

Distribution Shift Aware Neural Tabular Learning

Wangyang Ying¹, Nanxu Gong¹, Dongjie Wang², Xinyuan Wang¹, Arun Vignesh Malarkkan¹,
Vivek Gupta¹, Chandan K. Reddy³, Yanjie Fu¹

¹Arizona State University

²The University of Kansas

³Virginia Tech

{wyang4, nanxugong, xwang735, arun.malarkkan, vgupt140, yanjie.fu}@asu.edu, wangdongjie@ku.edu, reddy@cs.vt.edu

Abstract

Tabular learning transforms raw features into optimized spaces for downstream tasks, but its effectiveness deteriorates under distribution shifts between training and testing data. We formalize this challenge as the Distribution Shift Tabular Learning (DSTL) problem and propose a novel Shift-Aware Feature Transformation (SAFT) framework to address it. SAFT reframes tabular learning from a discrete search task into a continuous representation-generation paradigm, enabling differentiable optimization over transformed feature sets. SAFT integrates three mechanisms to ensure robustness: (i) shift-resistant representation via embedding decorrelation and sample reweighting, (ii) flatness-aware generation through suboptimal embedding averaging, and (iii) normalization-based alignment between training and test distributions. Extensive experiments show that SAFT consistently outperforms prior tabular learning methods in terms of robustness, effectiveness, and generalization ability under diverse real-world distribution shifts.

1 Introduction

Deep learning relies on large models and expensive GPUs. Data-Centric AI offers a cost-effective alternative by improving data quality to boost performance (Zha et al. 2023). Tabular data is prevalent in both academia and industry, where designing effective features is critical (Kusiak 2001; Ying et al. 2023). A key challenge arises when feature distributions shift between training and testing, even within the same domain, violating the common I.I.D. assumption. As a result, features learned during training can fail at test time (see **Figure 1**). We define this as the Distribution Shift Tabular Learning (DSTL) problem, which is crucial for improving generalization and robustness in real-world tabular data.

Related works solve DSTL only partially. **①** DSTL is related to feature transformation (e.g., $\{f_1, f_2\} \rightarrow \{f_1 + f_2\}$) (Ying et al. 2025). For instance, humans can manually transform a feature set with domain knowledge. Machine-assisted methods include principal components analysis (PCA) (Maćkiewicz and Ratajczak 1993), exhaustive-expansion-reduction approaches (Kanter and Veeramachaneni 2015; Khurana et al. 2016), iterative-feedback-improvement approaches (Khurana, Samulowitz, and Turaga 2018; Ying et al. 2023, 2024a). However, manual methods are time-consuming and incomplete. Among

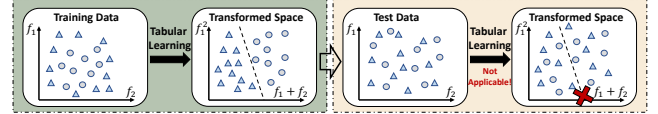


Figure 1: DSTL shows how distribution shifts between training and test sets impact tabular learning. The transformation applied during training fails to generalize in the test domain.

machine-assisted methods, PCA is only based on a straight linear feature correlation assumption, ignoring non-linear feature interaction, and PCA only reduces dimensionality instead of increasing dimensionality; other methods mostly rely on discrete search formulation with critical limitations: 1) large search space and time costly when enumerating combinations, 2) lacks generalization and robustness against imperfect data, such as distribution shifts. **②** DSTL is related to anti-shift learning, for instance, distribution alignment (Fan et al. 2024; Kim et al. 2021; Hu et al. 2023), parameter isolation (Zhang et al. 2023), adaptive experience replay (Li, Tang, and Li 2024). A key research gap remains: how can shift awareness be integrated into learning-based formulations that go beyond traditional discrete search?

There are two challenges to bridge the gap between feature transformation and shift awareness in tabular learning: 1) From search-based formulation to learning-based formulation. Existing studies view feature transformation as a search task for the best feature set among large, discrete feature combinations. The first challenge seeks to answer: what paradigm allows us to frame feature transformation as a learning problem - within a continuous space, guided by directional gradients, evaluated by tangible objectives, and measured using discriminative or generative models?

2) From shift-sensitive to shift-robust. Once the feature transformation learning paradigm is proposed, the second challenge intends to answer: what mechanisms can integrate shift awareness into such a learning paradigm?

Insight: Shift-aware Representation-Generation. We reformulate the DSTL problem as an optimization task in continuous space rather than a discrete search task. Imagine each discrete combination (e.g., the transformed feature set $\{f_1 + f_2, f_3 * f_4\}$) is a point in a high-dimensional space. By learning a continuous embedding of these combinations, we can reason over their relationships and apply gradient-

based optimization to search for better combinations. Although this representation-generation paradigm is common in domains like computer vision, its application to classical feature engineering introduces a new opportunity to integrate shift awareness into representation, generation, and pre/post-processing.

Summary of Proposed Solution. We adopt a representation generation-based deep tabular learning framework. In the representation step, an encoder is trained to embed transformed feature sets into vectors; an evaluator is optimized to predict performance from embeddings, and a decoder is learned to reconstruct the feature set from the embedding. The generation step then applies gradient ascent to identify the optimal embedding and decodes it into the best-transformed feature set. To address the distribution shift, we develop three mechanisms in response to shift-resistant representation, flatness-aware generation, and shift-aligned pre and post-processing: (i) Connecting embedding dimension decorrelation with sample reweighing can incorporate shift resistance into embedding. (ii) Leveraging flatness in gradient-based optimal embedding search can ensure that the performance of a transformed feature set embedding does not decrease significantly in a neighborhood around the maximum, thus mitigating distribution shift. (iii) Performing normalization in pre-processing and denormalization in post-processing can align distribution gaps between training and testing data.

Our contributions are: 1) We propose a representation-generation framework that enables continuous modeling of discrete feature transformations, addressing distribution shift in DSTL tasks. 2) We introduce key shift-resilient mechanisms (i.e., feature set representation, flatness-aware generation, and normalization) to enhance robustness under domain shifts. 3) Extensive experiments on benchmark datasets demonstrate the proposed method’s superior performance under distribution shift.

2 Problem Statement

The DSTL Problem. Given a training dataset and a test dataset that share the same set of features, and considering the existence of distribution shifts between training and test samples, we aim to learn a distribution shift-resilient tabular learning model that transforms the original datasets into a new representation space. The goal is to: 1) improve the performance of an ML task (e.g., regression, classification) and 2) ensure robustness to distribution shift. We adopt the perspective that knowledge of which feature combinations enhance data utility can be learned from past feature transformation–performance pairs. Based on this view, we formulate the DSTL problem as a shift-aware generative tabular learning task involving the following key concepts:

1) Feature Cross. We apply operations (e.g., $+$, $-$, $*$, $\sin(\cdot)$) to cross original features (e.g., $f_1 + f_2$).

2) Feature Cross Sequence. We convert feature crosses into postfix expressions and represent them as token sequences to reduce token redundancy and prevent invalid transformations. We concatenate multiple feature crosses using a separation token, and we prepend a start token and append an end token to the full sequence. For example, the two

feature crosses $f_1 + f_1/f_3 - f_2, \sqrt{f_1}$ are represented by: $\langle \text{sos} \rangle f_2 f_1 f_3 / f_3 - + \langle \text{sep} \rangle f_2 \sqrt{\cdot} \langle \text{eos} \rangle$.

3) Transformed Feature Set. A feature cross sequence defines a set of transformation rules. By applying each rule to the original features, we generate a collection of new features, which together form the transformed feature set.

We collect various feature cross sequences along with their corresponding performances and transformed feature sets as **observed values** from the training dataset. DSTL encodes these observed values into an embedding space and generates a new feature cross sequence. Using the transformation rules defined by the new feature cross sequence, we convert training and testing data points into a new space, thereby improving downstream performance.

