Feature Selection and Generation through Reinforcement Learning (RL) and Symbolic Reasoning

Srihari Tadala
Portland State University
stadala@pdx.edu

1 Abstract

Feature engineering is a vital stage in machine learning pipelines that greatly affects the performance, interpretability, and general efficacy of models. Filter, wrapper, and embedded techniques are common ways to choose and change features, but they often need manual heuristics and subject knowledge. They also don't work well in environments with a lot of dimensions and complexity. Recent studies have investigated automated methods that make use of large language models and reinforcement learning in order to overcome these constraints. A comprehensive and critically synthesized survey of state of the art works covering RL-based feature selection, RL-driven feature generation, and LLM-guided feature optimization is presented in this paper.

Three main paradigms of methodology are identified. In the first, feature selection is framed as a cooperative or guided decision making problem using interactive and multi-agent reinforcement learning techniques. These techniques allocate agents to features and maximize long-term rewards according to domain-specific significance, redundancy, or model accuracy. Combinatorial Multi-Armed Bandits (CMAB), a computationally lightweight alternative that provides scalable and effective feature selection with little learning overhead, is part of the second paradigm [1]. For the third group, LLMs are used to either learn successful reward functions or make new features. They do this by using reasoning-based prompts, external knowledge bases, and prototypical alignment.

This work also address open challenges in bias control, compute overhead, and generalization to unseen domains as well as underexplored gaps including the need of hybrid frameworks combining RL's exploration efficiency with LLMs's semantic reasoning.

2 Introduction

In machine learning, the building of the input feature space through either selection of relevant features or creation of new ones is often more impactful than model architecture or training method. Though conventional techniques including filter, wrapper, and embedded approaches have great success, they have shortcomings including poor scalability, weak interpretability, and dependence on hand heuristics [5]. Demand for automated, adaptive, and explainable feature engineering techniques has grown significantly as datasets get more highly dimensionally, sparse, and multimodally.

Recent studies have tackled this difficulty by characterizing feature optimization as a learning problem unto itself. Using task specific reward functions, reinforcement learning (RL) techniques model features as decision making agents that over time maximize selection strategies. Some effectively explore and exploit subsets of features using lightweight variants including multi-armed bandits (MAB)

[1]. Though they provide adaptability and flexibility, these approaches might find it difficult to generalize outside the numerical domain or with semantic understanding.

Large language models (LLMs) have shown an increasing importance in feature generation and evaluation concurrently by using natural language reasoning and domain knowledge. LLM-based systems can suggest relevant, explainable features informed by text metadata, external corpora, or human preferences by means of tools including Tree-of- Thought prompting, Retrieval-Augmented Generation (RAG), and few-shot reward modeling [2, 3].

Focusing on techniques that automate feature selection or generation using RL and LLMs, this paper offers a consistent survey of various works across several paradigms. We group them by architectural and methodological themes instead of summarizing every one separately, compare their design trade-offs, and examine their strengths in useful applications. This paper point up important gaps such as fairness, cost-efficiency, and lack of hybridization.

3 Previous Work

Previously, feature engineering has been hands-on manual process based on statistical heuristics, domain knowledge, and iterative processes based on error. Filter, wrapper, and embedded methods were first used to try to automate this work, but they weren't very flexible in environments that were changing quickly or had a lot of dimensions. Using reinforcement learning (RL) and information-theoretic formulations to sequentially choose subsets under reward signals linked to model accuracy and redundancy, recent work has reinterpreted feature selection as a learning problem. Though they often operate in a black-box with limited post-hoc interpretability, techniques like INVASE [6] and L2X [7] advanced this line by introducing instance-wise feature selection using actor-critic and mutual information approximations.

Many studies have also extended RL-based automation outside of selection into generation. Although Khurana et al.[8] put forth one of the first RL-based AutoFE models, their system needed manually created transformation templates and lacked semantic understanding. More recent efforts like AutoML-Zero [9] aim toward learning feature construction logic from scratch, but they are computationally demanding and lack the reasoning framework required for interpretable output. By combining structured reasoning with neural operators, neural-symbolic systems such as AutoFeature [10] seek to close this distance.

