional characteristics captured by each meta-feature type.

## 4 Experimental setup

**Datasets.** We run the experiments on five source/target tabular datasets provided in our repository: `electricity`, `taxi`, `income`, `california`, and `acs_accidents`. The `taxi` and `acs_accidents` datasets originally belong to the WHYSHIFT benchmark [18], so their ID and OOD subsamples are known. The remaining three datasets were obtained from the OpenML open source repository [1], and their data shifts were modelled by dividing them according to a specific variable (for example, for `income`, this is the "gender" variable). Table 1 summarizes the key characteristics of our evaluation datasets.

**Table 1:** Dataset characteristics

| Dataset | Features | Source samples | Target samples | Classes |
|---|---|---|---|---|
| electricity | 7 | 9,987 | 10,015 | 2 |
| taxi | 8 | 10,001 | 10,001 | 2 |
| income | 13 | 20,381 | 9,783 | 2 |
| california | 8 | 10,316 | 10,320 | 2 |
| acs_accidents | 46 | 22,654 | 3,956 | 2 |

**MFs based splits (`mfs_split`).** We run multi-objective evolutionary optimization over test-index sets (test fraction 0.2) to find train/test splits maximizing meta-feature differences while maintaining class balance. Each experimental configuration repeated 5 times with different random seeds for statistical reliability. We use population size 50, 200 generations, and `selNSGA2` selection. Crossover (probability 0.7) exchanges up to 1/4 of unique indices between parents; mutation (probability 0.2) replaces indices with unused ones (Appendix A, pseudocode 3). The optimizer yields multiple solutions saved as train/test files. Models are trained on MFs based splits and evaluated on corresponding test sets using F1-score. Random splits serve as baseline.

**MFs based synthetic OOD (`mfs_synthetic`).** Given a source/target pair, we compute target meta-feature values and generate synthetic datasets matching these targets via multi-objective optimization (`selNSGA3WithMemory`). We employ population size 100, 200 generations, crossover (probability 0.5), and mutation (probability 0.16). The mutation strategy combines distribution-based sampling, Gaussian noise addition, and covariance-based generation. Experiments are repeated 5 times to account for stochastic variation. Synthetic datasets test models trained on source data, enabling OOD evaluation without target access.

**Models and evaluation.** We validate the protocol across standard learners (Logistic Regression, XGBoost) and robustness-oriented architectures (IRM, AdversarialDRO) with fixed hyperparameters and unified training/evaluation, without tuning. Logistic Regression: `max_iter=1000`, `class_weight='balanced'`, `n_jobs=-1`. XGBoost: `n_estimators=100`, `eval_metric='logloss'`, `random_state=42`. IRM trains a two-layer MLP (`hidden_size=256`, `dropout=0.1`) for 1000 iterations with invariance penalty $\lambda = 1.0$, 100 annealing iterations, Adam (lr $= 10^{-3}$), and adaptive batch size $\min(128, \text{dataset\_size})$. AdversarialDRO: MLP trained for 10 epochs, `hidden=32`, `batch_size=64`, Adam (lr $= 0.01$), adversarial updates over label groups with $\eta\_\pi = 0.1$, $r = 0.1$, `clip_max=2.0`, $\varepsilon = 0.001$, $\beta = 0.999$ (`dropout=0.0`, `weight_decay=0.0`). Models are trained on MFs based splits or source data and evaluated on MF-split tests, real targets, or synthetic OOD datasets. Primary metric: F1-score; we report mean and variability across runs, using random splits as baseline.

## 5 Experimental results

### 5.1 Effectiveness of MFs based splits

Our primary research question asks whether meta-feature optimization can create more meaningful OOD tests than random splits. Table 2 provides direct evidence that MF_split consistently produces stronger distributional differences than random_split across most datasets and meta-feature dimensions. The bold entries in Table 2 highlight the most diverse meta-feature ratios from the neutral 1.0. By constructing a compressed representation of the data using PCA, a clear comparison was made of how the train and test sets were distributed for different splitting (figure 1). The findings unequivocally demonstrate that, in contrast to `mfs_split` splitting approach, the data exhibits minimal variation when subjected to random split.

**Table 2:** Ratios of meta-features between train and test

| Dataset | Split type | Meta-feature | | | | |
|---|---|---|---|---|---|---|
| | | attr_ent | class_conc | mut_inf | iq_range | joint_ent |
| electricity | MF_split | **1.600**±0.00 | **1.909**±0.53 | **2.743**±0.74 | **1.754**±0.09 | **1.400**±0.00 |
| | random_split | 1.090±0.03 | 0.730±0.01 | 0.990±0.00 | 1.000±0.01 | 1.070±0.02 |
| income | MF_split | **1.367**±0.04 | **2.134**±0.36 | **2.147**±0.39 | **1.872**±0.10 | **1.233**±0.02 |
| | random_split | 1.050±0.00 | 0.940±0.00 | 1.010±0.00 | 1.000±0.01 | 1.030±0.01 |
| taxi | MF_split | **1.291**±0.01 | **1.300**±0.53 | **1.160**±0.08 | **1.612**±0.04 | **1.213**±0.01 |
| | random_split | 1.090±0.00 | 0.710±0.00 | 0.980±0.00 | 1.000±0.01 | 1.070±0.00 |
| california | MF_split | **1.256**±0.02 | 1.264±0.21 | **1.583**±0.55 | **1.803**±0.00 | **1.200**±0.00 |
| | random_split | 0.900±0.00 | **0.730**±0.00 | 1.010±0.00 | 0.990±0.00 | 1.100±0.00 |
| acs_accidents | MF_split | **1.369**±0.12 | **1.359**±0.29 | **1.004**±0.03 | **1.923**±0.04 | **1.291**±0.06 |
| | random_split | 1.080±0.01 | 0.980±0.00 | 1.000±0.00 | 0.980±0.02 | 1.060±0.01 |