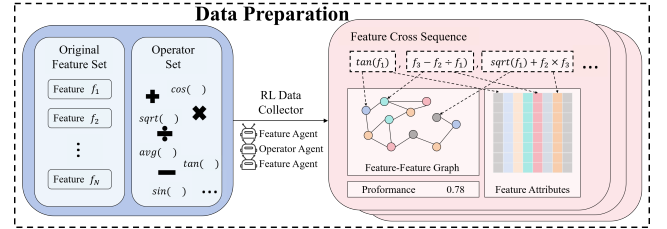


Figure 2: Data preparation pipeline. RL agents are employed to collect exemplary and diversified training data.

3 RL Agents for Data Collection

Before modeling, we first construct a supervised knowledge base by collecting various transformed feature sets (specifically, different combinations of feature crosses) and their corresponding performances (e.g., accuracy) on a specific downstream task (e.g., regression or classification). This forms the observation values needed to build a continuous representation space. To construct this space, we can manually or randomly generate feature sets through cross-operations and evaluate their performance on a controlled task, which provides the training data for learning the feature set-to-vector encoder. However, the three aspects of training data are critical: 1) *large volume*: sufficient training data; 2) *high quality*: high-accuracy feature set cases as successful experiences; 3) *high diversity*: training data should not ignore random and failure cases.

Leveraging reinforcement learning to explore high quality and diverse training data. We leverage reinforcement learning (RL) to automatically generate diverse and high-performing feature sets as training data for our representation model (Figure 2). Specifically, we design RL agents that iteratively perform feature crossing, guided by a reward signal based on downstream task performance. This process balances exploration and exploitation, enabling the collection of a large number of transformed feature sets paired with their performance metrics. These feature set–performance pairs form the supervised data used to train the feature set encoder. More details of the RL setup and training process are provided in Appendix A.

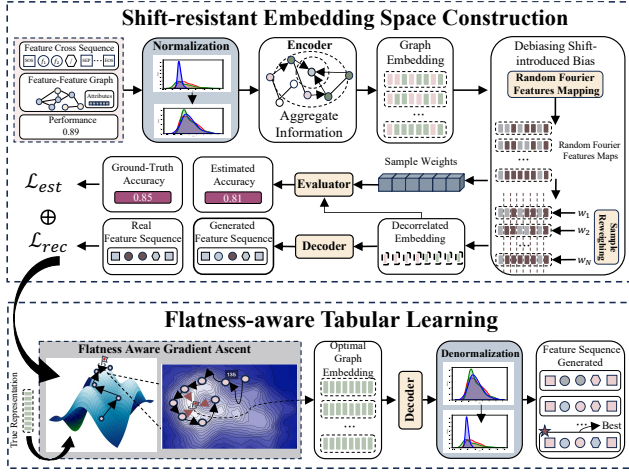


Figure 3: An overview of our proposed framework.

4 The Representation-Generation Framework

4.1 Framework Overview

Figure 3 shows our Shift-Aware Feature Transformation framework (SAFT) for tabular learning, which includes three components: 1) shift-resistant feature set representation; 2) flatness-aware tabular learning; 3) integrated pre-normalization and post-denormalization. To achieve a shift-resistant representation, we develop two insights: 1) feature sets as dynamic feature-feature interaction graphs and 2) a combination of sample reweighing and embedding dimension orthogonality for invariant representation. To achieve flatness-aware tabular learning, we incorporate flatness-aware gradient ascent, which seeks flatter optima in the embedding space for enhanced shift robustness. Finally, we integrate normalization in preprocessing and denormalization in postprocessing to further alleviate the impacts of shifts. We provide an algorithm in **Appendix B** to clarify the overall process.

4.2 Shift-resistant Feature Set Representation

To find the best transformed feature set, our key insight is to effectively represent explored transformed feature sets and corresponding performances in an embedding space. Using training data composed of transformed feature sets and their predictive performances, we train an encoder and a decoder. These components map transformed feature sets to vectors and vice versa, creating a continuous embedding space. We believe that the best transformed feature set, termed feature knowledge, is described in this continuous embedding space, and thus can be efficiently identified by gradient optimization. As a result, we don't need to explore exponentially growing possibilities of feature combinations. Although training-testing shifts of data points may distort the embedding space, it is learnable. Anti-shift mechanisms can rectify and adjust the distortion and ensure reliable and transferable feature knowledge in the testing set.

The Encoder-Evaluator-Decoder Framework ① Encoder for Learning Dynamic Feature-Feature Interaction Graph Embedding.

Representing transformed feature sets is challenging due to strong inter-feature dependencies, which are often ignored by traditional methods. Simple aggregation of statistics or sequence-based encoding treats features as independent, leading to biased representations and suboptimal feature set selection. We propose modeling a transformed feature set as a graph, where each node represents a feature, node attributes correspond to feature values, and edges reflect pairwise similarities (e.g., cosine similarity). This enables capturing feature-feature interactions directly. Although graph neural networks (e.g. GCNs) can embed such graphs, they assume fixed sizes and fixed topologies, making them unsuitable for feature sets with varying sizes and connections. Inspired by (Hamilton, Ying, and Leskovec 2017), we adapt node sampling and multi-hop neighborhood aggregation for dynamic feature-feature graph embedding. At each step, each node updates its representation by combining its own and neighbors' previous embeddings, followed by a nonlinear transformation via a fully connected layer. We normalize the output at each step and iterate to produce the final embedding. Formally, given a transformed feature set, we construct a feature-feature similarity graph $\mathcal{G}$ and encode it using an encoder ϕ_θ , yielding $E = \phi_\theta(\mathcal{G})$.

② *Evaluator for Estimating Feature Set Performance.* Our goal is to learn an embedding space that accurately represents transformed feature sets, allowing us to find the optimal embedding yielding higher performance, which is then used to reconstruct the optimal transformed feature set. To strengthen the expressiveness of the transformed feature set embedding, we incorporate a performance evaluator as a downstream task following the feature-feature graph encoder. Specifically, we employ a feedforward network to take an embedding as input and predict the performance of a transformed feature set, by minimizing the mean squared error between estimated performances and ground-truth. Formally, the evaluator loss is $\mathcal{L}_{est} = \frac{1}{N} \sum_{i=1}^N (p_i - \omega_\theta(E_i))^2$, where N is the total number of transformed feature sets in training data, i indexes each transformed feature set, p_i is the ground-truth performance corresponding to the transformed feature set, ω_θ is the evaluator function, and E_i is a specific graph embedding.

③ *Decoder.* The primary objective of the decoder is to output a feature cross sequence (deemed as a token sequence) given a feature-feature graph embedding. The decoder provides two main functionalities: 1) Ensuring embedding faithfulness: the embedding of a transformed feature set should be able to fully reconstruct its corresponding feature cross sequences; 2) Generating the optimal transformed feature set: a well-trained decoder can be used to generate the feature cross sequence of the optimal embedding. Our decoder is structured by a single-layer LSTM and a softmax layer, where the LSTM recursively learns hidden states over the current-previous dependent feature cross sequence, and the softmax layer is used to estimate the token probability when generating each token. Specifically, given ψ as a single-layer LSTM, the distribution of the single i -th to

ken is $P_\psi(\gamma_i|E, \Upsilon_{<i}) = \frac{\exp(s_i)}{\sum_M \exp(s)}$. where, Υ is a feature cross sequence with M token length, γ_i is the i -th token in Υ , and s_j is the j -th output of the softmax layer. Finally, we aim to minimize the negative log-likelihood of decoding a feature cross sequence as the reconstruction loss: $\mathcal{L}_{rec} = -\sum_N \sum_{i=1}^M \log P_\psi(\gamma_i|E, \Upsilon_{<i})$.

④ The Joint Objective Function. We optimize the joint loss of the encoder, evaluator, and decoder. Our objective is: $\mathcal{L}_{est} + \gamma \mathcal{L}_{rec}$, where γ is a hyperparameter that balances the loss. In this way, we enforce the encoder-evaluator-decoder structure to learn a faithful embedding that accurately predicts the performance of a transformed feature set and decode the corresponding feature cross sequence.

Correcting Training Biases under Distribution Shifts.