Leveraging domain knowledge that exists in natural language, large language models (LLMs) provide a semantic abstraction layer to this space. Many LLM based generators, on the other hand depend just on task-level accuracy as the indication of success. According to this paper, this kind of input is insufficient thus feature traceability and redundancy should also serve as guiding principles for generation. In order to provide both semantic depth and optimization rigor, hybrid systems where LLMs propose and RL agents verify are becoming increasingly crucial.

4 Methodologies

The recent surge in automated feature engineering research reflects a transition away from static, manual heuristics toward models that can reason, adapt, and learn transformations or selections. Across the works surveyed, we identify three major methodological archetypes that define how features are optimized: (i) reinforcement learning for feature selection, (ii) reinforcement learning for feature generation, and (iii) LLM-guided feature optimization. These categories reflect not just different architectures, but fundamentally distinct views on how to represent decision processes over the feature space.

Using reinforcement learning (RL) for feature selection, the first group presents the work as a sequential decision-making issue. In this context, the aim is to train agents capable of determining which features should be maintained or deleted so optimizing downstream model performance. Every feature, for instance, is modeled as an independent agent in the multi-agent RL framework proposed by Liu et al [5] receiving feedback in the form of task accuracy, redundancy penalties, and other reward signals. Gao et al.[11] build on this with external direction from KBest filters and decision trees. Differentiating "hesitant" from "assertive" agents improves interpretability and convergence by providing the latter focused guidance during early training rounds. Li et al. [1] model the problem using combinatorial multi-armed bandits (CMAB), so avoiding the complexity of full reinforcement learning by a lighter approach. This formulation lets fast and scalable feature selection by including built-in mechanisms for balancing mutual information-based relevance and redundancy.

By means of mathematical transformations, the second group of works generates new features rather than selecting ones. These approaches are driven by the belief that hidden structure in the data can be revealed by fresh representations created by operations including addition, logarithmic transformation, or interaction. Three cascading RL agents consecutively choose two feature groups and an operation to apply between them to introduce GRFG, a group-wise feature generator. Mathematical similarity and mutual information direct these actions. InHRecon, proposed by Zhang et al. [12] creates a hierarchical agent structure whereby transformation, feature targeting, and operation selection are modularized. Their model guarantees that the produced interactions reflect real second-order effects by using H-statistics, so allowing more interpretability.

For semantic reasoning and text-informed feature engineering, the third methodological class turns away from numerical modeling and instead employs large language models (LLMs). Still, LLMs serve several purposes across these systems. In Large language model based Feature Generation [2] and Text Informed Feature Generation [3],the LLM mostly functions as a feature generator suggesting changes depending on stimuli, metadata, or acquired knowledge. Proto-RM [4] learns preference-aligned scoring systems from minimal input using LLMs as part of the reward modeling pipeline instead. Understanding their capacity and constraints depends on this difference between *LLMs as agents* (actively generating features) versus *LLMs as tools* (evaluating or aligning outputs). While TIFG adds retrieval to ground the LLM's proposals, LFG uses Tree-of- Thought prompting with Monte Carlo Tree Search (MCTS) to refine generation. Conversely, Proto-RM substitutes human-aligned prototypical reasoning for conventional reward shaping. Evaluating scalability, reasoning depth, and deployment costs should take these functional roles into account.

These methodological clusters offer a more relevant prism for comparison than do paper-by–paper summaries. Every group shows trade-offs in generalization, interpretability, computation cost, and applicability to practical problems. Although sensitive to reward shaping, RL-based selectors are modular and light weight. Though they can be computationally costly, generative agents efficiently record interactions. Although they inject domain reasoning, LLM-guided models present difficulties with repeatability and bias reduction.