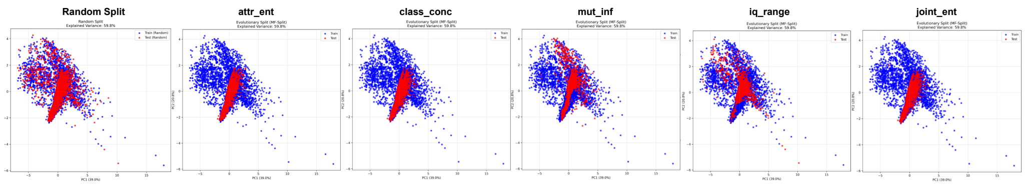


**Figure 1:** Comparison of training and test sets for the `electricity` dataset under random split and MF-split across different meta-features. The blue dots are the train, the red dots are the test.

Table 3 presents comprehensive F1-scores across all datasets, models, and meta-feature splitting criteria. For in-distribution (ID) evaluation, both training and test subsets are sampled from the source distribution using various splitting strategies. For out-of-distribution (OOD) evaluation, models are trained on the source distribution and tested on the target distribution. The colored percentages in parentheses indicate the performance difference between each model's result and the corresponding target value (on real OOD). Green values denote cases where the model performance exceeded the target (positive difference), while red values indicate performance below the target (negative difference). Bold black values show worst degradations per model-dataset pair; colored bold values indicate closest matches to target performance. The results demonstrate systematic patterns of performance degradation under MFs based splits compared to random baselines, validating our hypothesis that evolutionary optimization of meta-feature differences creates more challenging and realistic OOD test conditions.

**Table 3:** F1-scores by models across datasets and split criteria

| Metric | Dataset | LR | XGB | IRM | DRO |
|---|---|---|---|---|---|
| Random Split (ID) | electricity | 0.798 ± 0.00 (20%) | 0.832 ± 0.00 (20%) | 0.813 ± 0.01 (16%) | 0.814 ± 0.02 (17%) |
| | taxi | 0.752 ± 0.01 (6%) | 0.778 ± 0.01 (12%) | 0.790 ± 0.02 (15%) | 0.712 ± 0.02 (6%) |
| | income | 0.678 ± 0.00 (38%) | 0.716 ± 0.01 (23%) | 0.618 ± 0.02 (38%) | 0.514 ± 0.04 (10%) |
| | california | 0.823 ± 0.01 (8%) | 0.869 ± 0.01 (13%) | **0.693** ± 0.08 (10%) | 0.821 ± 0.01 (12%) |
| | acs_accidents | 0.719 ± 0.00 (16%) | 0.863 ± 0.00 (13%) | 0.867 ± 0.01 (45%) | 0.702 ± 0.07 (22%) |
| Attr_ent (ID) | electricity | 0.813 ± 0.00 (22%) | 0.830 ± 0.00 (20%) | 0.824 ± 0.01 (17%) | 0.811 ± 0.00 (16%) |
| | taxi | 0.746 ± 0.00 (6%) | 0.763 ± 0.00 (11%) | 0.894 ± 0.01 (26%) | 0.724 ± 0.01 (7%) |
| | income | 0.628 ± 0.00 (33%) | 0.612 ± 0.02 (12%) | 0.588 ± 0.06 (35%) | **0.381** ± 0.04 (4%) |
| | california | 0.834 ± 0.02 (9%) | 0.880 ± 0.01 (14%) | 0.953 ± 0.01 (16%) | 0.857 ± 0.03 (15%) |
| | acs_accidents | 0.542 ± 0.10 (2%) | 0.749 ± 0.05 (2%) | 0.805 ± 0.12 (39%) | 0.680 ± 0.01 (20%) |
| Joint_ent (ID) | electricity | 0.828 ± 0.01 (23%) | 0.841 ± 0.00 (21%) | 0.901 ± 0.01 (25%) | 0.832 ± 0.00 (19%) |
| | taxi | 0.751 ± 0.01 (6%) | 0.772 ± 0.01 (12%) | 0.851 ± 0.04 (21%) | 0.729 ± 0.01 (8%) |
| | income | 0.626 ± 0.01 (33%) | **0.605** ± 0.03 **(12%)** | **0.535** ± 0.08 **(30%)** | 0.392 ± 0.01 (3%) |
| | california | 0.877 ± 0.01 (13%) | 0.891 ± 0.01 (15%) | 0.973 ± 0.01 (18%) | 0.879 ± 0.01 (17%) |
| | acs_accidents | **0.461** ± 0.05 (10%) | **0.725** ± 0.03 **(0%)** | **0.716** ± 0.07 **(30%)** | **0.630** ± 0.05 **(15%)** |
| Mut_inf (ID) | electricity | **0.735** ± 0.02 **(14%)** | **0.749** ± 0.01 **(12%)** | **0.795** ± 0.02 **(14%)** | **0.766** ± 0.01 **(12%)** |
| | taxi | 0.723 ± 0.01 **(4%)** | 0.754 ± 0.01 (10%) | 0.899 ± 0.01 (26%) | 0.696 ± 0.01 **(4%)** |
| | income | **0.600** ± 0.02 **(30%)** | 0.617 ± 0.02 (13%) | 0.687 ± 0.03 (45%) | 0.405 ± 0.02 (1%) |
| | california | 0.789 ± 0.02 (5%) | 0.866 ± 0.04 (13%) | 0.837 ± 0.05 **(4%)** | **0.786** ± 0.02 **(8%)** |
| | acs_accidents | 0.718 ± 0.01 (16%) | 0.863 ± 0.00 (13%) | 0.893 ± 0.01 (48%) | 0.714 ± 0.01 (23%) |
| Class_conc (ID) | electricity | 0.736 ± 0.01 (14%) | 0.772 ± 0.01 (14%) | 0.842 ± 0.03 (19%) | 0.783 ± 0.01 (14%) |
| | taxi | **0.526** ± 0.10 (16%) | **0.592** ± 0.07 **(6%)** | **0.773** ± 0.10 **(14%)** | **0.505** ± 0.10 (15%) |
| | income | 0.622 ± 0.01 (32%) | 0.638 ± 0.00 (15%) | 0.670 ± 0.01 (43%) | 0.427 ± 0.01 **(1%)** |
| | california | **0.776** ± 0.01 **(3%)** | **0.831** ± 0.01 **(9%)** | 0.927 ± 0.03 (13%) | 0.815 ± 0.01 (11%) |
| | acs_accidents | 0.565 ± 0.06 **(0%)** | 0.785 ± 0.03 (6%) | 0.787 ± 0.06 (37%) | 0.719 ± 0.04 (24%) |
| IQ_range (ID) | electricity | 0.783 ± 0.01 (19%) | 0.806 ± 0.01 (17%) | 0.861 ± 0.02 (21%) | 0.807 ± 0.01 (16%) |
| | taxi | 0.741 ± 0.01 (5%) | 0.771 ± 0.00 (12%) | 0.948 ± 0.10 (31%) | 0.721 ± 0.01 (7%) |
| | income | 0.653 ± 0.00 (36%) | 0.685 ± 0.02 (20%) | 0.698 ± 0.02 (46%) | 0.474 ± 0.02 (6%) |
| | california | 0.853 ± 0.00 (11%) | 0.875 ± 0.00 (14%) | 0.876 ± 0.01 (8%) | 0.871 ± 0.01 (17%) |
| | acs_accidents | 0.671 ± 0.00 (11%) | 0.844 ± 0.00 (11%) | 0.872 ± 0.00 (46%) | 0.766 ± 0.03 (29%) |
| Target (real OOD) | electricity | 0.596 | 0.633 | 0.655 | 0.646 |
| | taxi | 0.687 | 0.655 | 0.637 | 0.654 |
| | income | 0.2974 | 0.488 | 0.240 | 0.418 |
| | california | 0.742 | 0.738 | 0.795 | 0.705 |
| | acs_accidents | 0.563 | 0.730 | 0.413 | 0.479 |