① Motivation: Toward Debaised Representations under Distribution Shifts. In open environments, distribution shifts introduce bias into the learned embedding space by entangling invariant (true) and spurious (false) representations. Their correlation can mislead optimization and reduce robustness. A common mitigation is to decorrelate embeddings by minimizing the Frobenius norm of the covariance matrix, but this approach assumes structural orthogonality and may cause information loss. We aim to suppress spurious bias by amplifying invariant signals. Inspired by stable sample weighting (Shen et al. 2020) and robust reweighting (Athey, Imbens, and Wager 2018), we propose a bilevel framework: the inner loop learns sample weights to decorrelate embeddings, and the outer loop reweights losses for graph-based feature evaluation. This jointly mitigates bias and improves generalization under distribution shift.

Algorithm 1: Shift-resistant Bilevel Training

Require: EPOCH_1 and EPOCH_2

Ensure: Learned Embedding Space

```

1: for out_loop_epoch  $\leftarrow$  1 to EPOCH_1 do
2:   Forward propagate
3:   for inner_loop_epoch  $\leftarrow$  1 to EPOCH_2 do
4:     Optimize the sample weighting by Equation (1)
5:   end for
6:   Back propagate with the weighted joint loss  $\mathcal{L} = \mathcal{L}_{est} + \gamma \mathcal{L}_{rec}$  to joint train
7: end for
```

② The Shift-resistant Bilevel Training. We introduce a two-loop algorithm to correct the training biases. 1) For the inner-loop, our objective is to learn the best sample weights to minimize the squared Frobenius norm as follows.

$$\mathbf{R}^* = \arg \min_{\mathbf{R}} \sum_{i < j} \left\| \hat{\mathbf{C}}_{E_{*i}, E_{*j}}^{\mathbf{R}} \right\|_F^2, \quad (1)$$

$$\text{where } \hat{\mathbf{C}}_{E_{*i}, E_{*j}}^{\mathbf{R}} = \frac{1}{N-1} \sum_{n=1}^N \left[\left(r_n f(E_{ni}) - \frac{1}{N} \sum_{m=1}^N r_m f(E_{mi}) \right)^\top \cdot \left(r_n g(E_{nj}) - \frac{1}{N} \sum_{m=1}^N r_m g(E_{mj}) \right) \right].$$

Here, $\hat{\mathbf{C}}_{E_{*i}, E_{*j}}^{\mathbf{R}}$ is the partial cross-covariance matrix, $\mathbf{R} = \{r_n\}_{n=1}^N$ is the graph weight vector, r_i is the weight of

i -th feature subset graph and we constrain $\sum_{n=1}^N r_n = N$. $f(\cdot)$ and $g(\cdot)$ are random Fourier feature functions, E_{*i} and E_{*j} denote different dimensions of the same training sample. 2) For the outer-loop training, we use the updated weights of the training samples $\mathbf{R}^* = \{r_n^*\}_{n=1}^N$ to reweight the estimation loss of each transformed feature set, given by $\mathcal{L}_{est} = \sum_{i=1}^N r_i^* (p_i - \omega_\theta(E_i))^2$. Then, we combine the new estimator loss and the decoder loss $\mathcal{L}_{rec}$ to obtain the weighted joint loss: $\mathcal{L} = \mathcal{L}_{est} + \gamma \mathcal{L}_{rec}$. The **Algorithm 1** provides a detailed pseudocode.

4.3 Flatness-aware Generation

After learning the embedding space of transformed feature sets, the embedding space accurately captures and represents all transformed feature sets, including the unobserved best transformed feature set. Hence, we utilize gradient ascent to identify the best embedding vector with the highest performance to decode the optimal transformed feature set.

Step 1: Gradient-ascent Optimization to Identify the Optimal Feature Set Embedding. Another benefit of learning an embedding performance evaluator in the representation step is that it enables differentiable gradient optimization. Specifically, we extract the gradient from the evaluator in the direction of maximizing the performance of the feature set. Formally, the gradient-ascent is defined by: $\hat{E} = E + \eta \frac{\partial \omega_\theta}{\partial E}$, where ω_θ is the evaluator, $\hat{E}$ is the optimal embedding, η is the step size. In the experiments, we select top-T transformed feature set embeddings in training data as the initialization seeds for gradient ascent. Therefore, we can identify T-improved embedding vectors as the optimal embedding set $\hat{\mathcal{E}} = \{\hat{E}^t\}_{t=1}^T$.

Algorithm 2: Flatness-Aware Gradient Ascent

Require: Initialized embedding E , LR bounds η_1, η_2 , cycle length c , number of iterations n

Ensure: Optimal embedding $\hat{E}$

```

1:  $\hat{E} \leftarrow E$ 
2: for  $i = 1, 2, \dots, n$  do
3:    $\eta \leftarrow \eta(i)$  { Calculate LR for the iteration,  $\eta_1 < \eta(i) < \eta_2$  }
4:    $E \leftarrow E + \eta \frac{\partial \omega_\theta}{\partial E}$  { Stochastic gradient update }
5:   if  $\text{mod}(i, c) = 0$  then
6:      $n_{\text{models}} \leftarrow \frac{i}{c}$  { Number of models }
7:      $\hat{E} \leftarrow \frac{\hat{E} \cdot n_{\text{models}} + E}{n_{\text{models}} + 1}$  { Update average }
8:   end if
9: end for
```

Step 2: Incorporating Flatness-awareness in Gradient Ascent to Mitigate Shifts. In gradient optimization, the flatness of the loss landscape has been shown to exhibit a close connection with distribution shift resistance both theoretically and empirically. When there exists a distribution shift between training and testing data, the embedding of the optimal transformed feature set in the test samples does not coincide with the optimal transformed feature set found in the training samples. Flatness ensures that the loss

does not increase significantly in a neighborhood around the identified minimum. Therefore, flatness leads to distribution shift resistance because the loss in the test examples does not increase significantly. Inspired by (Izmailov et al. 2018; Garipov et al. 2018), we take advantage of loss flatness and develop a flatness-aware gradient ascent approach.

Specifically, unlike the classic gradient ascent, we propose to enforce a search process that oscillates around the optimal embedding to collect more suboptimal embedding vectors. This suboptimal embedding set can pinpoint a flat region where the true optimal point is located for the test set. We then aggregate all the suboptimal points as the final averaged embedding that represents the center of the flat region in the loss landscape. However, averaging all the suboptimal points of each gradient ascent iteration introduces huge computational costs and may include the non-optimal points, which are far away from the optimal neighborhood. We therefore develop a cyclic scheme, in which a cycle includes multiple gradient ascent iterations, and we only average the suboptimal points at the end of a cycle. To avoid missing the optimal point when approaching the maximum, we utilize linearly decreasing learning rates within each cycle. In particular, **Algorithm 2** shows that we linearly decrease the learning rate $\eta(i)$ from η_1 to η_2 over iterations during a cycle. In each iteration, we initialize the learning rate and update the embedding by gradient ascent. At the end of each cycle, we average the embedding (Line 7).

Step 3: Decoding Embeddings to Reconstruct Optimal Feature Cross Sequences After obtaining the candidate embedding set $\hat{\mathcal{E}}$, we use the well-trained decoder ψ to generate the feature cross sequences, i.e., $\hat{\mathcal{E}} \xrightarrow{\psi} \{\hat{\mathbf{Y}}_i\}_{i=1}^T$. Specifically, the decoder iteratively generates the next token of a feature cross sequence, such as a feature token, an operator token, or a segmentation token ($< \text{sep} >$) in an autoregressive manner until it produces an end-of-sequence token ($< \text{eos} >$). Finally, we divide the generated sequence into multiple segments using the " $< \text{sep} >$ " token and transform each segment into a feature, resulting in the optimal transformed feature set with the best estimated performance.

4.4 Integrating Normalization-Denormalization into Pre- and Post-Processing

In addition to embedding dimension orthogonality and flatness-aware optimization, we propose a data-centric perspective to alleviate distribution shift by aligning and recovering the distribution gaps between training and test data in pre- and post-processing. We incorporate a normalization and denormalization mechanism. Specifically, we first normalize (e.g., z-score) both training and testing data before feature transformation, obtaining normalized data with zero mean and unit variance. We train our method on normalized training data to generate the optimal feature cross-sequence under a normalized distribution. Next, we apply this learned sequence to transform the normalized test data. Then, we denormalize (e.g., reverse z-score) the transformed test data to restore the original distribution. This approach enables feature transformation learning in a normalized distribution and avoids shift-induced bias.