4.1 RL-Based Feature Selection

Reinforcement learning-based automated feature selection is a logical progression from static filter-based or wrapper-based techniques to dynamic, feedback-driven selection. The main concept is to represent the process of choosing a feature subset as a sequential decision-making task in which the learner observes a state (the chosen feature subset) and is rewarded with delayed rewards (such as downstream accuracy) when they finish it. This area gives rise to three unique approaches, each of which tackles the issues of reward sparsity, interpretability, and scalability.

Reward Formulation: Across most RL-based selectors, the reward is commonly structured as a

weighted combination of accuracy, redundancy, and relevance [5, 1]:

$$R = \lambda_1 \cdot \text{Accuracy} - \lambda_2 \cdot \text{Redundancy} + \lambda_3 \cdot \text{Relevance} \tag{1}$$

Mutual information (MI) is commonly used in computation of redundancy and relevance. In particular, redundancy penalizes overlapping features in the chosen subset, whereas relevance favors features consistent with the target label.:

$$\text{Redundancy} = \frac{1}{|S|^2} \sum_{i,j \in S} I(f_i; f_j) \tag{2}$$

$$\text{Relevance} = \frac{1}{|S|} \sum_{i \in S} I(f_i; y) \tag{3}$$

These metrics help guide the learning process toward informative and diverse subsets, balancing model performance and interpretability.

In multi-agent reinforcement learning, the first representative work models each as an autonomous agent. Every agent learns a policy by a shared environment evaluating the combined subset on a downstream activity and makes binary decisions such as select or drop. Particularly in early peroid, this arrangement suffers from delayed and sparse rewards even if it is intrinsically parallelizable and encourages distributed exploration. The authors use group-level normalisation and correlation aware reward components to lower feature redundancy and so help to mitigate this.

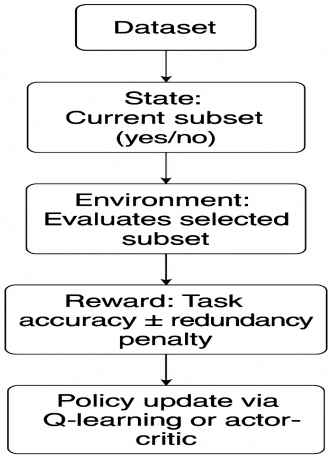

Figure 1: Reinforcement learning loop for automated feature selection.

Upper Confidence Bound (UCB): The CMAB approach uses UCB to balance exploration and exploitation. The reward estimate for feature arm i at time t is computed as [1]:

$$\text{Score}_i = \mu_i + \sqrt{\frac{2 \log t}{n_i}} \tag{4}$$

Here, μ_i is the mean reward of arm i, n_i is the number of times it has been selected, and t is the total time steps. This formulation encourages selection of promising but under-explored features.

Building on the limits of pure MARL, a second method uses interaction with conventional models to include external monitoring into the RL loop. Agents in this design are labeled as ”assertive” or

Table 1: Comparison of RL-Based Feature Selection Methods

Method	Agent Architecture	Reward Signal	Strengths	Limitations
MARL	One agent per feature	Global accuracy + redundancy	Flexible coordination of feature agents	Sparse/delayed rewards
Interactive RL	Agents with trainer support	Model-based + task accuracy	Fast convergence, interpretable decisions	Requires external model feedback
CMAB (CUCB, GFS)	Feature-as-arm	MI + accuracy-based reward	Low compute, strong baseline performance	Limited to additive feature interactions

"hesitant" based on their trust levels. Hesitant agents get help from traditional trainers like decision trees or statistical filters. Particularly in low-data conditions, this yields faster convergence, better selection quality, and more stable learning. The system dynamically moves from supervised guidance to autonomous policy learning, so fusing modern RL flexibility with classical interpretability.

