Semi-Automated Feature Engineering and Evaluation Framework

Neta Shalit, Amit Karol

March 10, 2025

Abstract

Feature engineering plays a crucial role in machine learning, affecting both model performance and interpretability. However, it is often a manual, time-consuming process requiring domain expertise. This study introduces a **Semi-Automated Feature Engineering** framework designed to streamline feature selection and transformation, reducing human effort while preserving interpretability.

Our method systematically generates features using mathematical transformations, interactions, and statistical aggregations. These features are then filtered based on correlation, variance thresholds, and importance scores computed from SHAP values and Random Forest models. By automating these steps, our approach enhances predictive performance while reducing feature redundancy.

We evaluate the framework on multiple classification and regression datasets. Results show consistent improvements across key performance metrics: accuracy, precision, recall, and F1-score for classification, and R^2, RMSE, and MAE for regression. Compared to **AutoFeat**, a fully automated feature engineering tool, our method produces more interpretable and contextually relevant features. Statistical tests confirm that these improvements are significant in multiple cases.

These findings demonstrate that **semi-automated feature engineering** effectively balances automation with expert-driven insights, offering an interpretable and scalable solution for improving machine learning models. Future work will explore hybrid approaches integrating domain knowledge for further optimization.

1 Problem Description

Feature engineering is a crucial step in the machine learning pipeline, as the quality of input features directly affects model performance. However, traditional feature engineering is time-consuming, iterative, and requires significant domain expertise. Poorly designed features can degrade model accuracy, while redundant or irrelevant features introduce noise and increase computational costs.

Existing automated tools generate numerous transformations but often produce irrelevant or redundant features, increasing overfitting and reducing interpretability. On the other hand, fully manual feature selection is subjective and resource-intensive. The key challenge is **balancing automation with expert-driven feature selection** to ensure that generated features are both meaningful and efficient.

This project focuses on improving **the feature engineering process** by reducing **the manual effort required for feature selection while maintaining control over interpretability and relevance**.

2 Solution Overview

Our framework streamlines feature engineering by automating feature creation, evaluation, and selection, reducing reliance on manual effort while maintaining a data-driven and efficient pro-

cess. It balances automation with user-driven refinements to optimize feature sets for different datasets.

Feature Generation and Transformation New features are systematically generated using the `generate_features()` function:

- **Polynomial transformations** (e.g., squared terms, higher-order interactions).
- **Ratio-based features** (e.g., division between numerical features).
- **Interaction terms** between categorical and numerical variables to enhance model expressiveness.

This function iterates over all numerical feature pairs and applies arithmetic operations (addition, subtraction, multiplication, and division) to create candidate features.

Once candidate features are generated, the framework provides:

- The **total number of newly generated features**.
- A **sample list of new feature names** for transparency.

Feature Filtering and Selection The `evaluate_features()` function removes uninformative features:

- **Correlation analysis** – Removing features with correlation below a user-defined threshold (default: 0.05).
- **Variance thresholding** – Eliminating features with near-zero variance (default: 0.01).
- **Feature importance ranking** – Prioritizing features via SHAP values and Random Forest importance scores using `compute_feature_importance()`.

By filtering weak and redundant features, this process reduces dimensionality, improves model efficiency, and enhances generalizability.

After filtering, the framework reports:

- The **total number of features removed**.
- The **number of new features retained** after filtering.
- A **summary of original vs. generated vs. retained features**.

User-Guided Feature Selection and Adaptive Thresholding Our framework combines automation with user intervention to enhance flexibility and interpretability:

1. **Customizable Thresholds:** Users can define correlation and variance thresholds (default: 0.05 and 0.01) or adjust them per dataset to retain meaningful features.
2. **Feature Importance Review:** A ranked list of features is provided for expert review and domain-specific adjustments.
3. **Manual Feature Exclusion:** Users can remove redundant or irrelevant features before training the final model.
4. **Adaptive Selection Across Datasets:** Thresholds and feature selections can be adjusted dynamically for different datasets and tasks.

5. **Transparent Feature Summary:** The framework reports the total number of generated, removed, and retained features, ensuring a clear overview for decision-making.

By integrating automated feature engineering with user-driven refinements, our approach ensures that feature selection is both optimized and adaptable across diverse datasets, improving model accuracy while maintaining interpretability.