5 Experimental Results

5.1 Experimental Setup

Datasets. We perform experiments on 16 publicly available datasets from UCI (Dua and Graff 2017) and OpenML (Vanschoren et al. 2013), which are widely used to evaluate tabular learning methods. The statistics of all datasets are described in **Table 1**. To evaluate SAFT’s robustness to distribution shifts, we iteratively split the dataset (80%/20%) along each feature and apply the Kolmogorov–Smirnov test (Berger and Zhou 2014) at 95% confidence. If no shift is detected in any feature, the dataset is excluded.

Evaluation Metrics. To minimize the influence of downstream models, we employ Random Forests (RF) for all tasks. Evaluation uses F1, Precision, and Recall for classification, and 1-RAE, 1-MAE, and 1-MSE (Hill 2012; Hodson 2022) for regression, where higher scores reflect better feature transformations.

Baselines. We compare our framework with 10 widely used feature transformation methods: 1) **Random Generation (RDG)**; 2) **Essential Random Generation (ERG)**; 3) **Latent Dirichlet Allocation (LDA)** (Blei, Ng, and Jordan 2003); 4) **AutoFeat Automated Transformation (AFAT)** (Horn, Pack, and Rieger 2020); 5) **Neural Feature Search (NFS)** (Chen et al. 2019); 6) **Traversal Transformation Graph (TTG)** (Khurana, Samulowitz, and Turaga 2018); 7) **Group-wise Reinforcement Feature Generation (GRFG)** (Wang et al. 2022); 8) **reinforceMent-enhanced autoregressive feAture Transformation (MOAT)** (Wang et al. 2023); 9) **uNsupervised generative feAture Transformation (NEAT)** (Ying et al. 2024d); 10) **Evolutionary Large Language Model framework for automated Feature Transformation (ELLM-FT)** (Gong et al. 2025c). More details about the baselines are provided in **Appendix C**.

To make the experiment reproducible, we provide the details about the parameters and environment in **Appendix C**.

5.2 Performance Results

In this experiment, we compare the feature transformation performance of SAFT with baseline models. **Table 1** shows the overall comparison results in terms of F1 score and 1-RAE. We observe that SAFT consistently outperforms baseline models across all datasets. The underlying driver for this observation is that SAFT can compress the feature learning knowledge into a robust embedding space through shift-resistant embedding learning, and smoothly search for the optimal feature space via flatness-aware weight averaging. This helps preserve generalizable patterns across shifts. Moreover, we notice that the second-best model varies among different cases. A potential reason for this observation is that baseline approaches overlook the impact of distribution shifts, resulting in an unstable identification of the optimal feature space. This sensitivity reflects a lack of robustness under varying distributions. This experiment underscores that SAFT effectively captures invariant knowledge in feature learning against distribution shifts and produces a robust transformed feature space.

Table 1: Overall performance across various real-world datasets. The best and second-best outcomes are indicated by bold and underlined fonts, respectively. We measure the performance on classification (C) and regression (R) tasks using F1-score and (1-RAE) metrics, respectively. A higher value indicates higher quality of the feature transformation space.

Dataset	C/R	Samples	Features	RDG	ERG	LDA	AFAT	NFS	TTG	GRFG	MOAT	NEAT	ELLM-FT	SAFT
Housing Boston	R	506	13	0.375	0.366	0.146	0.387	0.395	0.383	0.361	0.395	0.367	0.395	0.405
Airfoil	R	1503	5	0.733	0.695	0.522	0.742	0.742	0.738	0.614	0.724	0.698	0.743	0.743
openml.586	R	1000	25	0.542	0.536	0.104	0.540	0.543	0.543	0.334	0.616	0.626	0.616	0.649
openml.589	R	1000	50	0.509	0.472	0.099	0.467	0.470	0.469	0.436	0.496	0.469	0.509	0.582
openml.607	R	1000	50	0.306	0.310	0.054	0.344	0.358	0.354	0.371	0.424	0.358	0.438	0.516
openml.616	R	500	50	0.197	0.311	0.014	0.342	0.344	0.343	0.459	0.369	0.528	0.528	0.534
openml.618	R	1000	50	0.421	0.391	0.058	0.436	0.430	0.431	0.257	0.427	0.391	0.468	0.468
openml.620	R	1000	25	0.510	0.464	0.025	0.475	0.464	0.462	0.495	0.566	0.545	0.545	0.545
openml.637	R	500	50	0.265	0.340	0.015	0.365	0.344	0.339	0.380	0.381	0.380	0.400	0.424
Higgs Boston	C	50000	28	0.693	0.694	0.508	0.692	0.691	0.696	0.698	0.702	0.702	0.704	0.704
SpectF	C	267	44	0.674	0.792	0.651	0.643	0.716	0.672	0.728	0.766	0.776	0.792	0.799
UCI Credit	C	30000	25	0.809	0.808	0.743	0.805	0.805	0.803	0.797	0.808	0.809	0.803	0.816
Wine Quality Red	C	999	12	0.658	0.491	0.588	0.662	0.650	0.675	0.668	0.681	0.622	0.680	0.700
Wine Quality White	C	4900	12	0.730	0.726	0.602	0.715	0.724	0.718	0.680	0.730	0.718	0.730	0.734
PimaIndian	C	768	8	0.679	0.761	0.685	0.636	0.691	0.734	0.689	0.763	0.754	0.760	0.780
German Credit	C	1000	24	0.697	0.662	0.693	0.656	0.698	0.683	0.647	0.707	0.723	0.701	0.743

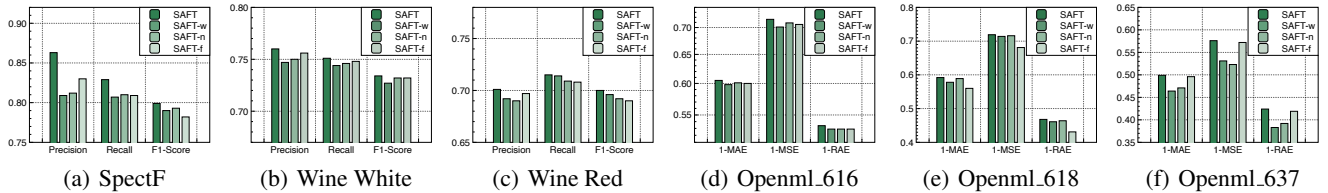


Figure 4: Results on the impact of normalization, Flatness Aware Gradient Ascent, and reweighting.

5.3 Ablation Study

To evaluate the necessity of various components in SAFT, we develop three model variants: 1) SAFT-f, which excludes the Flatness-Aware Gradient Ascent and does not perform weight averaging; 2) SAFT-n, which omits the normalization process; 3) SAFT-w, which removes the optimization of graph weights. We select three classification and three regression datasets for conducting experiments. **Figure 4** shows comparison results. First, we observe that, in all situations, the performance of the three model variants declines compared to SAFT. A potential reason is that disregarding distribution-shift-aware strategies fails to consider the variation between training and testing sets, resulting in suboptimal feature transformation performance. Moreover, SAFT can perform better compared to SAFT-w. This observation indicates that the learned graph weights help eliminate spurious correlations among features, thus enhancing the transformed feature space. Furthermore, we find that SAFToutperforms SAFT-f. The underlying driver is that Flatness-Aware Gradient Ascent prevents the gradient search process from converging to locally optimal points, thereby smoothing and making the search process more robust. Furthermore, SAFT-n is inferior to SAFT. This observation reflects that aligning statistical properties through normalization is an effective strategy to tackle the OOD issue in feature space. Thus, these experiments demonstrate that each technical component of SAFT is indispensable for mitigating the distribution shift on feature transformation.