The initial observation is that the splitting of data according to characteristics such as *mut_info*, *class_conc*, and *joint_ent* frequently results in alterations in the quality of machine learning models.

This phenomenon can be attributed to the fact that these meta-features are indicative of concept shift, which is a more prevalent cause of alterations in the quality of machine learning models than covariance shift. That is why splitting by *iq_range* and *attr_ent* does not significantly change the quality of machine learning models. It is evident that all machine learning models are susceptible to the effects of splitting, even those that are considered robust. This assertion is corroborated by the conclusions drawn by the authors [7], which posit that robust models exhibit a comparable degree of quality to ERM models in the context of substantial shifts in data. It is noteworthy that although `mfs_split` generates complex splitting for machine learning models and typically compromises their quality, under certain circumstances it can be employed to enhance the quality of machine learning models in comparison to `random_split`. This phenomenon occurs when splitting by *attr_ent* meta-feature. This discrepancy may be attributed to the observation that the training sample encompasses a greater variety of observation features, characterised by higher entropy, while the test sample exhibits a higher degree of homogeneity, that is, simplicity. However, it is challenging to ascertain this with certainty, as the entropy of each predictor in the dataset differs between the train and test sets (due to the aggregation of the meta-feature vector we can't see it, see the section 3.3 for further details). This deficiency can be identified as a method's inherent shortcoming.

To contextualize the effectiveness of our MFs based split, Table 4 compares it against the MMD based clustering method proposed by Napoli and White [22]. For the Best MF row, we report the worst-case performance across all meta-features from Table 3 (indicated by bold values in that table). The percentages in parentheses show the closest gap to real OOD performance achieved by any meta-feature for each model-dataset combination. For the MMD row, we run the algorithm 5 times with different random seeds, each producing an 80%-20% train-test split. We report mean $\pm$ std across runs, with target gaps in parentheses.

**Table 4:** Performance comparison: MFs based vs. MMD based split

| Split Method | Dataset | LR | XGB | IRM | DRO |
|---|---|---|---|---|---|
| Best MF | electricity | 0.735 $\pm$ 0.02 **(14%)** | 0.749 $\pm$ 0.01 **(12%)** | 0.795 $\pm$ 0.02 **(14%)** | 0.766 $\pm$ 0.01 **(12%)** |
| | taxi | **0.526** $\pm$ 0.10 (4%) | 0.592 $\pm$ 0.07 **(6%)** | 0.773 $\pm$ 0.10 (14%) | 0.505 $\pm$ 0.10 **(4%)** |
| | income | **0.600** $\pm$ 0.02 **(30%)** | **0.605** $\pm$ 0.03 **(12%)** | **0.535** $\pm$ 0.08 **(30%)** | **0.381** $\pm$ 0.04 **(1%)** |
| | california | **0.776** $\pm$ 0.01 **(3%)** | 0.831 $\pm$ 0.01 (9%) | **0.837** $\pm$ 0.05 **(4%)** | **0.786** $\pm$ 0.02 **(8%)** |
| | acs_accidents | 0.461 $\pm$ 0.05 **(0%)** | 0.725 $\pm$ 0.03 **(0%)** | 0.716 $\pm$ 0.07 (30%) | 0.630 $\pm$ 0.05 (15%) |
| MMD | electricity | **0.419** $\pm$ 0.00 (18%) | **0.335** $\pm$ 0.02 (30%) | **0.435** $\pm$ 0.06 (22%) | **0.355** $\pm$ 0.07 (29%) |
| | taxi | 0.703 $\pm$ 0.00 **(2%)** | **0.564** $\pm$ 0.02 (9%) | **0.690** $\pm$ 0.01 **(5%)** | **0.448** $\pm$ 0.09 (21%) |
| | income | 0.629 $\pm$ 0.00 (33%) | 0.622 $\pm$ 0.01 (13%) | 0.644 $\pm$ 0.02 (40%) | 0.650 $\pm$ 0.01 (23%) |
| | california | 0.828 $\pm$ 0.00 (9%) | **0.829** $\pm$ 0.02 **(9%)** | 0.890 $\pm$ 0.02 (10%) | 0.855 $\pm$ 0.01 (15%) |
| | acs_accidents | **0.459** $\pm$ 0.00 (10%) | **0.673** $\pm$ 0.01 (6%) | **0.583** $\pm$ 0.11 **(17%)** | **0.468** $\pm$ 0.12 **(1%)** |