Unlike the agent-heavy formulations above, a third direction frames each feature as an independent arm [1], so simplifying the decision structure by combinatorial multi-armed bandits (CMAB). Different from complete RL methods, these techniques do not reflect environment state or long-term policy trajectories. Rather, CMAB maximizes feature subset selection based on short-term reward estimates—typically via generative Beta sampling or confidence upper bounds. Although CMAB models are substantially faster to train, easier to interpret, and more robust in low-resource or latency-sensitive environments, their lack of state transitions causes CMAB to be less expressive than deep RL in sequential decision problems. They also simplify tuning and honor reward design effort. CMAB is still a good choice for tabular datasets where feature independence is assumed or accepted even if it lacks the adaptability to model inter-feature dependencies across timesteps. Much of the differences in method design between MARL and bandit-based approaches are driven by this contrast in expressiveness and efficiency. These models differ substantially in agent architecture, reward granularity, and computational requirements. A comparative summary is provided in Table 1.

4.2 RL-Based Feature Generation

While feature generating techniques synthesize new attributes by mathematical or statistical transformations, feature selection techniques concentrate on selecting an ideal subset of current features. A methodical approach to automatically guide agents across operations including addition, multiplication, or log transformations is offered by reinforcement learning (RL). These methods preserve a degree of interpretability in addition to enhancing model performance by tracking each produced feature back to its original constituents.

Group-wise Reinforced Feature Generation (GRFG) [12] presents a representative method in this regard and generates a cascade of three RL agents. While the third agent choose a transformation operation (e.g., addition, logarithmic transform, or cross-product), the first two agents choose sets of original features. The group-wise interaction of GRFG is novel it combines two feature groups instead of aggregating two individual features so producing several new features in one step. This layout accelerates training and enhances reward feedback quality. Mutual information and cosine similarity guide the agents to guarantee both relevance and diversity.

By means of a hierarchical reinforcement learning framework, InHRecon [13] breaks out feature generation into three sequential decisions: choosing the transformation operation, spotting the first feature, and subsequently the second. The framework uses H-statistics to estimate the strength of second-order interactions and feature type checks numeric vs categorical to guarantee appropriate operations. These design decisions are especially fit for fields where higher-order interactions are otherwise difficult to find manually and expert knowledge is rare. InHRecon balances interpretability with generalization

power by assessing both transformation validity and interaction quality.

Feature Interaction Strength: To measure the second-order interaction between features, InHRecon uses Friedman's H-statistic [13]:

$$H^2 = \frac{\mathbb{V}\left[\mathbb{E}[Y \mid X_1, X_2] - \mathbb{E}[Y \mid X_1] - \mathbb{E}[Y \mid X_2]\right]}{\mathbb{V}[Y]} \tag{5}$$

This quantifies the proportion of label variance attributable to interaction effects, ensuring that the generated features reflect meaningful dependencies rather than additive noise.

Both GRFG and InHRecon aim to maximize downstream model performance through structured and interpretable feature construction. However, their differences in agent orchestration, guidance signals, and feedback granularity yield unique trade-offs, summarized below in Table 2.

Table 2: Comparison of RL-Based Feature Generation Methods

Method	**Agent Design**	**Feature Construction Strategy**	**Guidance Signal Used**	**Strengths**	**Limitations**
GRFG	Cascaded agents (Group1 $\rightarrow$ Op $\rightarrow$ Group2)	Crosses feature groups using selected operation	Mutual info + Cosine similarity	Efficient multi-feature generation; better reward signal	Requires clustering; sensitive to group quality
InHRecon	Hierarchical agents (Op $\rightarrow$ f1 $\rightarrow$ f2)	Builds pairwise interactions through staged ops	H-statistics + Type-aware rules	Fine-grained control over feature interaction	More sequential steps; higher agent coordination

In GRFG, the term **cascaded agents** refers to a sequential reinforcement learning setup where each agent makes a decision that feeds into the next stage of the feature generation pipeline.

- Here **Group1** and **Group2** refer to clusters or subsets of original features selected by the first two agents.
- **Op** represents the operation (e.g., addition, multiplication, log, etc.) selected by the third agent to apply elementwise between Group1 and Group2.

This design enables batch generation of cross features such as `log(Age × Income)` or `CreditScore / SpendingRate`, while maintaining modular control over each decision point.