5.4 Robustness Analysis

Dataset Split Robustness. To check the robustness of SAFT for different data splits, we use five distinct splits on the

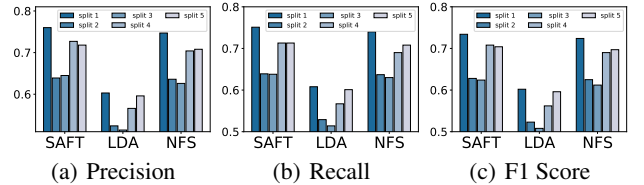


Figure 5: Robustness of the dataset split methods.

Wine Quality White dataset for evaluation. Each split defines a unique 20% proportion of the dataset as the testing set, moving sequentially through the dataset from beginning to end, with the remainder serving as the training set. **Figure 5** presents the comparison results in terms of F1-score, Precision, and Recall. We notice that despite potential distribution shifts introduced by different dataset split approaches, SAFT still demonstrates robust performance in addressing these shifts. The main reason for this observation is that SAFT can preserve the invariant and robust feature learning knowledge within the embedding space, enabling a smooth and stable search. Such a learning mechanism can alleviate the impact of distribution shifts across different data splits. Hence, this experiment shows that SAFT exhibits robust performance in addressing distribution shift, regardless of the dataset split method.

Downstream Performance Robustness. To check the robustness of SAFT for distinct downstream ML models, we substitute the downstream model with Random Forest (RF), Support Vector Machine (SVM), K-Nearest Neighborhood (KNN), Ridge, LASSO, and Decision Tree (DT). **Table 2** shows the comparison results on the Wine Quality Red dataset in terms of the F1-score. We find that SAFT outperforms baselines in most cases. A potential reason for this observation is that SAFT captures invariant and generaliz-

Table 2: Robustness check on ML models for the Wine Quality White dataset.

	RF	SVM	KNN	DT	LASSO	Ridge
RDG	0.658	0.632	0.629	0.630	0.634	0.606
ERG	0.491	0.354	0.358	0.413	0.470	0.522
LDA	0.588	0.439	0.512	0.579	0.390	0.410
AFAT	0.662	0.582	0.443	0.663	0.596	0.587
NFS	0.650	0.634	0.542	0.639	0.634	0.601
TTG	0.675	0.644	0.525	0.602	0.630	0.620
GRFG	0.668	0.313	0.502	0.616	0.551	0.540
MOAT	0.588	0.359	0.512	0.629	0.590	0.606
NEAT	0.653	0.630	0.500	0.633	0.630	0.601
ELLM-FT	0.662	0.634	0.542	0.668	0.659	0.638
SAFT	0.700	0.655	0.556	0.677	0.659	0.658

able feature knowledge, leading to its superior generalization across various downstream ML models. Thus, this experiment demonstrates the robustness of SAFT across various ML models.

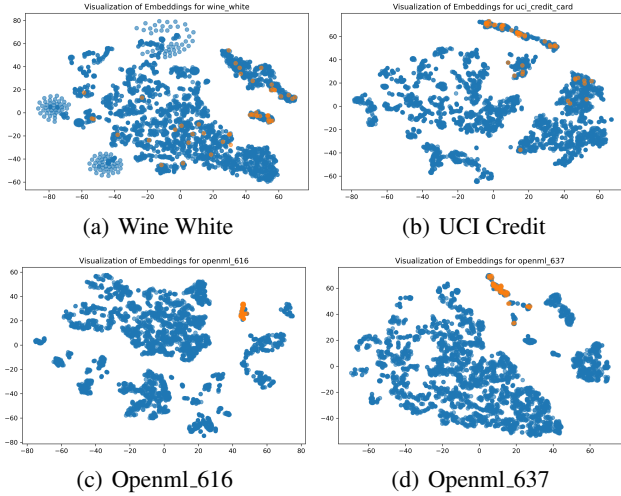


Figure 6: Visualization of the converged embedding space. Each point represents a feature transformation sequence, with orange points indicating the top 50 embeddings ranked by downstream performance.

5.5 Visualization for the Embedding Space

In this section, we visualize the embedding space to validate the effectiveness of the intermediate encoding. We collect latent embeddings from transformation records and apply t-SNE to project them into two dimensions. **Figure 6** presents results on four datasets, where each point represents a transformation operation sequence. Despite varying sequence lengths leading to spread-out distributions, we observe clear clustering patterns, particularly among top-performing sequences (orange points), which are densely grouped. This suggests that high-quality transformations tend to concentrate in specific regions, enabling efficient gradient-based search when using these clusters as initialization. These results demonstrate that the embedding space captures meaningful structure, supporting effective reconstruction and performance estimation.

To make the experiments more convincing, we conduct qualitative analysis, complexity comparisons with base-lines, and complexity analysis of SAFTs presented in **Appendix C.1, C.2, and C.3**, respectively.

6 Related Work

Feature Transformation aims to construct an enhanced feature space by transforming original features. Previous research can be categorized into three groups: 1) *Expansion-reduction based approaches* (Kanter and Veeramachaneni 2015; Horn, Pack, and Rieger 2020; Khurana et al. 2016), where the original feature space is expanded through explicit or greedy mathematical transformations and subsequently reduced by selecting valuable features. 2) *Evolution-evaluation approaches* (Hu et al. 2024; Khurana, Samulowitz, and Turaga 2018; Tran, Xue, and Zhang 2016), where evolutionary algorithms or reinforcement learning models are employed to optimize the process of iteratively creating effective features and retaining significant ones. 3) *Generative AI-based* (Gong et al. 2025a,b,d,c), where the most appropriate model architecture is automatically identified to formulate AFT as an AutoML task. However, these methods face two challenges: 1) Generating high-order feature transformations is difficult; 2) Their transformation performance is often unstable. Distribution Shift Problems: arising from inconsistent distributions between the training set and test set, are prevalent in the real-world domains. Existing works to solve this problem mainly focus on time series analysis (Fan et al. 2023), computer vision (Yu et al. 2023), and natural language processing (Dou et al. 2022), etc. However, distribution shift in the context of feature transformation remains underexplored, despite its significant impact on the effectiveness of transformed features. To address this gap, we integrate three shift-aware mechanisms at different levels of our framework: 1) *Shift-resistant representation*, which captures invariant representations under DSTL; 2) *Flatness-aware generation*, which identifies robust embeddings during the search process; and 3) *Shift-aligned pre- and post-processing*, which reduces distribution mismatches during data transformation. This work advances feature engineering (Ying et al. 2024b; Gong et al. 2025e; Ying et al. 2024c; Wang et al. 2024; Bai et al. 2025; Wang et al. 2025). Such progress benefits critical domains, from renewable energy forecasting (Huo et al. 2025a) to image translation (Zhu et al. 2025) and fraud detection (Huo et al. 2025b), where reliable representations enhance sustainability, creativity, and financial security.

7 Conclusion

We propose the Shift-Aware Feature Transformation (SAFT) framework, which integrates representation learning, generation, and anti-shift mechanisms to enable robust feature transformation. Our contributions are summarized as follows: First, we reformulate DSTL as a gradient-based optimization problem and design a representation-generation framework that enables the deployment of an anti-shift mechanism to solve the OOD problem. Second, to address the distribution shift, we incorporate three mechanisms at different levels of the framework: 1) *Shift-resistant representation*, to learn invariant features under DSTL; 2) *Flatness-aware generation*, to improve embedding robustness during optimization; 3) *Distribution-aligned pre- and post-processing*, to reduce train-test discrepancies. Finally,

empirical results demonstrate that SAFT generates a robust feature space and discovers an optimal transformed feature set using only training data, without requiring knowledge of the test distribution. This highlights its strong generalization ability and potential impact across open-ended domains.