This evaluation demonstrates dataset-specific behavior for both methods. For electricity, MFs based split achieves gaps of 12-14% from target values, while MMD based split shows deviations of 18-30%. On taxi, MMD based split achieves tighter approximations (2-5% gaps) for most models. Both methods show substantial deviations on income (30-40%), indicating severe real-world shift. A key distinction is interpretability: MFs based split provides explicit control over distributional characteristics, while MMD uses an aggregate kernel-based distance.

## 5.2 Synthetic OOD generation effectiveness

In light of the findings from the preceding experiment, it is possible to determine certain datasets for which `mfs_split` has a significant impact on the quality of machine learning models, yet does not result in a substantial advancement in the evaluation of the actual target. The present experiment was established with the specific objective of generating synthetic OOD data for such datasets. The table 3 presents the evaluation results on synthetic OOD data for two such datasets (more detailed information on the results of synthetic OOD data generation is provided in the Appendix C). It is evident that these evaluations demonstrate a strong correlation with the actual quality of the OOD dataset. In this experiment, the number of meta-features utilised for generation was restricted, as an excessive number of meta-features resulted in suboptimal convergence. This example demonstrates the applicability of synthetic data as OOD test data. However, it may be beneficial to explore more advanced generative models for future data generation.

**Table 5:** F1-scores for models trained on original data and tested on synthetic data

| Dataset | Meta-features | LR | XGB | DRO | IRM |
|---|---|---|---|---|---|
| electricity | mut_inf, class_conc, iq_range | 0.613 ± 0.08 | 0.641 ± 0.09 | 0.587 ± 0.08 | 0.613 ± 0.08 |
| | mut_inf, class_conc | 0.611 ± 0.01 | 0.625 ± 0.01 | 0.589 ± 0.01 | 0.632 ± 0.02 |
| california | mut_inf, class_conc, iq_range | 0.636 ± 0.05 | 0.692 ± 0.02 | 0.661 ± 0.02 | 0.561 ± 0.11 |
| | mut_inf, class_conc | 0.679 ± 0.07 | 0.713 ± 0.03 | 0.628 ± 0.10 | 0.682 ± 0.05 |

## 6 Conclusion and Discussion

We introduced a practical protocol that turns dataset-level meta-features into actionable proxy OOD evaluations for tabular data, via two complementary strategies: evolutionary meta-features based splitting and meta-features based synthetic data generation. Across five real-world datasets and a set of standard and robustness-oriented learners, MFs based splits produce consistently larger distributional differences than random splits and often generate more challenging - and more realistic-stress tests for model performance. When splitting alone does not align with the true target degradation, targeted synthetic generation can close the gap, demonstrating that meta-features are useful signals for constructing pre-deployment OOD tests. Our empirical analysis identifies a small set of meta-features (notably mutual information, class concentration and joint entropy) that are particularly indicative of concept shifts and predictive of model degradation; other meta-features (e.g., simple attribute entropy or IQR) are less consistently informative for the kinds of concept shifts studied. We also highlight several limitations. First, optimization and generation are sensitive to the number and choice of meta-features: using too many objectives degrades convergence, and aggregating meta-feature vectors can hide per-predictor effects. Second, our synthetic generation relies on a set of relatively simple mutation/crossover operators and a specific generative prior; more powerful generative models (e.g., diffusion-based or conditional tabular generators) may improve fidelity. Looking forward, promising directions include: (i) integrating stronger conditional generative models to improve synthetic OOD fidelity; (ii) devising automated meta-feature selection strategies that balance informativeness and optimization tractability; (iii) extending the protocol to richer shift taxonomies (e.g., subtle covariate shifts, label noise, or compound shifts) and multi-class problems; and (iv) exploring light-weight approximations of the evolutionary search to reduce computational cost for practitioners.

## 7 Acknowledgments

This work supported by the Ministry of Economic Development of the Russian Federation (IGK 000000C313925P4C0002), agreement No139-15-2025-010.

## References

[1] Bernd Bischl, Giuseppe Casalicchio, Taniya Das, Matthias Feurer, Sebastian Fischer, Pieter Gijsbers, Subhaditya Mukherjee, Andreas C Müller, László Németh, Luis Oala, et al. Openml: Insights from 10 years and more than a thousand papers. *Patterns*, 2025.

[2] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *International conference on parallel problem solving from nature*, pages 849–858. Springer, 2000.

[3] Irina Deeva and Alena Kropacheva. To select or not to select? the role of meta-features selection in meta-learning tasks with tabular data. In *International Conference on Computational Science*, pages 294–308. Springer, 2025.

[4] Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. Tabarena: A living benchmark for machine learning on tabular data. *arXiv preprint arXiv:2506.16791*, 2025.

