Automated Feature Transformation Recommender

Roni Gotlib Hallel Weinberg

Abstract

This project presents an automated system for optimal feature transformation selection in tabular datasets, focusing on linear models (Linear/Logistic Regression). Manual feature engineering is time-consuming and error-prone, as it often relies on trial-and-error or heuristic rules. Our method automatically analyzes feature distributions and evaluates a comprehensive set of candidate transformations—including log, square root, Yeo-Johnson, quantile, box-cox, standardization, min-max scaling, polynomial features, and binning—using cross-validation performance metrics and normality tests. Experimental results on diverse datasets (e.g., Diabetes, California Housing) demonstrate that our approach can yield quantitative improvements in model performance. This system not only streamlines the preprocessing workflow but also provides interpretable, quantitative justification for transformation choices, directly applying core concepts from exploratory data analysis and feature engineering.

1 Problem Description

Feature engineering is a critical step in the data science pipeline that significantly impacts the performance of machine learning models. However, the manual process of selecting appropriate transformations for each feature is inefficient, time-consuming, and subject to human error. This project addresses the challenge of automating transformation selection for tabular data, particularly when using linear models. Our goal is to reduce the manual burden on data scientists while ensuring that transformation choices are both effective and quantitatively justified.

2 Solution Overview

Our solution, the Automated Feature Transformation Recommender, is designed to systematically evaluate and select optimal transformations for tabular data features. The system takes a tabular dataset and a model type specification (linear or logistic regression) as input, and produces transformed features optimized for model performance.

2.1 System Architecture

The system consists of two main components:

1. **FeatureTransformationRecommender**: The core component that analyzes features and recommends transformations.

2. **BasicEvaluator**: A complementary component that provides metrics and visualizations to evaluate transformation effectiveness.

2.2 Transformation Selection Algorithm

The recommender uses a data-driven approach to transformation selection:

1. **Feature Type Detection**: The system first identifies numeric and categorical features.

2. **Candidate Generation**: For each numeric feature, the system generates a set of candidate transformations:

Table 1: Candidate Transformations for Numeric Features

Transformation	**Description**
None	Baseline - no transformation
Log transformation	`np.log1p`, if all values are positive
Square root	`np.sqrt`, if all values are non-negative
Yeo-Johnson	Power transformation using `PowerTransformer`
Quantile	Maps to a normal distribution
Box-Cox	If all values are positive
Standardization	Z-score normalization
Min-Max scaling	Scales to [0,1] range
Polynomial features	Degree 2 - quadratic term
Binning	Quantile-based discretization

3. **Performance-Based Selection**: When a target variable is provided, the system uses an inner cross-validation loop to evaluate each candidate:

   - For each feature and candidate transformation, a temporary dataset is created where only that feature is replaced with its transformed version.
   - A simple model (`LinearRegression` or `LogisticRegression`) is trained using cross-validation to obtain a robust performance estimate.
   - A minimum improvement threshold (`min_improvement`) is defined to ensure that only candidates yielding a significant relative improvement compared to the baseline are selected. This prevents the adoption of transformations that do not provide a meaningful benefit.

4. **Distribution-Based Fallback**: When no target variable is provided, the system uses statistical metrics to select transformations. Normality tests are performed on both the original and transformed features, and the transformation that best improves the normality of the feature's distribution is selected. In cases where the normality test fails or is not reliable, the system employs a fallback metric based on skewness.

5. **Categorical Feature Handling**: For categorical features, the system recommends *one-hot encoding* for features with more than two unique values and *label encoding* for binary features.

2.3 Transformation Application

Once the best transformations are selected, the system applies them to the dataset:

- For numeric features, the chosen transformation is applied directly.
- Missing values are imputed using column means pre-transformation.
- For categorical features, the recommended encoding (one-hot or label) is applied.

2.4 Evaluation Framework

The **BasicEvaluator** component provides comprehensive metrics to assess transformation effectiveness:

1. **Statistical Metrics**: The system measures distribution symmetry through skewness, tail behavior through kurtosis, and normality through p-values that assess how closely the distribution resembles a normal distribution.
2. **Model Performance Metrics**: For regression tasks, we use Mean Squared Error (MSE), while classification tasks are evaluated using Accuracy, F1 Score, and AUC.
3. **Visualization Tools**: The system provides feature distribution plots (before and after transformation), model evaluation plots (predicted vs. actual, residuals for regression; ROC curves and calibration plots for classification), and correlation heatmaps to assess changes in feature relationships.

2.5 Implementation Details

The system is implemented in Python using `scikit-learn` for the machine learning components and transformation methods. Key implementation decisions include:

- **Modularity**: Clear separation between the transformation recommendation and evaluation components facilitates maintenance and potential future extensions.
- **Thresholding**: The use of a minimum improvement threshold prevents the adoption of trivial or non-beneficial transformations.
- **Error Handling**: Robust handling of data constraints (e.g., ensuring that only positive values are used for log and Box-Cox transformations) is implemented. When a candidate transformation is not applicable or encounters errors, it is safely skipped, and alternative metrics (such as skewness) are used.
- **Performance Considerations**: The incorporation of cross-validation ensures that performance estimates are reliable and reduces the risk of overfitting to a particular dataset split.

