Symbolic Feature Engineering Using Spearman-Guided Construction and Mutual Information-Based Selection for Classification Tasks

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—Feature engineering plays a vital role in improving classification performance in machine learning. This paper proposes a novel feature engineering framework that leverages genetic programming to construct new, potentially complex features that are difficult to discover using traditional feature engineering methods. These constructed features are first evaluated using Spearman correlation to measure their relationship with the target variable. A final feature set, comprising both original and constructed features, is then selected based on mutual-information based framework, Maximize Relevance with Minimum Redundancy using KSG Estimator (MRwMR-BUR-KSG). Experimental results on 6 datasets from PMLB demonstrate that the proposed approach, when combined with a classification algorithm, achieves classification performance comparable to that of established feature engineering libraries. These findings highlight the potential of integrating symbolic representation learning with rank-based and information-theoretic criteria for effective feature engineering in noisy or structured data settings.**

***Index Terms*—genetic programming, feature engineering, classification**

I. Introduction

Feature engineering, a crucial step in supervised learning, involves two main phases: feature construction and feature selection. In the feature construction phase, we create new, more informative features, often by mathematically combining existing ones, to improve the model's predictive accuracy and minimize errors by capturing the complex inherent relationships. Feature selection, on the other hand, focuses on identifying the most relevant features while eliminating redundant or irrelevant information to improve efficiency and generalization. Selecting an appropriate strategy for feature engineering can be challenging. While existing automated feature engineering libraries streamline the process, they often compromise interpretability and computational efficiency, and their performance is often inconsistent across datasets.

The existing feature engineering approaches used in this study include TabNet, AutoFeat, and LightAutoML. TabNet is a deep learning architecture specifically designed for tabular data, aiming to strike a balance between predictive performance and interpretability. TabNet first processes all input features through shared feature layers to create a common representation. At each decision step, the attentive transformer generates sparse selection masks that determine which features are completely selected or ignored (rather than prioritized with weights). The selected features then pass through step-specific feature transformers for further processing. This sequential feature selection creates an explicit feature usage pattern that can be interpreted as the model's decision process. The feature engineering process in TabNet is fully automated and integrated within its learning architecture.

AutoFeat is another automated feature engineering approach utilized in this study. It automatically generates tens of thousands of non-linear features from the original inputs by applying user-defined transformations and combining feature pairs using mathematical operators such as addition, subtraction, and multiplication. To reduce redundancy, AutoFeat filters out features that are highly correlated with the original inputs or with simpler engineered features. It then performs multi-step feature selection, primarily using L1-regularized linear models to retain only the most informative features [1]. While AutoFeat offers a high degree of automation and interpretability, it is computationally intensive, and in some cases, the performance gains may not justify the time required for processing.

LightAutoML is an open-source library for automated machine learning that is lightweight and supports both data processing and feature selection. It offers three feature selection strategies: no selection, selection based on feature importance, and a stricter method using forward selection [2]. Feature importance is evaluated using a Gradient Boosting Machine (GBM) model. While LightAutoML delivers strong performance overall, its effectiveness decreases on datasets with a high level of noise.

We propose a symbolic feature transformer that leverages Spearman correlation and Maximum Relevance with Minimum Redundancy by Boosting Unique Relevance using KSG Estimator (MRwMR-BUR-KSG) to guide feature construction and selection. Overall, the proposed framework is simple, interpretable, and demonstrates competitive performance com-

pared to existing feature engineering techniques.

II. Materials and Methodology

A. Datasets and parameter settings

This research utilizes six classification datasets from the Penn Machine Learning Benchmarks, comprising of four binary classification datasets and three multi-class classification datasets [3]. Each dataset includes at least one numerical column, and one of them contains noise introduced by PMLB. Detailed information on all six datasets is presented in TABLE I.

The feature construction method is implemented using the Symbolic Transformer from gplearn [4] library. The parameters for the Symbolic Transformer were determined by using nested cross validation to achieve highest performance. The hyperparameter settings used in the experiments are detailed in Table II, while all other parameters remain at their default values. To ensure a fair comparison across methods, the parameters are kept consistent throughout the evaluation. For MRwMR-BUR-KSG implementation, the parameter β is set to 0.1 to balance Unique Relevance (UR) [5].

The proposed approach utilizes a CatBoost classifier for the final classification stage, as it achieved the highest accuracy when combined with the new approach, as shown in Table III. In contrast, the other feature engineering libraries utilize their respective built-in classifiers: TabNetClassifier for TabNet, AutoFeatClassifier for AutoFeat, and TabularAutoML for LightAutoML.