[5] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

[6] Josh Gardner, Zoran Popovic, and Ludwig Schmidt. Benchmarking distribution shift in tabular data with tableshift. *Advances in Neural Information Processing Systems*, 36:53385–53432, 2023.

[7] Josh Gardner, Zoran Popovic, and Ludwig Schmidt. Benchmarking distribution shift in tabular data with tableshift. *Advances in Neural Information Processing Systems*, 2023.

[8] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *International conference on learning representations*, 2018.

[9] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization, 2020.

[10] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[11] Sérgio Jesus, José Pombal, Duarte Alves, André Cruz, Pedro Saleiro, Rita Ribeiro, João Gama, and Pedro Bizarro. Turning the tables: Biased, imbalanced, dynamic tabular datasets for ml evaluation. *Advances in Neural Information Processing Systems*, 35:33563–33575, 2022.

[12] Alexia Jolicoeur-Martineau, Kilian Fatras, and Tal Kachman. Generating and imputing tabular data via diffusion and flow-based gradient-boosted trees, 2023.

[13] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The uci machine learning repository. `https://archive.ics.uci.edu`.

[14] Sergey Kolesnikov. Wild-tab: A benchmark for out-of-distribution generalization in tabular regression. *arXiv preprint arXiv:2312.01792*, 2023.

[15] Joanna Komorniczak and Paweł Ksieniewicz. On metafeatures' ability of implicit concept identification. *Machine Learning*, 113(10):7931–7966, 2024.

[16] Mike Li, Hongseok Namkoong, and Shangzhou Xia. Evaluating model performance under worst-case subpopulations, 2024.

[17] Zhiheng Li, Ivan Evtimov, Albert Gordo, Caner Hazirbas, Tal Hassner, Cristian Canton Ferrer, Chenliang Xu, and Mark Ibrahim. A whac-a-mole dilemma: Shortcuts come in multiples where mitigating one amplifies others. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20071–20082, 2023.

[18] Jiashuo Liu, Tianyu Wang, Peng Cui, and Hongseok Namkoong. On the need for a language describing distribution shifts: Illustrations on tabular datasets. *Advances in Neural Information Processing Systems*, 36:51371–51408, 2023.

[19] Terrance Liu, Jingwu Tang, Giuseppe Vietri, and Steven Wu. Generating private synthetic data with genetic algorithms. In *International Conference on Machine Learning*, pages 22009–22027. PMLR, 2023.

[20] Thibaut Loiseau, Tuan-Hung Vu, Mickael Chen, Patrick Pérez, and Matthieu Cord. Reliability in semantic segmentation: Can we use synthetic data?, 2024.

[21] Wang Lu, Wang Wang, Jindong Yidong, and Xing Xie. Towards optimization and model selection for domain generalization: A mixup-guided solution. In *The KDD'23 Workshop on Causal Discovery, Prediction and Decision*, pages 75–97. PMLR, 2023.

[22] Andrea Napoli and Paul White. Clustering-based validation splits for model selection under domain shift, 2025.

[23] Bhagyashree Puranik, Bugra Can, and Yi Fan. Tabular out-of-distribution data synthesis for enhancing robustness. 2024.

[24] Adriano Rivolli, Luís PF Garcia, Carlos Soares, Joaquin Vanschoren, and André CPLF de Carvalho. Meta-features for meta-learning. *Knowledge-Based Systems*, 240:108101, 2022.

[25] Ivan Rubachev, Nikolay Kartashev, Yury Gorishniy, and Artem Babenko. Tabred: Analyzing pitfalls and filling the gaps in tabular deep learning benchmarks. *arXiv preprint arXiv:2406.19380*, 2024.

[26] Hanna Wecker, Annemarie Friedrich, and Heike Adel. Clusterdatasplit: Exploring challenging clustering-based data splits for model performance evaluation. In *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, pages 155–163, 2020.

[27] Han Yu, Jiashuo Liu, Xingxuan Zhang, Jiayun Wu, and Peng Cui. A survey on evaluation of out-of-distribution generalization, 2024.

# A Implementation details of the proposed algorithms

## A.1 MFs based splitting

The following pseudocodes provide a comprehensive overview of the algorithmic components employed for the purpose of data segmentation based on meta-characteristics.

**Algorithm 2** Create Individual By Ordering

```
Require: Dataset $D$ (feature matrix $X$), test size $k$
Ensure: Individual: list of $k$ distinct indices (test set)
 1: $n \leftarrow$ number of samples in $X$
 2: choose random column index $j \sim$ Uniform$\{1, \dots, d\}$
 3: $S_{\text{sorted}} \leftarrow$ argsort($X[:, j]$)                    ▷ indices sorted by feature $j$
 4: choose pattern $\in$ {first_k, last_k, contiguous, every_other, random}
 5: if pattern = first_k then
 6:     test_indices $\leftarrow S_{\text{sorted}}[1..k]$
 7: else if pattern = last_k then
 8:     test_indices $\leftarrow S_{\text{sorted}}[n - k + 1..n]$
 9: else if pattern = contiguous then
10:     start $\leftarrow$ random integer in $[1, n - k + 1]$
11:     test_indices $\leftarrow S_{\text{sorted}}$[start..start $+ k - 1$]
12: else if pattern = every_other then
13:     choose start $\in \{1, 2\}$, candidate $\leftarrow S_{\text{sorted}}$[start :: 2]
14:     test_indices $\leftarrow$ first $k$ of candidate (fallback to random if too short)
15: else
16:     test_indices $\leftarrow$ uniform random sample of $k$ distinct indices from $\{1, \dots, n\}$
17: end if
18: return Individual(test_indices)
```