In contrast, InHRecon employs a **hierarchical agent structure**, where:

- The first agent selects a valid transformation operation (**Op**) appropriate to the feature types (Example numerical, categorical).
- The second and third agents choose specific features (**f1** and **f2**) to serve as operands in the interaction.

This staged design allows finer-grained control over feature construction and supports second order interaction modeling guided by statistical criteria like H-statistics.

Table 3: Comparison of LLM-Guided Feature Generation and Evaluation Methods

Method	Core Role	Toolset / Strategy Used	Strengths	Limitations
LFG	Feature generation via reasoning	Tree-of-Thought + MCTS	Reasoned transformations; minimal labeled data	Sensitive to prompt quality; compute-intensive
TIFG	Knowledge-informed generation	Retrieval-Augmented Generation	Domain-aware features; grounded in external text	Depends on corpus quality; hard to reproduce
Proto-RM	Preference-aligned evaluation	Prototypical Reward Modeling	Data-efficient reward learning for LLM outputs	Requires curated comparisons; less generation

4.3 LLM-Guided Feature Generation

Large language models (LLMs) present a fundamentally different paradigm—semantic reasoning while reinforcement learning based methods explore feature space via reward-optimized exploration. LLM-guided systems propose, evaluate, or rank new features using language, metadata, and external knowledge rather than depending just on numerical correlations or task-specific rewards. These models shine in bringing contextual awareness to tabular data and in encoding expert like logic. Three recent works show three distinct roles LLMs can perform preference based reward modeling (Proto-RM), knowledge-informed augmentation (TIFG), and feature generation (LFG).

Every feature transformation is seen by the LFG framework [2] as a reasoning task. Tree-of-Thought (ToT) prompting is used by several LLM agents to create candidate features, explain their reasoning, and edit depending on downstream comments. Layer on top a Monte Carlo Tree Search (MCTS) mechanism to help choose the best thinking paths depending on model performance (e.g., F1 score or accuracy). This approach combines symbolic reasoning with probabilistic search to enable LLMs to investigate several transformations—e.g., log(income), weight × age—while avoiding redundant or incoherent proposals. With little labeled supervision, LFG performs well across models including KNN, MLP, and Random Forest.

TIFG [3] expands on the theory that real-world characteristics sometimes have semantic connections hidden in their names or dataset metadata. Retrieval-Augmented Generation (RAG) searches Wikipedia or other corpora for task-relevant concepts, then asks the LLM to synthesize features like ”density = population / land area,” or ”BMI = weight / height2.” Financial, and healthcare datasets where domain knowledge is not ingrained in the raw numerical data benefit especially from this method. TIFG supports multi-round thinking and justification, so producing features with both interpretability and novelty.

Proto-RM [4] turns attention from feature generation to feature evaluation and alignment. It trains a prototypical reward model whereby prototype vectors are generated from feedback from a small number of human comparisons (selected against rejected features or outputs). Then based on their resemblance to these acquired prototypes, new examples get scores. Especially for LLM output alignment, preference modeling, or safety tuning in chat systems, this model enhances reward accuracy in low-label settings.

Prototype-Guided Loss Function: Proto-RM optimizes a composite loss to align model outputs with human preference [4]:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{reward}} + \rho_d \cdot \mathcal{L}_{\text{diversity}} \tag{6}$$

In this formula, $\mathcal{L}_{\text{reward}}$ encourages the model to assign higher scores to preferred outputs typically based on pairwise comparisons, such as human preference rankings or correct classifications. The $\mathcal{L}_{\text{diversity}}$ term promotes variation among learned prototype representations, which helps prevent mode collapse and improves generalization to diverse input types. The hyperparameter ρ_d controls the trade-off between alignment precision and representation spread.

5 Applications Across Domains

Implementing the diversity in design between RL-based and LLM-guided feature engineering techniques directly results in different strengths in practical fields. Every approach from policy and preference modeling to healthcare and finance showcases fit for particular operational restrictions, data structures, and interpretability requirements.