TABLE I
Datasets Overview

Dataset	Features	Instances	Classes
allbp	29	3772	3
Hill_Valley_with_noise	100	1212	2
Hill_Valley_without_noise	100	1212	2
adult	14	48842	2
allhyper	29	3771	4
breast_cancer	9	286	2

TABLE II
Parameter settings

Parameters	Parameter value
Population size	5000, 10000
Function set	+, -, *, $\sqrt{}$, abs, neg, max, min
Number of Generations	10, 20
n_components	10
parsimony_coefficient	0.001, 0.01
metric	spearman
hall_of_fame	100

B. Methodology

Our method is built upon the gplearn symbolic transformer which enables automated generation of complex non-linear feature interactions that are often difficult to identify through traditional methods. The symbolic transformer is guided by a rank-based fitness function, specifically the Spearman correlation, which measures the strength of a monotonic relationship between generated features and the target variable. Unlike Pearson correlation, Spearman can capture both linear and non-linear monotonic dependencies and is more robust to outliers [6], making it suitable for feature construction in varied data conditions.

TABLE III
Comparison in performance between different classification algorithm

Classification Algorithm	Decision Tree	Random Forest
allbp	0.957 ± 0.010	$\mathbf{0.974 \pm 0.006}$
Hill_Valley_with_noise	0.858 ± 0.027	0.889 ± 0.017
Hill_Valley_without_noise	0.997 ± 0.005	0.997 ± 0.005
adult	0.812 ± 0.005	0.852 ± 0.004
allhyper	0.970 ± 0.007	0.979 ± 0.004
breast_cancer	0.587 ± 0.028	0.689 ± 0.070

Classification Algorithm	CatBoost	SVM
allbp	0.967 ± 0.003	0.957 ± 0.008
Hill_Valley_with_noise	$\mathbf{0.892 \pm 0.016}$	0.595 ± 0.027
Hill_Valley_without_noise	$\mathbf{0.999 \pm 0.002}$	0.512 ± 0.027
adult	$\mathbf{0.874 \pm 0.001}$	0.812 ± 0.016
allhyper	$\mathbf{0.980 \pm 0.005}$	0.973 ± 0.003
breast_cancer	$\mathbf{0.731 \pm 0.029}$	0.675 ± 0.037

Because symbolic learning is only effective on continuous data, we first pass our continuous features to the Symbolic Transformer to generate new features. We then use the MRwMR-BUR-KSG algorithm to select the most valuable features from the combined set of original and newly created ones. MRwMR-BUR-KSG is a variant of Maximize Relevance with Minimum Redundancy (MRwMR) with the objective of boosting UR [5]. Boosting UR reduces redundancy in selected features while ensuring their optimality. The MRwMR-BUR-KSG variant calculates unique feature relevance independently of a classifier, which means any classification algorithm can be used for prediction.

The metric is calculated using the given formula:

$$J_{new}(X_i) = (1 - \beta) \times J_{org}(X_i) + \beta \times J_{UR}(X_i)$$

where $J_{org}(X_i)$ is the original MRwMR score and $J_{UR}(X_i)$ is the score for UR. MRwMR-BUR-KSG first picks the feature with the highest $J_{new}(X_i)$ score. It then incrementally adds features that offer the best balance of high relevance to the target and low redundancy with the features already chosen. The MRwMR-BUR-KSG variant calculates unique feature relevance independently of a classifier, which means any classification algorithm can be used for prediction. Relevance is quantified using mutual information, where a higher score signifies a stronger relationship with the target [7]. Redundancy is measured by averaging the mutual information between a candidate feature and all previously selected features. UR is calculated from the conditional mutual information between X_k and target variable, when removed from Ω (We define set

of all features as $\Omega = \{X_k, k = 1, \ldots, M\}$, where M is the number of features.):

$$UR = I(X_k; Y|\Omega \backslash X_k)$$

We experimented with selecting the top 25%, 50%, and 75% of features based on mutual information scores. While selecting more features slightly improved average accuracy (by approximately 1%), the difference was marginal. To balance performance and model simplicity, we selected the top 50% as the default threshold in our experiments. This helps eliminate redundant or irrelevant variables by removing features, resulting in a cleaner and more informative feature set. The final feature set can then be used with a variety of classification algorithms, including tree-based models and symbolic classifiers from gplearn. Figure 1 shows the overall framework of the new approach.

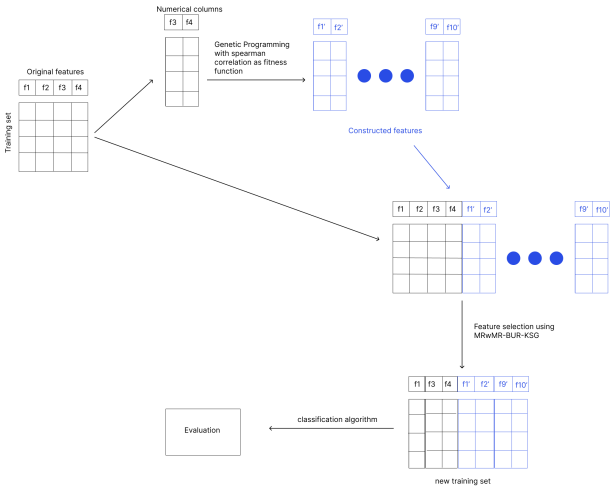

Fig. 1. Overview of the proposed symbolic feature engineering framework

III. Result and Discussion

The results of our proposed approach compared to other feature engineering libraries are presented in Table IV. As shown in the figure, the accuracy of our method is generally within 1% of the competing approaches.