**Algorithm 3** Crossover and Mutation Operators (variation)

```
 1: function CROSSOVER(parent$_1$, parent$_2$)
 2:     $A \leftarrow$ set(parent$_1$); $B \leftarrow$ set(parent$_2$)
 3:     only_1 $\leftarrow A \setminus B$; only_2 $\leftarrow B \setminus A$
 4:     if only_1 $\neq \emptyset$ and only_2 $\neq \emptyset$ then
 5:         $m \leftarrow \min(|$only_1$|, |$only_2$|, \lfloor |parent|/4 \rfloor)$
 6:         $r \leftarrow$ uniform integer in $[1, m]$
 7:         pick $r$ indices from only_1 and $r$ from only_2
 8:         swap selected indices between parents preserving position uniqueness
 9:     end if
10:     return child1, child2
11: end function
12: function MUTATE(individual, indpb)
13:     available $\leftarrow \{1, \dots, N\} \setminus$ set(individual)
14:     if available $= \emptyset$ or rand() $>$ indpb then
15:         return individual
16:     end if
17:     $n_{\text{mut}} \leftarrow$ random integer in $[1, \min(3, |$individual$|)]$
18:     select $n_{\text{mut}}$ random positions in individual
19:     select $n_{\text{mut}}$ indices from available
20:     for each selected position $p$ and new index $i$ do
21:         individual$[p] \leftarrow i$
22:     end for
23:     return individual
24: end function
```

### A.2 MFs based generation

The following pseudocodes detail the mutation and crossover operators for generating synthetic tabular data with specified meta-characteristics. The implementation of these operators has the following characteristics:

- All mutation functions operate on a row-wise basis, considering each data point (row) independently;
- The functions handle both continuous and categorical features appropriately;
- Continuous features are modified using Gaussian noise or distribution sampling (4, 5);
- Categorical features are modified by sampling from probability distributions (6);
- The crossover function can operate on either rows or columns with configurable probabilities (7);
- All functions ensure data validity by handling NaN values and clipping categorical values to valid ranges.

**Algorithm 4** Mutate Noise

```
1: function MUTATE_NOISE(individual, mutation_prob, noise_scale, categorical_idx, continuous_idx, cat_probs, n_features)
2:     individual_data ← reshape individual to 2D array with $n\_features$ columns
3:     mutated ← copy of individual_data
4:     for each row $i$ in individual_data do
5:         if random value < mutation_prob then
6:             if continuous_idx exists then
7:                 noise_scale_adjusted ← noise_scale × |mutated[$i$][continuous_idx]|
8:                 noise ← sample from $\mathcal{N}$(0, noise_scale_adjusted)
9:                 mutated[$i$][continuous_idx] ← mutated[$i$][continuous_idx] + noise
10:            end if
11:            if categorical_idx and cat_probs exist then
12:                for each categorical index $j$ do
13:                    if random value < mutation_prob then
14:                        mutated[$i$, cat_idx] ← sample from categorical distribution with probabilities cat_probs[$j$]
15:                    end if
16:                end for
17:            end if
18:        end if
19:    end for
20:    mutated ← replace NaN values with 0
21:    round and clip categorical values to valid range
22:    return flatten mutated array to 1D list
23: end function
```

**Algorithm 5** Mutate Distribution

```
 1: function MUTATE_DIST(individual, mutation_prob, gmm, categorical_idx, continuous_idx,
    cat_probs, n_features)
 2:     individual_data ← reshape individual to 2D array with $n\_features$ columns
 3:     mutated ← copy of individual_data
 4:     for each row $i$ in individual_data do
 5:         if random value < mutation_prob then
 6:             new_categorical ← empty list
 7:             new_continuous ← empty list
 8:             if categorical_idx and cat_probs exist then
 9:                 for each probability distribution $p$ in cat_probs do
10:                     append sample from categorical distribution with probabilities $p$ to new_categorical
11:                 end for
12:             end if
13:             if continuous_idx and gmm exists then
14:                 new_continuous ← sample from GMM
15:             end if
16:             new_row ← create new row with appropriate values at categorical and continuous indices
17:             mutated[$i$] ← new_row
18:         end if
19:     end for
20:     mutated ← replace NaN values with 0
21:     round and clip categorical values to valid range
22:     return flatten mutated array to 1D list
23: end function
```

**Algorithm 6** Mutate Covariance

```
 1: function MUTATE_COV(individual, mutation_prob, categorical_idx, continuous_idx, cat_probs,
    n_features)
 2:     individual_data ← reshape individual to 2D array with $n\_features$ columns
 3:     mutated ← copy of individual_data
 4:     if continuous_idx exists then
 5:         continuous_data ← individual_data[:, continuous_idx]
 6:         current_cov ← covariance matrix of continuous_data
 7:         ensure current_cov is positive definite
 8:         mean_vector ← mean of continuous_data
 9:         for each row $i$ in individual_data do
10:             if random value < mutation_prob then
11:                 new_continuous ← sample from multivariate normal with mean_vector and current_cov
12:                 mutated[$i$, continuous_idx] ← new_continuous
13:             end if
14:         end for
15:     end if
16:     if categorical_idx and cat_probs exist then
17:         for each row $i$ in individual_data do
18:             if random value < mutation_prob then
19:                 for each categorical index $j$ do
20:                     mutated[$i$, cat_idx] ← sample from categorical distribution with probabilities cat_probs[$j$]
21:                 end for
22:             end if
23:         end for
24:     end if
25:     mutated ← replace NaN values with 0
26:     round and clip categorical values to valid range
27:     return flatten mutated array to 1D list
28: end function
```