References

- Athey, S.; Imbens, G. W.; and Wager, S. 2018. Approximate residual balancing: debiased inference of average treatment effects in high dimensions. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 80(4): 597–623.
- Bai, H.; Ying, W.; Gong, N.; Wang, X.; and Fu, Y. 2025. Privacy-preserving data reprogramming. *npj Artificial Intelligence*, 1(1): 10.
- Berger, V. W.; and Zhou, Y. 2014. Kolmogorov–smirnov test: Overview. *Wiley statsref: Statistics reference online*.
- Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan): 993–1022.
- Chen, X.; Lin, Q.; Luo, C.; Li, X.; Zhang, H.; Xu, Y.; Dang, Y.; Sui, K.; Zhang, X.; Qiao, B.; et al. 2019. Neural feature search: A neural architecture for automated feature engineering. In *2019 IEEE International Conference on Data Mining (ICDM)*, 71–80. IEEE.
- Dou, S.; Zheng, R.; Wu, T.; Gao, S.; Shan, J.; Zhang, Q.; Wu, Y.; and Huang, X. 2022. Decorrelate irrelevant, purify relevant: Overcome textual spurious correlations from a feature perspective. *arXiv preprint arXiv:2202.08048*.
- Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository.
- Fan, W.; Wang, P.; Wang, D.; Wang, D.; Zhou, Y.; and Fu, Y. 2023. Dish-TS: a general paradigm for alleviating distribution shift in time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 7522–7529.
- Fan, W.; Zheng, S.; Wang, P.; Xie, R.; Bian, J.; and Fu, Y. 2024. Addressing Distribution Shift in Time Series Forecasting with Instance Normalization Flows. *arXiv preprint arXiv:2401.16777*.
- Garipov, T.; Izmailov, P.; Podoprikin, D.; Vetrov, D. P.; and Wilson, A. G. 2018. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31.
- Gong, N.; Dong, S.; Bai, H.; Wang, X.; Ying, W.; and Fu, Y. 2025a. Agentic Feature Augmentation: Unifying Selection and Generation with Teaming, Planning, and Memories. *arXiv preprint arXiv:2505.15076*.
- Gong, N.; Li, Z.; Dong, S.; Bai, H.; Ying, W.; Wang, X.; and Fu, Y. 2025b. Sculpting Features from Noise: Reward-Guided Hierarchical Diffusion for Task-Optimal Feature Transformation. *arXiv preprint arXiv:2505.15152*.
- Gong, N.; Reddy, C. K.; Ying, W.; Chen, H.; and Fu, Y. 2025c. Evolutionary large language model for automated feature transformation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 16844–16852.
- Gong, N.; Wang, X.; Ying, W.; Bai, H.; Dong, S.; Chen, H.; and Fu, Y. 2025d. Unsupervised Feature Transformation via In-context Generation, Generator-critic LLM Agents, and Duet-play Teaming. *arXiv preprint arXiv:2504.21304*.
- Gong, N.; Ying, W.; Wang, D.; and Fu, Y. 2025e. Neuro-symbolic embedding for short and effective feature selection via autoregressive generation. *ACM Transactions on Intelligent Systems and Technology*, 16(2): 1–21.
- Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.
- Hill, A. V. 2012. *The encyclopedia of operations management: a field manual and glossary of operations management terms and concepts*. Ft Press.
- Hodson, T. O. 2022. Root-mean-square error (RMSE) or mean absolute error (MAE): When to use them or not. *Geoscientific Model Development*, 15(14): 5481–5487.
- Horn, F.; Pack, R.; and Rieger, M. 2020. The autofeat python library for automated feature engineering and selection. In *Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, 111–120. Springer.
- Hu, X.; Fan, W.; Yi, K.; Wang, P.; Xu, Y.; Fu, Y.; and Wang, P. 2023. Boosting Urban Prediction via Addressing Spatial-Temporal Distribution Shift. In *2023 IEEE International Conference on Data Mining (ICDM)*, 160–169. IEEE Computer Society.
- Hu, X.; Wang, D.; Ying, W.; and Fu, Y. 2024. Reinforcement Feature Transformation for Polymer Property Performance Prediction. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 4538–4545.
- Huo, M.; Lu, K.; Li, Y.; Zhu, Q.; and Chen, Z. 2025a. Ct-patchtst: Channel-time patch time-series transformer for long-term renewable energy forecasting. *arXiv preprint arXiv:2501.08620*.
- Huo, M.; Lu, K.; Zhu, Q.; and Chen, Z. 2025b. Enhancing customer contact efficiency with graph neural networks in credit card fraud detection workflow. *arXiv preprint arXiv:2504.02275*.
- Izmailov, P.; Podoprikin, D.; Garipov, T.; Vetrov, D.; and Wilson, A. G. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.
- Kanter, J. M.; and Veeramachaneni, K. 2015. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, 1–10. IEEE.
- Khurana, U.; Samulowitz, H.; and Turaga, D. 2018. Feature engineering for predictive modeling using reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.
- Khurana, U.; Turaga, D.; Samulowitz, H.; and Parthasarathy, S. 2016. Cognito: Automated feature engineering for supervised learning. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 1304–1307. IEEE.

Kim, T.; Kim, J.; Tae, Y.; Park, C.; Choi, J.-H.; and Choo, J. 2021. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*.

Kusiak, A. 2001. Feature transformation methods in data mining. *IEEE Transactions on Electronics packaging manufacturing*, 24(3): 214–221.

Li, X.; Tang, B.; and Li, H. 2024. AdaER: An adaptive experience replay approach for continual lifelong learning. *Neurocomputing*, 572: 127204.

Maćkiewicz, A.; and Ratajczak, W. 1993. Principal components analysis (PCA). *Computers & Geosciences*, 19(3): 303–342.

Shen, Z.; Cui, P.; Zhang, T.; and Kunag, K. 2020. Stable learning via sample reweighting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 5692–5699.

Tran, B.; Xue, B.; and Zhang, M. 2016. Genetic programming for feature construction and selection in classification on high-dimensional data. *Memetic Computing*, 8: 3–15.

Vanschoren, J.; van Rijn, J. N.; Bischl, B.; and Torgo, L. 2013. OpenML: networked science in machine learning. *SIGKDD Explorations*, 15(2): 49–60.

Wang, D.; Fu, Y.; Liu, K.; Li, X.; and Solihin, Y. 2022. Group-wise reinforcement feature generation for optimal and explainable representation space reconstruction. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1826–1834.

Wang, D.; Xiao, M.; Wu, M.; Wang, P.; Zhou, Y.; and Fu, Y. 2023. Reinforcement-Enhanced Autoregressive Feature Transformation: Gradient-steered Search in Continuous Space for Postfix Expressions. *arXiv preprint arXiv:2309.13618*.

Wang, X.; Bai, H.; Gong, N.; Ying, W.; Dong, S.; Cui, X.; and Fu, Y. 2025. LLM-ML Teaming: Integrated Symbolic Decoding and Gradient Search for Valid and Stable Generative Feature Transformation. *arXiv preprint arXiv:2506.09085*.

Wang, X.; Wang, D.; Ying, W.; Xie, R.; Chen, H.; and Fu, Y. 2024. Knockoff-Guided Feature Selection via A Single Pre-trained Reinforced Agent. *arXiv preprint arXiv:2403.04015*.

Ying, W.; Bai, H.; Liu, K.; and Fu, Y. 2024a. Topology-aware Reinforcement Feature Space Reconstruction for Graph Data. *arXiv preprint arXiv:2411.05742*.

Ying, W.; Wang, D.; Chen, H.; and Fu, Y. 2024b. Feature selection as deep sequential generative learning. *ACM Transactions on Knowledge Discovery from Data*, 18(9): 1–21.

Ying, W.; Wang, D.; Hu, X.; Qiu, J.; Park, J.; and Fu, Y. 2024c. Revolutionizing Biomarker Discovery: Leveraging Generative AI for Bio-Knowledge-Embedded Continuous Space Exploration. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 5046–5053.

Ying, W.; Wang, D.; Hu, X.; Zhou, Y.; Aggarwal, C. C.; and Fu, Y. 2024d. Unsupervised generative feature transformation via graph contrastive pre-training and multi-objective

fine-tuning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3966–3976.

Ying, W.; Wang, D.; Liu, K.; Sun, L.; and Fu, Y. 2023. Self-optimizing feature generation via categorical hashing representation and hierarchical reinforcement crossing. In *2023 IEEE International Conference on Data Mining (ICDM)*, 748–757. IEEE.

Ying, W.; Wei, C.; Gong, N.; Wang, X.; Bai, H.; Malarkkan, A. V.; Dong, S.; Wang, D.; Zhang, D.; and Fu, Y. 2025. A Survey on Data-Centric AI: Tabular Learning from Reinforcement Learning and Generative AI Perspective. *arXiv preprint arXiv:2502.08828v2*.

