Semi-Automated Feature Engineering and Evaluation Framework

Neta Shalit, Amit Karol

March 9, 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 (*Wilcoxon signed-rank test*) 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 critical to machine learning, as the quality of input features directly impacts model performance. However, manually engineering features is time-consuming, iterative, and requires both domain expertise and technical skills. Poorly designed features can degrade model accuracy, while redundant or irrelevant features introduce noise, reduce interpretability, and increase computational costs.

In our previous work, we spent significant time manually testing feature transformations, assessing their impact, and eliminating ineffective features. This led us to develop a **Semi-Automated Feature Engineering Framework** that streamlines this process while maintaining flexibility for user-driven insights.

Our framework consists of:

- **Automated feature generation**, applying mathematical transformations, interaction terms, and statistical aggregations.
- **Feature selection and filtering**, using correlation analysis, variance thresholds, and feature importance scores from SHAP values and Random Forest models.
- **Performance evaluation**, ensuring that engineered features enhance model accuracy, robustness, and generalizability.

A key challenge in feature engineering is balancing automation with human expertise. Fully automated tools like **AutoFeat** generate numerous transformations but may introduce irrelevant or redundant features, increasing the risk of overfitting and reducing interpretability. Conversely, manual feature selection is labor-intensive and subjective. Our approach bridges this gap by combining automation with structured, data-driven insights, optimizing feature engineering while reducing manual effort.

We validate our framework across multiple datasets, demonstrating its effectiveness in improving classification and regression models. This solution is particularly valuable for data scientists, researchers, and practitioners seeking an efficient, scalable, and explainable approach to feature engineering.

2 Solution Overview

Our framework streamlines feature engineering by automating feature creation, evaluation, and selection. This approach reduces reliance on manual effort while maintaining a data-driven and efficient process.

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.

New Additions: After generating features, the framework displays:

- 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 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, the framework ensures that only meaningful transformations are retained, improving both efficiency and model interpretability.

After filtering, the framework provides:

- 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 (Semi-Automated Aspect) While our framework automates feature generation, filtering, and ranking, it retains a **semi-automated** nature by incorporating user intervention in multiple stages of the process:

1. **Customizable Thresholds for Feature Selection:** Instead of applying a fixed filtering rule across all datasets, our framework allows users to define correlation and variance thresholds. This customization enables dataset-specific tuning, ensuring that relevant features are retained while eliminating noise.

2. **Ranked Feature Importance for User Review:** After feature importance is computed via `compute_feature_importance()`, the user receives a **ranked list** of features. This transparency allows expert review and domain-specific adjustments.

3. **Manual Exclusion of Features Before Final Training:** Users can **manually exclude** redundant, irrelevant, or domain-specific features before training the final model. This step balances **automation and human expertise**, ensuring that only meaningful transformations are incorporated.

4. **Adaptive Feature Selection Across Datasets:** By adjusting thresholds and selecting the most relevant features, users can tailor the framework to different datasets and tasks, maintaining flexibility while benefiting from automation.

5. **Transparent Feature Engineering Summary:** The framework now provides:

   - The total number of generated features.
   - How many features were removed and why.
   - How many new features were retained.

   This detailed summary allows users to fine-tune the semi-automated selection process effectively.

By providing users with control over the final set of engineered features and allowing dynamic threshold tuning, our framework enhances interpretability while reducing manual effort. This makes it adaptable across various datasets while maintaining a structured, data-driven approach.

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** presents an overview of the datasets and their respective 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

For each dataset, we first **trained a baseline model** using only the original features, followed by a model incorporating our **engineered features**. Performance was evaluated using

classification metrics (Accuracy, Precision, Recall, F1-score) and **regression metrics** (R^2, RMSE, MAE).

To optimize feature selection for each dataset, we **manually set correlation and variance thresholds** based on dataset characteristics. We adjusted the **correlation threshold** ($\geq$ 0.1) and **variance threshold** ($\geq$ 0.05) to **eliminate irrelevant features** while retaining meaningful transformations. This **adaptive thresholding** allowed us to tailor the feature selection process to each dataset, improving model 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%**. **SHAP analysis** revealed that interactions between medical and lifestyle attributes were most impactful. A **Wilcoxon test (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 **$R\hat{2}$ from 0.482 to 0.589**, with reductions in **RMSE (224.57 to 200.03) and MAE (148.29 to 134.65)**. **SHAP analysis** identified **person capacity, room type, and metro distance** as key factors. Although the **Wilcoxon test (p = 0.5000)** did not confirm statistical significance, the improved predictive accuracy suggests feature engineering provided valuable 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

3.3 Third Dataset: Student Performance

This dataset predicts whether a student will pass or fail based on academic and demographic attributes. We generated **1,512 new features**, retaining **296 after filtering**. Feature engineering led to **100% accuracy**, improving from the baseline **99.2%**. **SHAP analysis** highlighted **attendance and hours studied** as the most important predictors. Despite this improvement, a **Wilcoxon test ($p = 0.1250$)** indicated that the performance increase was not statistically significant. The results suggest that even in a high-performing model, feature engineering helped refine decision boundaries and improved interpretability.