Particularly suited for healthcare data pipelines are joint feature-instance selection models including the one presented in [14]. Along with redundant features, these databases sometimes feature noisy or irrelevant samples that is, mislabeled or outlier patient records. Proposed in DAIRS, the dual-agent reinforcement learning system helps both axes features and instances to be selectively filtered, so enhancing generalization and clinical robustness. Furthermore in line with regulatory needs for interpretability in healthcare applications is the ability to track selection decisions.

Transformer-based architectures for feature weighting, such those applied in high-cardinality tabular domains, are naturally extensible to financial risk modeling and fraud detection chores. These settings call for dynamic reweighting of over time credit history, transaction patterns, and behavior metrics. Although the transformer paper itself was not included in this survey, feature relevance modeling using structured agents as explored in GRFG [12] offers equivalent value for these kinds of financial tabular tasks.

Text-enhanced, semantically grounded features often help policy and insurance-related usage cases. TIFG [3] shows this by producing features informed by external corpora such as Wikipedia or internal documentation, so enabling applications including risk profiling or policy scoring from both numerical attributes and textual descriptions. When domain knowledge is buried in unstructured forms, these methods especially help.

Lastly, methods like Proto-RM [4] help preference aligned tasks including chatbot tuning and reinforcement learning from human feedback (RLHF) benefit from Learning strong reward models from few samples is absolutely crucial in low-label or sparse-feedback systems. Prototypical based architecture of Proto-RM is especially pertinent in dialog agents, educational platforms, and interactive ranking systems where feedback is often implicit or noisy.

6 Challenges

Although automated feature optimization shows great potential, present techniques have several important problems that restrict their practical relevance. Lack of standardized benchmark datasets especially meant for assessing feature generation and selection techniques is one recurring restriction. Most techniques evaluate on UCI tabular datasets or domain-specific collections, which lack consistent ground truth for produced feature quality. Fair comparison of algorithms across fields and research groups becomes challenging without generally agreed upon benchmarks.

The close integration of reinforcement learning with language model-based thinking also reveals another significant disparity. Few systems really combine structured exploration and optimization from RL methods with semantic generalization and knowledge retrieval from LLMs. Usually choosing one paradigm, current models neglect the advantages of the other. Hybrid models in which LLMs suggest or defend changes while RL agents validate and improve those decisions over time could help future work.

Additionally mostly omitted are bias and fairness. Particularly in text-informed or feedback-guided environments like TIFG [3] or Proto-RM [4], produced features may embed implicit society biases from outside corpora or training distributions. In high-stakes fields including hiring, lending, or healthcare, this begs grave questions. One could find great direction in including adversarial debiassing or fairness

constraints into the reward model.

Computational cost is ultimately still a non-trivial barrier particularly for models such as GRFG [12] and LFG [2], which depend on cascaded agents or several rounds of LLM reasoning. Low-resource settings or time-sensitive applications could find these methods unworkable. Policy distillation, curriculum learning, or surrogate modeling can help to lower overhead and increase their deployability. These challenges taken together highlight the need of more modular, efficient, and ethically conscious frameworks that can adapt across sectors while keeping interpretability and computational practicality.

7 Future Work

Recent developments in automated feature engineering point toward more integrated, semantically aware systems combining the generative reasoning of large language models with the structured optimization of reinforcement learning. Although RL-based approaches have made progress in feature selection [5, 1] and transformation [12, 13], they frequently lack semantic context. Conversely, LLM-guided systems such as TIFG and LFG [3, 2] can domain -aware reasoning but demand expensive feedback cycles. Future studies could investigate hybrid architectures whereby RL agents validate features proposed by LLMs using reward-guided learning based on domain knowledge. Although it needs extension to multimodal tasks, prototype-based alignment [4] shows promise in low-label environments. Still vitally important are addressing generalization across datasets, energy cost, and bias propagation. For interpretable and effective feature space construction, the next generation of methods must ultimately unite symbolic reasoning, statistical learning, and ethical constraints.