**Algorithm 7** Crossover

```
function CROSSOVER(ind1, ind2, cxpb, row_mode_prob, n_features)
    if random value $\geq$ cxpb then
        return ind1, ind2                                    ▷ No crossover performed
    end if
    matrix1 $\leftarrow$ reshape ind1 to 2D array with $n\_features$ columns
    matrix2 $\leftarrow$ reshape ind2 to 2D array with $n\_features$ columns
    $n\_samples$ $\leftarrow$ number of rows in matrix1
    if random value $<$ row_mode_prob then
        perform row-wise crossover
        $n\_rows \leftarrow \lfloor 0.3 \times n\_samples \rfloor$
        select $n\_rows$ random row indices
        swap selected rows between matrix1 and matrix2
    else
        perform column-wise crossover
        $n\_cols \leftarrow \lfloor 0.3 \times n\_features \rfloor$
        select $n\_cols$ random column indices
        swap selected columns between matrix1 and matrix2
    end if
    matrix1 $\leftarrow$ replace NaN values with 0
    matrix2 $\leftarrow$ replace NaN values with 0
    ind1 $\leftarrow$ flatten matrix1 to 1D list
    ind2 $\leftarrow$ flatten matrix2 to 1D list
    return ind1, ind2
end function
```

## B Selection of meta-features

The following experiment was conducted with the objective of selecting the most informative meta-features for the purpose of OOD evaluation. Such meta-features, which exhibit changes in their values that would accurately indicate a shift in the data, are of particular interest. The generation of toy-data with different types of shifts (synthetic) was undertaken, and then the changes in the values of meta-features (absolute differences) were measured on subsamples within the domain and between domains. Subsequently, a range of methodologies for meta-feature selection ([3]) were employed, and the frequency with which each meta-feature was selected was calculated (see Figure 2). The data with shifts were modeled synthetically; for example, the distribution $p(y|x)$ (concept shift) was explicitly altered, or the distributions of predictors (covariance shift) were explicitly changed.

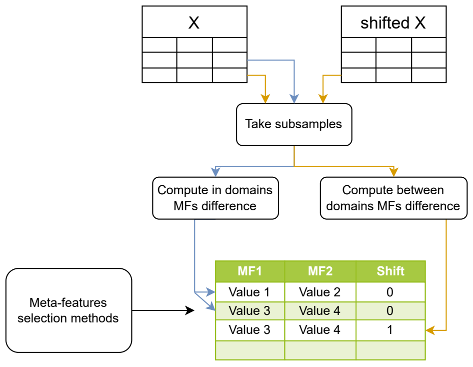
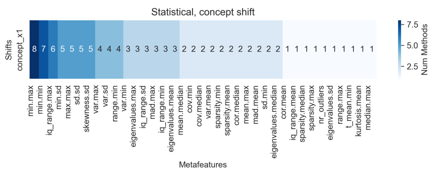
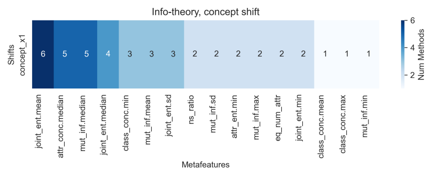

X
shifted X
Take subsamples
Compute in domains MFs difference
Compute between domains MFs difference
Meta-features selection methods
MF1
MF2
Shift
Value 1
Value 2
0
Value 3
Value 4
0
Value 3
Value 4
1

**Figure 2:** General outline of the experiment to determine the meta-features that respond best to the shift.

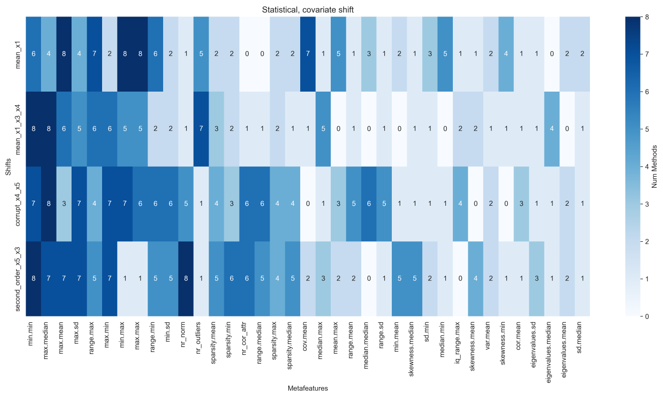
The selection of statistical and information-theoretical meta-characteristics as the primary groups was made on the basis of their relative simplicity in terms of interpretation. Furthermore, an investigation was conducted into various summarizing functions. Further information regarding meta-features and summarizing functions can be found in the following source ([24]). The figure 3 presents the concluding outcomes of the investigation into the variability of meta-characteristics across diverse groups under varying shifts.