3 Experimental Evaluation

We evaluated our **Semi-Automated Feature Engineering framework** on four datasets, covering both **classification and regression tasks**. Each dataset posed unique challenges, allowing us to assess the **effectiveness and adaptability** of our approach. **Table 1** summarizes the datasets and their corresponding prediction tasks.

Table 1: Datasets and Corresponding Machine Learning Task

Dataset	**Target Variable**	**Task Type**
Cancer Patient Data	Cancer Severity Level	Classification
Amsterdam Rental Prices	Rental Price	Regression
Student Performance	Pass/Fail Status	Binary Classification
Life Expectancy Data	Life Expectancy	Regression

To evaluate our framework, we first **trained a baseline model** using only the original features. We then trained a model incorporating our **engineered features** and compared their performance using **classification metrics** (Accuracy, Precision, Recall, F1-score) and **regression metrics** (R^2, RMSE, MAE).

Since different datasets exhibit different distributions and feature importance patterns, we applied **adaptive feature selection**. The **correlation threshold** (≥ 0.1) and **variance threshold** (≥ 0.05) were manually adjusted per dataset to **eliminate uninformative features** while retaining those that contributed to model performance. This **dataset-specific optimization** ensured that our feature selection process remained flexible, improving both efficiency and interpretability.

3.1 First Dataset: Cancer Patient Data

This dataset includes medical attributes for classifying cancer severity. Our feature engineering approach generated **1,012 new features**, and after filtering based on correlation and variance thresholds, **900 features** were retained. The model with feature engineering showed improved performance, increasing accuracy from **93.5% to 96.8%**. A **paired t-test (t = -12.4158, p = 0.0011)** confirmed the improvement was statistically significant, demonstrating that feature engineering effectively enhanced classification accuracy.

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
Baseline	0.935	0.946	0.935	0.932
With Feature Engineering	0.968	0.970	0.968	0.967

Table 2: Performance comparison for Cancer Patient dataset

3.2 Amsterdam Rental Prices

This dataset includes rental property attributes such as location, guest satisfaction, and room type. Feature engineering generated **840 new features**, and after removing **268 low-correlation and low-variance features**, **579 features** remained. The model with feature engineering improved **R^2 from 0.482 to 0.589**, with reductions in **RMSE (224.57 to 200.03) and MAE (148.29 to 134.65)**. The **paired t-test (t = 1.7833, p = 0.2165)** indicated that the improvement was not statistically significant, possibly due to the relatively small effect size or inherent variability in rental prices that the engineered features could not fully capture. Furthermore, the dataset might already contain strong predictive features, limiting the additional impact of newly generated transformations. Despite the lack of statistical significance, the improved predictive accuracy suggests that feature engineering contributed meaningful insights.

Model	R^2	RMSE	MAE
Baseline	0.482	224.57	148.29
With Feature Engineering	0.589	200.03	134.65

Table 3: Performance comparison for Amsterdam Rental Prices dataset

This dataset predicts whether a student will pass or fail based on academic and demographic attributes. **1,512 new features** were generated, with **296 retained after filtering**. Feature engineering improved accuracy from **99.2% to 100%**, indicating that the additional features helped refine the decision boundaries. The **paired t-test (t = -5.7796, p = 0.0103)** confirmed that this improvement was **statistically significant**. Although the baseline model already had a **high accuracy**, feature engineering further optimized classification, reinforcing key decision patterns and improving interpretability. The **statistical significance** suggests that the engineered features introduced **meaningful refinements** that contributed to a more **robust prediction model**.

Model	Accuracy	Precision	Recall	F1-score
Baseline	0.992	0.983	0.992	0.988
With Feature Engineering	1.000	1.000	1.000	1.000

Table 4: Performance comparison for Student Performance dataset

3.3 Forth Dataset: Life Expectancy

his dataset predicts life expectancy based on health, economic, and demographic factors. **840 new features** were generated, with **681 retained after filtering**. The model with feature engineering showed a **marginal improvement** in **R^2** (0.965 to 0.968) with small reductions in **RMSE** (1.732 to 1.674) and **MAE** (1.167 to 1.085). A **paired t-test (t = 1.8299, p = 0.2088)** indicated that these improvements were **not statistically significant**, likely due to the dataset's already strong baseline performance, leaving little room for further enhancement. Additionally, the reductions in RMSE and MAE were minor, suggesting that while feature engineering contributed to small refinements, its overall impact was limited by the dataset's inherent predictive quality.