Yu, R.; Liu, S.; Yang, X.; and Wang, X. 2023. Distribution Shift Inversion for Out-of-Distribution Prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3592–3602.

Zha, D.; Bhat, Z. P.; Lai, K.-H.; Yang, F.; Jiang, Z.; Zhong, S.; and Hu, X. 2023. Data-centric artificial intelligence: A survey. *arXiv preprint arXiv:2303.10158*.

Zhang, P.; Yan, Y.; Li, C.; Wang, S.; Xie, X.; Song, G.; and Kim, S. 2023. Continual Learning on Dynamic Graphs via Parameter Isolation. *arXiv preprint arXiv:2305.13825*.

Zhu, Q.; Lu, K.; Huo, M.; and Li, Y. 2025. Image-to-image translation with diffusion transformers and clip-based image conditioning. *arXiv preprint arXiv:2505.16001*.

A RL Agents for Data Collection

Leveraging reinforcement learning to explore high-quality and diverse training data. Inspired by the exploitation, exploration, and self-learning abilities of reinforcement learning, our idea is to view an RL agent as a training data collector in order to achieve volume (self-learning enabled automation), diversity (exploration), and quality (exploitation). Specifically, we design RL agents to automatically decide how to perform feature cross and generate new feature sets. **Figure 2** shows that the reinforcement exploration experiences and corresponding accuracy will be collected and stored as training data. The training data collector includes:

- 1) **Multiple Agents:** We design three agents to perform feature crossing: a head feature agent, an operation agent, and a tail feature agent.
- 2) **Actions:** In each reinforcement iteration, the three agents collaborate to select a head feature, an operator (e.g., +, -, *, /), and a tail feature to generate a new feature (e.g. $f_1 + f_2$). The newly generated feature is later added to the feature set for the next feature generation. To balance diversity and quality, we employ the ϵ -greedy DQN algorithm. For each step, each agent has a ϵ probability of selecting a feature/operation based on policy and has a $(1-\epsilon)$ probability of selecting a feature/operation. The ϵ will increase over time so that agents can first explore more and then exploit more.
- 3) **Environment:** The environment is the feature space, representing an updated feature set. When three feature agents generate and add a new feature to the previous feature set, the state of the feature space (i.e., the environment) changes. The state represents the statistical characteristics (such as aggregated average mean, and

variance) of the feature subspace. 4) **Reward:** We formulate the reward as the improvement of the performance of the explored feature set on a downstream task on the current iteration, compared with that of the explored feature set on the previous iteration. In this way, we reinforce agents to explore high-quality feature sets. 5) **Training and Optimization:** Our reinforcement data collector includes many feature crossing steps. Each step consists of two stages - control and training. In the control stage, each feature agent takes actions to change the size and contents of a new feature set and compute a reward. This reward is assigned to each participating agent. In the training stage, the agents train their policies via experience replay independently by minimizing the mean squared error (MSE) of the Bellman Equation. 6) **Transformed Feature Set-Performance Pairs:** We introduce a hyperparameter to control the length of RL-explored feature set and test each of them with a downstream ML task (e.g., random forest classification) to collect feature set-performance pairs. We select the top 5,000 feature set-performance pairs based on accuracy as training data.

B Overall framework

Algorithm 3: Overall Framework

Require: Original dataset

Ensure: Optimal Transformed Feature set

- 1: Initialize the encoder, evaluator, and decoder
 - 2: Leveraging RL agents to collect high-quality data
 - 3: Construct feature-feature similarity graphs
 - 4: Normalize the collected data
 - 5: Shift-resistant Bilevel Training to train encoder, decoder, and evaluator
 - 6: Identify the optimal embedding
 - 7: Leverage a well-trained decoder to generate the feature transformation operation sequences in an autoregressive manner based on the optimal embedding
 - 8: Denormalize the Optimal Transformed Feature set
-

Algorithm 3 describes the overall process. It begins by initializing three core components: an encoder to represent features, an evaluator to assess transformations, and a decoder to generate transformation operations. RL agents are employed to explore the transformation space and collect high-quality transformed data. These data are used to construct a feature-feature similarity graph, enabling the model to capture structural relationships among features. After normalization, a shift-resistant bilevel training strategy is applied to jointly train the encoder, decoder, and evaluator, ensuring robustness against data distribution shifts. Once training stabilizes, the optimal embedding is selected. Based on this embedding, the decoder generates transformation operation sequences in an autoregressive manner, which are then used to produce the transformed features. Finally, a denormalization step is applied to recover the optimal transformed feature set in the original data space.

C Experimental Setup

Datasets. We perform experiments on 16 publicly available datasets from UCI (Dua and Graff 2017) and OpenML (Vanschoren et al. 2013), which are widely used to verify tabular learning methods. To evaluate the robustness of SAFT against distribution shifts, we iteratively generate training and testing sets. Starting from the first feature, we split the dataset into 80% for training and 20% for testing based on their observation index. Using the Kolmogorov-Smirnov test (Berger and Zhou 2014) at a 95% confidence level, we identify distribution shifts. If detected, we finalize the training and testing sets with allocations of 80% and 20%. If no shift is detected, we move to the next feature. If no shift is detected across all features, we exclude the dataset.

Baselines. We compare our framework with 8 widely-used feature transformation methods: 1) **Random Generation (RDG)** randomly produces feature-operation-feature transformations to generate a new feature space; 2) **Essential Random Generation (ERG)** initially expands the feature space by applying operations to each feature, and then chooses the essential features as the new feature space; 3) **Latent Dirichlet Allocation (LDA)** (Blei, Ng, and Jordan 2003) refines the feature space to get the factorized hidden state via matrix factorization; 4) **AutoFeat Automated Transformation (AFAT)** (Horn, Pack, and Rieger 2020) is an enhanced version of ERG, which repeatedly generates new features and employs multi-step feature selection to identify informative ones; 5) **Neural Feature Search (NFS)** (Chen et al. 2019) generates the transformation sequence for each feature and the entire process is optimized by RL; 6) **Traversal Transformation Graph (TTG)** (Khurana, Samulowitz, and Turaga 2018) formulates the transformation process as a graph and subsequently employs an RL-based search method to find the optimal feature set; 7) **Group-wise Reinforcement Feature Generation (GRFG)** (Wang et al. 2022) employs three collaborative reinforced agents to perform feature generation for feature space refinement. 8) **reinforceMent-enhanced autOregressive feAture Transformation (MOAT)** (Wang et al. 2023) formulates the discrete feature transformation problem as a continuous optimization task, while still assuming the I.I.D. condition. 9) **uNsupervised gENERative feAture Transformation (NEAT)** (Ying et al. 2024d) introduces a contrastive learning framework that optimizes feature transformations without using labels, by aligning representations from original and transformed features through self-supervised objectives. 10) **Evolutionary Large Language Model framework for automated Feature Transformation (ELLM-FT)** (Gong et al. 2025c) formulates feature transformation as a sequential generation task, leveraging few-shot prompting of LLMs guided by a reinforcement-evolved database to enable efficient and generalizable optimization beyond task-specific knowledge.