Notably, on the noisy dataset (Hill Valley with noise), our approach achieves the highest accuracy (90%), outperforming the second-best method, LightAutoML, by 17%. This highlights the robustness of our method in handling noisy data. The accuracy of LightAutoML on noisy datasets may be impacted by its reliance on LightGBM as the core machine learning algorithm. Among boosting methods, the GBM algorithm exhibits the highest Equalized Loss of Accuracy (ELA) score, indicating lower robustness to noise compared to other boosting algorithms [8].

While the new approach handles continuous data effectively, it falls short against LightAutoML whenever categorical data is present. LightAutoML builds a richer feature set by analyzing all data types, but the new approach only does transformation on numerical data.

By fixing the parameter, the Symbolic Transformer typically generates interpretable features with simple equation trees (a depth of 2). However, for noisy datasets, these trees can grow up to 9 levels deep, making the resulting features significantly harder to understand as the equation is becoming more complex to differentiate signal and random noise. An evaluation of computational efficiency across six datasets demonstrated that TabNet had the lowest average runtime at 2 minutes. The new approach followed with a runtime of 22 minutes, succeeded by LightAutoML at 38 minutes. Autofeat was substantially slower, with a runtime exceeding one day.

TABLE IV
Comparison of different automated machine learning methods.

Dataset/Method	New Approach	Tabnet	Autofeat	LightAutoML
allbp	0.974	0.962	0.848	**0.975**
Hill_Valley_with_noise	**0.892**	0.494	0.770	0.737
Hill_Valley_without_noise	0.999	0.490	**1.000**	0.757
adult	0.874	0.852	0.694	**0.875**
allhyper	0.980	0.987	0.791	**0.992**
breast_cancer	**0.731**	0.621	0.672	0.707

IV. Conclusions

This paper presents a novel two-stage feature engineering framework as an alternative to existing automated solutions. The approach integrates a Symbolic Transformer, guided by Spearman correlation for feature construction, with mutual information-based feature selection. The resulting feature set is compatible with various classification algorithms. Experimental evaluations demonstrate that the proposed method performs competitively with established libraries and shows notable robustness on noisy datasets. Due to its inherent design, the Symbolic Transformer makes the new approach ineffective for datasets containing many categorical values. Additionally, while this work focuses specifically on the feature construction and selection stages, it does not encompass the full spectrum of preprocessing tasks such as data imputation or normalization. Future work will explore integrating this method into a more comprehensive feature engineering pipeline.

References

[1] F. Horn, R. Pack, and M. Rieger, "The autofeat python library for automated feature engineering and selection," pp. 4–5, 2020. [Online]. Available: https://arxiv.org/abs/1901.07329

[2] A. Vakhrushev, A. Ryzhkov, M. Savchenko, D. Simakov, R. Damdinov, and A. Tuzhilin, "Lightautoml: Automl solution for a large financial services ecosystem," p. 4, 2022. [Online]. Available: https://arxiv.org/abs/2109.01528

[3] R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, and J. H. Moore, "Pmlb: a large benchmark suite for machine learning evaluation and comparison," *BioData Mining*, vol. 10, no. 36, pp. 1–13, Dec 2017. [Online]. Available: https://doi.org/10.1186/s13040-017-0154-4

[4] T. Stein, "gplearn: Genetic programming in python," https://github.com/trevorstephens/gplearn, 2016, accessed: 2024-06-04.

[5] S. Liu and M. Motani, "Improving mutual information based feature selection by boosting unique relevance," pp. 6–11, 2022. [Online]. Available: https://arxiv.org/abs/2212.06143

[6] M. M. Mukaka, "Statistics corner: A guide to appropriate use of correlation coefficient in medical research," *Malawi Medical Journal*, vol. 24, no. 3, pp. 69–71, Sep. 2012.

[7] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical Review E*, vol. 69, no. 6, p. 1, Jun. 2004. [Online]. Available: http://dx.doi.org/10.1103/PhysRevE.69.066138

[8] A. Gómez-Ríos, J. Luengo, and F. Herrera, "A study on the noise label influence in boosting algorithms: Adaboost, gbm and xgboost," in *Hybrid Artificial Intelligent Systems*, F. J. Martínez de Pisón, R. Urraca, H. Quintián, and E. Corchado, Eds. Cham: Springer International Publishing, 2017, p. 277.