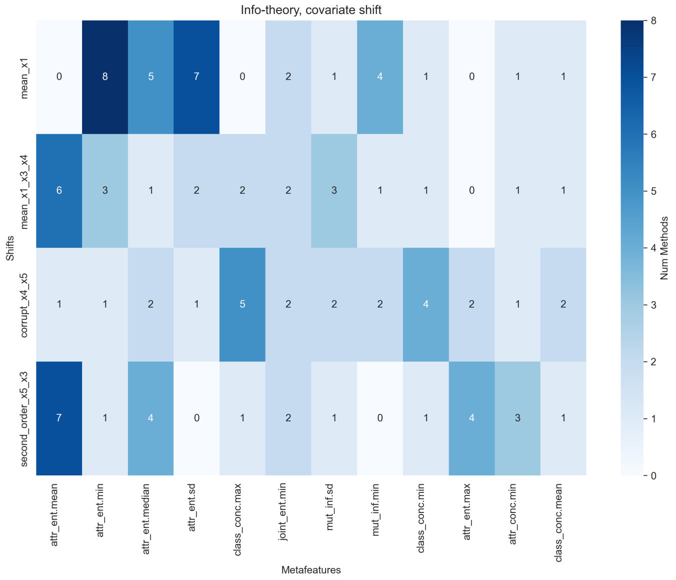
**(a)**



**(b)**



**(c)**

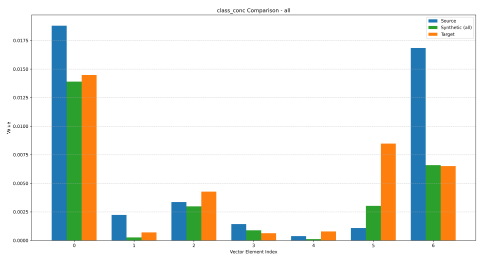


**(d)**

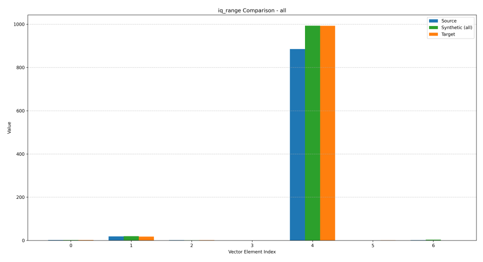
**Figure 3:** Results of the analysis of meta-feature variability under different shifts. Here, the meta-feature is named according to the principle $meta\_feature\_name.summarizing\_function$. The number on the diagram indicates how many meta-feature selection methods chose this meta-feature as significant for a given shift type.

## C Generation of synthetic OOD data

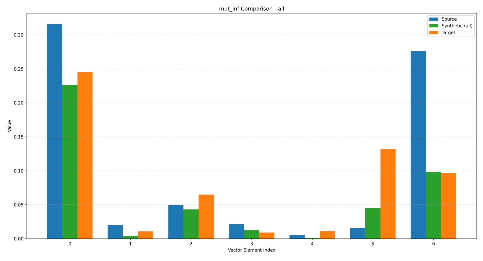
The following graphs compare the values of meta-features on real data and synthetic data (figure 4 for california dataset and figure 5 for electricity dataset). It is important to note that a decision was taken not to aggregate the meta-feature vector in the generation task, as this resulted in substantial quality degradation. Consequently, each meta-feature is represented by a vector in this study.



**(a)** class_conc

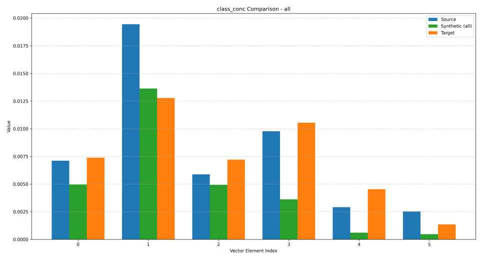


**(b)** iq_range

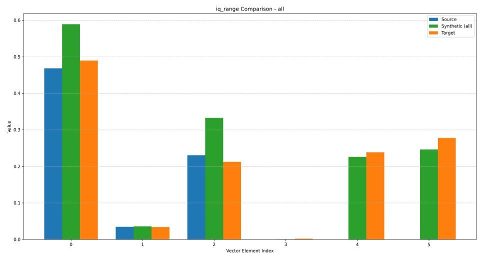


**(c)** mut_info

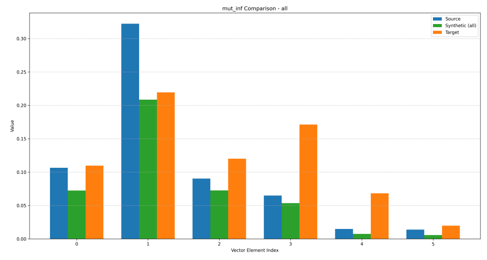
**Figure 4:** Comparison of meta-feature values on real and synthetic data for the dataset california.



**(a)**



**(b)**



**(c)**

**Figure 5:** Comparison of meta-feature values on real and synthetic data for the dataset electricity.

## 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: The Abstract and the "Contributions" paragraph explicitly state the two complementary protocols (MF-based splitting and synthetic generation) and claim empirical validation across datasets and learners, which matches the experiments and conclusions presented.

   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: The Conclusion contains an explicit limitations paragraph noting sensitivity to number/choice of meta-features, aggregation hiding per-predictor effects, and reliance on the chosen generative prior and operators.

   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: [NA]

Justification: The paper is empirical/algorithmic and does not present formal theorems or proofs, so the question is not applicable.

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 4 and Appendix A provide dataset descriptions, split procedures, evolutionary optimizer settings, hyperparameters, repetition counts (5 seeds), and model training details; a code/data repository link is also given to enable reproduction.

   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: The manuscript includes an anonymized repository URL for code and datasets and refers to implementation details in Appendix A, indicating provision of materials for reproduction.

   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, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 and the Models and evaluation subsection list data splits, detailed optimizer and model hyperparameters (LR, XGBoost, IRM, DRO), evolutionary parameters, and run counts.

   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: Tables report mean ± variability and the text states experiments are repeated five times to capture stochastic variation; error bars/variability are reported alongside the metrics supporting main claims.

   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: [No]

   Justification: The paper specifies algorithmic and experimental parameters (populations, generations, model hyperparameters) but does not report hardware (CPU/GPU types), memory, or runtime estimates required to reproduce the experiments.

   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: [NA]

   Justification: The manuscript does not include an explicit statement addressing adherence to the NeurIPS Code of Ethics or an ethics review, so an explicit confirmation is not present in the paper.

   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: [NA]

Justification: There is no societal impact of the work performed

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: the paper poses no such risks

    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: All resources used are correctly cited.

    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: [Yes]

    Justification: The authors state that source/target splits, synthetic datasets, and code are available in the linked repository and provide implementation details and pseudocode in Appendix A, B, which documents the assets and how they were produced.

    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: The work uses tabular datasets and synthetic data generation and does not involve crowdsourcing or human-participant data collection.

    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: The research does not involve human subjects or crowdsourcing; IRB considerations are not applicable.

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: The paper does not use large language models as part of the methods or experiments, so this item is not applicable.

    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.