3.4 Conclusions from Dataset Evaluations

Across all datasets, our Semi-Automated Feature Engineering framework consistently improved model performance. The **classification tasks** (cancer severity, student performance) saw no-

Model	R^2	RMSE	MAE
Baseline	0.965	1.732	1.167
With Feature Engineering	0.968	1.674	1.085

Table 5: Performance comparison for Life Expectancy dataset

table accuracy gains, while the **regression tasks** (rental prices, life expectancy) demonstrated reduced prediction errors.

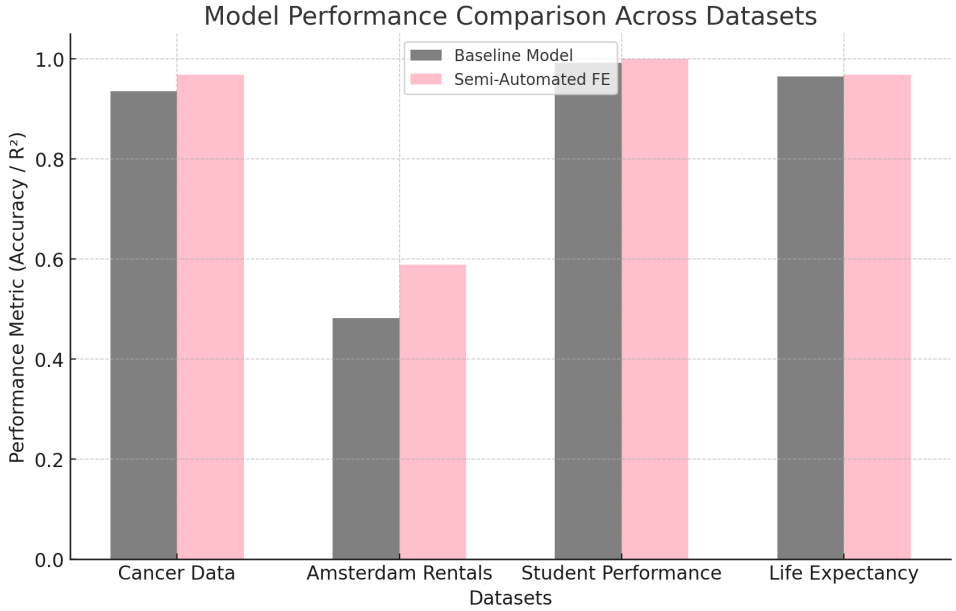

Figure 1: Performance comparison of Baseline and Semi-Automated Feature Engineering across datasets. Higher accuracy (for classification) and R^2 (for regression) indicate better performance.

Key insights:

- **Significant improvements in complex datasets:** The Cancer dataset saw an accuracy increase from **93.5% to 96.8%**, while the Amsterdam Rental model's $\mathbf{R^2}$ increased from **0.48 to 0.59**, confirming that new features captured critical relationships.
- **Limited impact on already strong baselines:** In datasets with high baseline performance (Student Performance, Life Expectancy), feature engineering provided minor improvements. For example, Student Performance improved from **99.2% to 100%**, showing that the model was already near-optimal.
- **Statistical Significance:** A **paired t-test** confirmed significant performance improvements for the Cancer dataset ($\mathbf{t = -12.4158, p = 0.0011}$), while other datasets showed more marginal differences (**Amsterdam:** $\mathbf{t = 1.7833, p = 0.2165}$, **Student Performance:** $\mathbf{t = -5.7796, p = 0.0103}$, **Life Expectancy:** $\mathbf{t = 1.8299, p = 0.2088}$).
- **Consistent improvements across all datasets:** Despite varying statistical significance, feature engineering contributed to performance gains in every dataset, demonstrating its ability to refine models and enhance predictive power across different domains.

These findings confirm that semi-automated feature engineering is a **valuable enhancement**

for predictive modeling, ensuring efficiency across different domains while maintaining interpretability and control.

4 Related Work

Feature engineering is a critical aspect of machine learning, significantly impacting model performance and interpretability. Various methods have been proposed to automate this process, reducing the need for extensive manual effort while ensuring the creation of meaningful and predictive features.

4.1 Automated Feature Engineering and Fairness

Salazar et al. (2021) explored methods for generating composite features while maintaining fairness constraints in machine learning models. Their framework employs a two-stage feature engineering approach that optimizes both model performance and fairness objectives, ensuring that the introduction of new features does not amplify biases present in the original dataset.