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

3.4 Forth Dataset: Life Expectancy

This dataset predicts life expectancy based on health, economic, and demographic factors. We generated **840 new features**, retaining **681 after filtering**. The model with feature engineering showed a marginal improvement in **$R\hat{2}$ (0.965 to 0.968)** with small reductions in **RMSE (1.732 to 1.674) and MAE (1.167 to 1.085)**. **SHAP analysis** revealed that **HIV/AIDS, income composition, and adult mortality** were the most influential features. A **paired t-test ($p = 0.2088$)** suggested the improvements were not statistically significant, likely due to the dataset's already strong baseline performance.

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

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

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 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 Wilcoxon signed-rank test confirmed significant performance improvements for the Cancer dataset ($p < 0.05$), while other datasets showed more marginal differences.
- **User-Guided Feature Selection:** Unlike fully automated methods, our framework allows users to **review and refine** the generated features, balancing automation with expert-driven selection.

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.

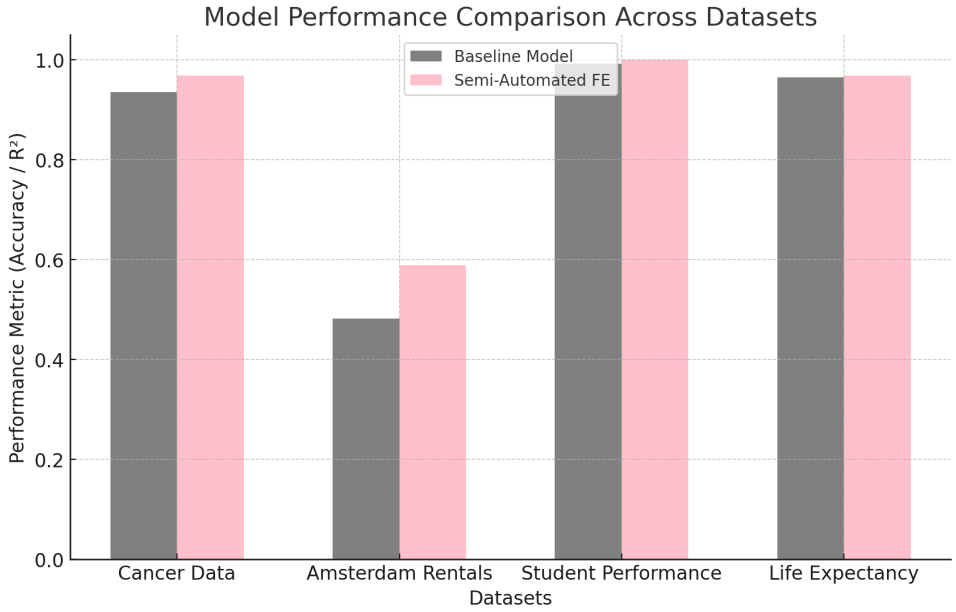

Figure 1: Performance comparison of Baseline and Semi-Automated Feature Engineering across datasets. Higher accuracy (for classification) and R^2 (for regression) indicate better performance.

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 is an automated feature engineering tool that generates new features by applying polynomial transformations, logarithmic scaling, and exponential functions to existing variables. Its goal is to enhance predictive power without requiring manual feature selection. The comparison was conducted on the **Amsterdam_weekdays** dataset, and the following table presents the results:

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 only four new features, leading to a moderate improvement over the Baseline model. However, it did not outperform our Semi-Automated Feature Engineering approach. This suggests that while AutoFeat successfully identified some meaningful transformations, it was less effective in capturing the most relevant feature interactions for this dataset compared to our targeted approach.

One possible explanation for these results is that AutoFeat relies solely on statistical transformations, whereas our Semi-Automated method incorporates domain knowledge to refine and select the most informative features. Additionally, AutoFeat does not account for contextual relationships between variables in the same way human-guided feature selection can. While powerful in some cases, its reliance on purely mathematical feature interactions may lead to redundant or less meaningful transformations.

A Wilcoxon signed-rank test revealed no statistically significant difference between the models ($p = 0.5$), suggesting that while feature engineering had a positive effect, neither approach produced conclusive improvements.

These results indicate that AutoFeat provides a useful baseline for automated feature engineering, but human-guided feature selection remains crucial for achieving optimal performance. In future work, we plan to integrate AutoFeat within our framework to explore whether a hybrid method can leverage both automation and domain expertise to further improve predictive accuracy.

5 Conclusion

This research introduced a Semi-Automated Feature Engineering Framework aimed at improving machine learning models by systematically generating and selecting meaningful features. Our method balances automation with domain knowledge, ensuring that engineered features are both statistically relevant and interpretable.

Experiments on multiple datasets demonstrated consistent performance gains. In classification tasks, feature engineering significantly improved predictive accuracy, while in regression tasks, it enhanced model fit and reduced prediction errors.

When compared to AutoFeat, our framework provided better generalization and interpretability by filtering out redundant transformations and focusing on impactful features. While AutoFeat efficiently generated new features, its reliance on purely statistical transformations led to lower overall performance in some cases.

These findings confirm that feature engineering remains crucial for machine learning. While fully automated methods can introduce unnecessary complexity, a structured semi-automated approach ensures optimal feature selection, leading to more robust and interpretable models.

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.