8 Conclusion

The present work investigated recent advancements in automated feature space building under the prism of reinforcement learning and large language models. The study arranged methods into three methodological archetypes one is RL-based selection, second is RL-guided generation, and third is LLM-driven semantic reasoning instead of viewing present efforts as isolated. Every class offers different trade-off between generalization, scalability, and interpretability. While generative RL agents expose higher-order interactions, multi-agent systems and bandit-based selectors offer modular control over relevance and redundancy. Though they usually only use task-level feedback and incur large computational overhead, LLM-guided approaches improve semantic alignment even more.

Many of these techniques remain domain fragile, sensitive to hyperparameter tuning, and computationally expensive in practice even if their predictive performance and adaptation show clear improvements. Few systems today combine symbolic reward modeling with knowledge grounded feature construction. Bias, traceability, and evaluation consistency are also understudied. To automate feature design across multiple fields, a more comprehensive approach is required, one that makes use of statistical decision procedures and semantic abstraction. Future feature engineering will be shaped by hybrid systems that combine semantic reasoning and symbolic exploration, fusing the intuition of LLMs with the rigor of RL.

References

[1] Bo Li, Cheng Wang, Kunpeng Liu, Wei Wang, and Yi Chang. Multi-armed bandit based feature selection. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. Early Access.

[2] Kunpeng Liu, Shuo Zhang, Yuanwei Gao, Yi Chang, and Wei Wang. Dynamic and adaptive feature generation with large language models. arXiv preprint arXiv:2304.10561, 2023.

[3] Yuanwei Gao, Kunpeng Liu, Shuo Zhang, Wei Wang, and Yi Chang. Tifg: Text-informed feature generation with large language models. arXiv preprint arXiv:2308.10184, 2023.

[4] Kunpeng Liu, Shuo Zhang, Tengfei Ma, Yi Chang, and Wei Wang. Prototypical reward network for data-efficient model alignment. arXiv preprint arXiv:2305.09919, 2023.

[5] Kunpeng Liu, Yuanwei Gao, Chen Wu, Tengfei Ma, Wei Wang, Yi Chang, and Jun Ma. Automated feature selection: A reinforcement learning perspective. *IEEE Transactions on Knowledge and Data Engineering*, 34(11):5676–5689, 2022.

[6] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. Invase: Instance-wise variable selection using neural networks. *ICLR*, 2020.

[7] Jianbo Chen, Le Song, Martin J Wainwright, and Michael I Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *ICML*, 2018.

[8] Uday Khurana, Deepak Turaga, Horst Samulowitz, and Srinivasan Parthasarathy. Feature engineering for predictive modeling using reinforcement learning. In *AAAI*, 2018.

[9] Esteban Real, Sherry Aggarwal, Yanping Huang, and Quoc V Le. Automl-zero: Evolving machine learning algorithms from scratch. In *ICML*, 2020.

[10] Linyuan Zhou, Hao Tang, Xinyu Wang, and Mengdi Zhang. Autofeature: Neural-symbolic reasoning for automated feature construction. In *KDD*, 2023.

[11] Yuanwei Gao, Kunpeng Liu, Shuo Zhang, Tengfei Ma, Yi Chang, and Wei Wang. Interactive reinforced feature selection with decision tree in the loop. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM)*, pages 2589–2598, 2021.

[12] Yuhao Zheng, Kunpeng Liu, Yuanwei Gao, Tengfei Ma, Yi Chang, and Wei Wang. Group-wise reinforcement feature generation for optimal and explainable representation space reconstruction. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2546–2552, 2022.

[13] Shuo Zhang, Kunpeng Liu, Tengfei Ma, Wei Wang, and Yi Chang. Feature interaction aware automated data representation transformation. *IEEE Transactions on Knowledge and Data Engineering*, 2022. Early Access.

[14] Bo Li, Yuanwei Gao, Kunpeng Liu, Yi Chang, and Wei Wang. Feature and instance joint selection: A reinforcement learning perspective. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2553–2559, 2022.

[15] Shuo Zhang, Kunpeng Liu, Tengfei Ma, Yi Chang, and Wei Wang. Tabular feature weighting with transformer. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 3811–3820, 2022.

[16] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *ACL*, 2020.