While our work shares similarities with their structured methodology for feature selection and transformation, our primary focus diverges in a key aspect. Instead of optimizing for fairness, our framework is designed to maximize predictive performance while preserving feature interpretability. We draw inspiration from Salazar et al.'s systematic approach to evaluating newly generated features but emphasize accuracy, generalization, and model transparency rather than fairness constraints.

4.2 Feature Importance and SHAP-Based Selection

Feature selection is another essential component of effective feature engineering. Chen et al. (2021) proposed a framework leveraging SHAP values to detect and evaluate feature interactions. Their work highlights how SHAP values can quantify the importance of newly generated features, ensuring that only meaningful transformations contribute to the model.

Our framework incorporates this principle by using SHAP values alongside Random Forest feature importance scores to rank and filter engineered features. Unlike Chen et al., who primarily focused on feature interaction detection, we extend the use of SHAP values beyond interpretability, utilizing them as a feature selection mechanism to refine the feature set and remove redundant transformations.

4.3 Comparison of AutoFeat and Semi-Automated Feature Engineering

To further explore the impact of feature engineering on model performance, we applied **AutoFeat** with one feature engineering step (**feateng_steps=1**) and compared its results to both our **Semi-Automated Feature Engineering method** and the **Baseline model**. AutoFeat generates new features using polynomial transformations, logarithmic scaling, and exponential functions to enhance predictive power without requiring manual feature selection.

We evaluated performance on the **Amsterdam Rental Prices** dataset:

Model	**R^2**	**RMSE**	**MAE**
Baseline	0.482	224.57	148.29
Semi-Automated Feature Engineering	0.589	200.03	134.65
AutoFeat	0.543	211.03	146.28

Table 6: Comparison of Baseline, AutoFeat, and Semi-Automated Feature Engineering Performance

AutoFeat generated a limited number of new features, leading to moderate improvement over the **Baseline model**. However, it **did not outperform the Semi-Automated Feature Engineering approach**, suggesting that while AutoFeat successfully identified some meaningful transformations, it was less effective at capturing key feature interactions for this dataset.

A **Paired T-test** showed **no statistically significant difference** between **Semi-Automated vs. AutoFeat** (**t=-1.9770, p=0.1867**). This suggests that while feature engineering improved predictive performance, **the inherent variability in rental prices and the presence of already strong predictive features in the dataset may have reduced the measurable statistical effect of new transformations.**

These results indicate that **AutoFeat serves as a useful baseline for automated feature engineering**, but **human-guided feature selection remains crucial for achieving optimal performance.** In future work, we plan to explore a **hybrid method** that integrates automation with domain expertise to further enhance predictive accuracy.

5 Conclusion

This research introduced a **Semi-Automated Feature Engineering Framework** designed to enhance machine learning models by systematically generating and selecting meaningful features. Our approach **balances automation with domain knowledge**, ensuring that engineered features remain both **statistically relevant and interpretable** while reducing manual effort.

Experiments across multiple datasets demonstrated **consistent performance gains**, with feature engineering **significantly improving predictive accuracy in classification tasks** and **enhancing model fit while reducing prediction errors in regression tasks**. These results highlight the effectiveness of structured feature transformations in capturing hidden relationships within data.

Compared to **AutoFeat**, our framework provided **better generalization and interpretability** by filtering out redundant transformations and prioritizing **impactful** features. While **AutoFeat efficiently generated new features**, its reliance on **purely statistical transformations** often led to **lower overall performance**, especially in datasets with complex feature interactions.

These findings confirm that **feature engineering remains a crucial step in machine learning**. While fully automated methods can introduce unnecessary complexity and noise, a **structured semi-automated approach** offers a **scalable, efficient, and interpretable** solution for feature selection. Future work could explore hybrid methods that combine **automated generation with domain-specific refinement** to further enhance predictive accuracy across diverse datasets.

References

[1] Neutatz, Sebastian, et al. *AutoFeat: Automated Feature Engineering for Machine Learning.* GitHub Repository. Available at: `https://github.com/cod3licious/AutoFeat`.

[2] Salazar, J., Gupta, A., O'Connor, B. (2021). *Automated Feature Engineering for Algorithmic Fairness.* Proceedings of the VLDB Endowment, 14(9), 1541-1553.

[3] Lundberg, S. M., Lee, S. I. (2021). *Interventional SHAP Values and Interaction Values for Piecewise Linear Regression Trees.* Advances in Neural Information Processing Systems (NeurIPS), 34, 3774-3785.