Evaluation Metrics & Hyperparameters Setting. To control the variance of the downstream model’s impact on evaluation, we apply Random Forests (RF) for both classification and regression tasks. For classification, we use the F1-score, Precision, and Recall as the evaluation metrics. In regression, we assess performance using the following met-

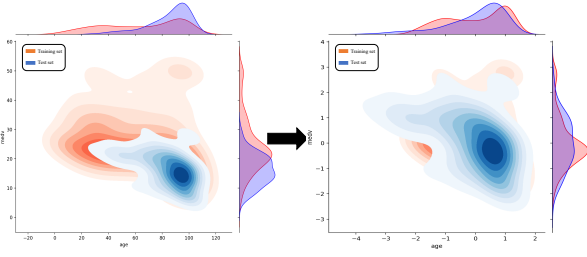


Figure 7: Changes in the joint and marginal distributions of age and MEDV (target) on Housing Boston, comparing datasets before (left) and after (right) normalization.

rics (Hill 2012; Hodson 2022) 1 - Relative Absolute Error (1-RAE), 1 - Mean Absolute Error (1-MAE), and 1 - Mean Squared Error (1-MSE). For all these metrics, a higher value indicates a more effective feature transformation space. 1) **RL collector:** We use the reinforcement data collector to collect explored feature set-feature utility score pairs. The collector explores 512 episodes, and each episode includes 10 steps. 2) **Graph Construction:** The threshold for creating an edge between two features is the 95th percentile of all similarity values. 3) **Our framework:** we map the attribute of each node to a 64-dimensional embedding, and use a 2-layer GNN network and a 2-layer projection head to integrate such information. The decoder is a 1-layer LSTM network, which reconstructs a feature cross sequence. The evaluator is a 2-layer feed-forward network, in which the dimension of each layer is 200. During optimization, we assign α (i.e., 10) as the weight for estimation loss and β (i.e., 0.1) as the weight for reconstruction loss, the batch size is 256, the epochs are 500, and the learning rate range is 0.001-0.0005.

Environmental Settings. Experiments are conducted on the Ubuntu 22.04.3 LTS operating system, Intel(R) Core(TM) i9-13900KF CPU@ 3GHz, and 1 way RTX 4090 and 32GB of RAM, with the framework of Python 3.11.4 and PyTorch 2.0.1.

C.1 Qualitative Analysis

Shift Elimination. We conduct a case study to examine the effect of normalization in reducing distribution shifts between training and test sets. **Figures 7 and 8** show the distributional changes in the Housing Boston and Airfoil datasets, respectively. After normalization, the joint and marginal distributions of age and MEDV (target) become significantly aligned across the training and test sets. In the Airfoil dataset, while the joint distribution of delta and SSPL (target) remains relatively stable, the marginal distributions are better aligned. These results confirm that normalization effectively mitigates distribution shift between training and test data.

Feature Importance. We select the top 10 most important features—measured by mutual information with the target—from the original training set, test set, and the transformed feature space generated by SAFT on the openml.616 dataset. **Figure 9** shows that important features differ significantly between training and test sets, highlighting the impact of distribution shift and the unreliability of directly transfer-

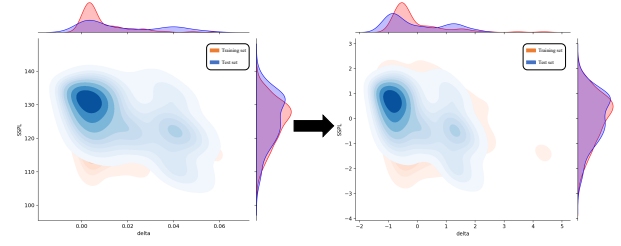


Figure 8: Comparison of joint and marginal distributions of delta and SSPL (target) on the Airfoil dataset before (left) and after (right) normalization, for both training and test sets.

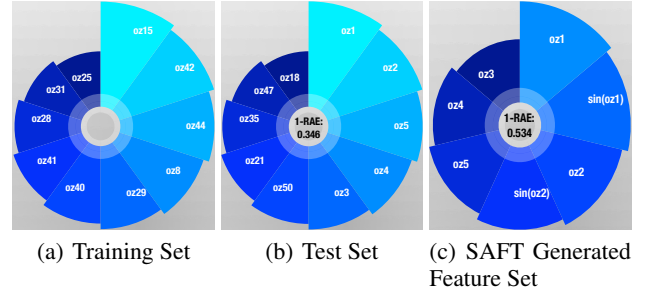


Figure 9: Comparison of the most important features in the original feature space of training and test set, and the SAFT generated feature space.

ring feature importance across domains. Notably, SAFT selects only 7 features (including 2 newly generated ones), yet improves ML model performance by 29.24%. This demonstrates that SAFT can generate effective and transferable features by capturing robust and shift-invariant representations.

C.2 Time and Space complexity comparisons with baselines

We compare the computational cost of SAFT with baseline methods across six datasets. **Table 3** shows methods like RDG, LDA, NSF, and TTG are faster, but they perform worse due to their inability to handle distribution shifts. In contrast, SAFT achieves better accuracy with comparable or lower cost than GRFG and MOAT. Notably, feature transformation is not a major bottleneck in most pipelines, and our method offers significant time savings over manual feature engineering. While RL-based data collection introduces overhead, it is conducted offline and asynchronously. The additional cost stems from collecting high-quality data and training sequence models over complex feature operations. **Table 4** shows SAFT maintains a small and stable memory footprint across datasets. This is enabled by the EOG framework, which embeds variable-length feature sequences into fixed-size vectors, thereby keeping the parameter size constant as the data grows. These results indicate that SAFT scales well both in time and space, maintains competitive efficiency, and ensures scalability for large-scale applications.

Table 3: Time cost comparisons with baselines.

Dataset	RL Data Collection (min)	RDG(s)	LDA(s)	ERG(s)	AFAT(s)	NFS(s)	TTG(s)	GRFG(s)	MOAT(min)	SAFT(min)
SpectF	33.7	3.0	0.3	9.5	3.8	8.3	9.1	282.1	156.0	142.1
Wine Quality White	114.7	30.8	0.5	29.4	4.2	24.7	26.4	373.2	195.8	39.9
Wine Quality Red	36.5	7.4	0.6	13.9	3.6	9.2	10.3	182.2	107.5	43.1
openml_616	156.6	27.5	0.2	55.9	2.6	18.1	24.6	936.5	430.1	160.7
openml_618	239.6	67.7	0.2	82.4	3.0	30.9	38.7	821.6	373.8	95.9
openml_637	171.3	28.1	0.2	55.8	2.8	19.3	25.1	731.7	300.9	147.9

Table 4: Space complexity comparisons with baselines.

Dataset	RL Data Collection (MB)	RDG(MB)	LDA(MB)	ERG(MB)	AFAT(MB)	NFS(MB)	TTG(MB)	GRFG(MB)	MOAT(MB)	SAFT(MB)
SpectF	0.19	0.13	0.07	0.14	0.08	0.14	0.13	1.04	0.14	0.16
Wine Quality White	0.21	0.15	0.08	0.16	0.09	0.15	0.14	18.70	0.13	0.48
Wine Quality Red	0.19	0.14	0.07	0.15	0.08	0.15	0.15	6.14	0.13	0.19
openml_616	0.20	0.14	0.07	0.16	0.08	0.16	0.14	1.94	0.14	0.17
openml_618	0.23	0.15	0.09	0.16	0.11	0.17	0.14	3.84	0.14	0.23
openml_637	0.23	0.16	0.10	0.17	0.11	0.17	0.15	1.94	0.14	0.17

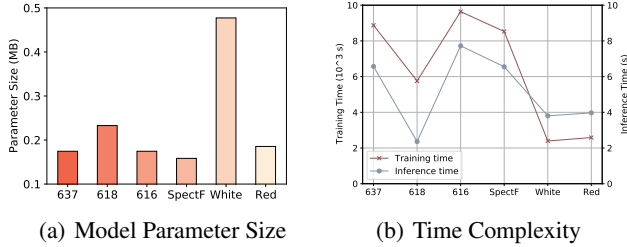


Figure 10: Complexity analysis results (a) model parameter size and (b) Time Complexity.

C.3 Complexity Analysis of SAFT

We investigate the space and time complexity, by comparing model parameter size, training time, and inference time across six datasets. **Figure 10** demonstrates the results. We can observe that the training and inference times for openml_616, openml_618, openml_637, and SpectF are significantly higher than Wine Quality Red and Wine Quality White. The main reason is that, with an increasing number of features, the model has to exert more effort to capture the relevant representation, construct a robust embedding space, and generate the optimal transformed feature set. However, the inference time is relatively low. Therefore, after the convergence of SAFT, we can quickly adapt to the unseen dataset with different distributions, making it practical for dynamic real-world scenarios. Moreover, SAFT upholds a small parameter size. The potential reason is that our GNN-based encoder can sample the most essential knowledge of the feature space to enhance the scalability and robustness of SAFT.