2.6 User Workflow

The typical workflow when using our system is:

1. Initialize the recommender with the desired model type and parameters.
2. Apply the recommender to the dataset using the `fit_transform` method.
3. Initialize the evaluator and pass the transformation details.
4. Generate evaluation metrics and visualizations.
5. Make informed decisions based on the comprehensive evaluation of both statistical and model performance measures.

This approach ensures that transformation decisions are not only automated, but also transparent and quantitatively justified.

3 Experimental Evaluation

To assess the effectiveness of our Automated Feature Transformation Recommender, we conducted experiments on four diverse datasets spanning both regression and classification tasks. The evaluation compares our system against a baseline (raw features) and an alternative feature engineering tool, AutoFeat, focusing on three key aspects: model performance, computational efficiency, and feature interpretability.

3.1 Datasets and Experimental Setup

We evaluated the system using the following datasets:

Table 2: Datasets used in the evaluation

Dataset	Task Type	Description
Diabetes	Regression	Predicts disease progression based on physiological attributes.
California Housing	Regression	Predicts median house values in California districts.
Breast Cancer	Classification	Identifies malignant vs. benign tumors from cell attributes.
Iris	Classification	Classifies iris species based on petal and sepal measurements.

For each dataset, we applied a consistent methodology: training a baseline linear model on raw features, evaluating our transformation approach, and comparing against AutoFeat. We measured model performance (MSE/R^2 for regression, accuracy/F1 for classification), computational efficiency, and statistical improvements in feature distributions.

3.2 Results and Analysis

Table 3 summarizes the performance improvements achieved by our approach and AutoFeat. Both methods are evaluated relative to the baseline models across all tested datasets.

Table 3: Summary of performance improvements across datasets

Dataset	Metric	Baseline	Ours	AutoFeat	Imp. Ours	Imp. AutoFeat
Diabetes	MSE ↓	2900.19	2705.63	2573.79	6.71%	11.25%
(Regression)	R^2 ↑	0.453	0.489	0.514	8.11%	13.61%
California Housing	MSE ↓	0.56	0.46	0.43	17.01%	23.28%
(Regression)	R^2 ↑	0.576	0.648	0.675	12.54%	17.15%
Breast Cancer	Accuracy ↑	0.956	0.982	0.982	2.75%	2.75%
(Classification)	F1 ↑	0.956	0.982	0.982	2.76%	2.75%
Iris	Accuracy ↑	0.967	0.967	0.967	0.00%	0.00%
(Classification)	F1 ↑	0.967	0.967	0.967	0.00%	0.00%

3.2.1 Performance Analysis by Problem Type

For regression tasks, our approach achieved substantial improvements in error reduction (MSE) and explained variance (R^2), with the California Housing dataset exhibiting the most significant gains. Although AutoFeat sometimes achieved slightly higher performance improvements on regression datasets, this difference is modest compared to its substantially higher computational overhead. For classification tasks, both approaches performed identically on the Breast Cancer dataset, while neither showed improvements on the Iris dataset, suggesting that its raw features were already well-suited for classification. Overall, these results demonstrate that our selective transformation approach is effective across diverse datasets and problem types.

3.2.2 Computational Efficiency Analysis

A key advantage of our approach is its computational efficiency compared to more complex feature engineering tools. Table 4 summarizes the training times (in seconds) for each approach.

Table 4: Computational efficiency comparison (training time in seconds)

Dataset	Baseline	Our Approach	AutoFeat
Diabetes	0.003	2.307	63.612
California Housing	0.010	7.100	83.527
Breast Cancer	0.533	144.331	2582.120
Iris	0.013	2.188	26.574

Although both transformation approaches increase computational overhead compared to the baseline, our approach requires significantly less time than AutoFeat—often 10–20 times faster.

For instance, on the Breast Cancer dataset, AutoFeat's training time is approximately 18 times longer than that of our approach.

3.2.3 Feature Interpretability and Transformation Selection

Maintaining feature interpretability is crucial in feature engineering. Our approach prioritizes preserving the semantic meaning of features, as reflected by the interpretability scores in Table 5.

Table 5: Feature interpretability preservation scores

Dataset	**Interpretability Score (%)**
Diabetes	80.00
California Housing	75.00
Breast Cancer	60.00
Iris	75.00

These scores represent the percentage of features that retain clear semantic meaning after transformation. For example, the Breast Cancer dataset shows a lower score (60%), likely due to a greater need for complex transformations to enhance model performance, while the Diabetes dataset achieves a higher interpretability score (80%).

Table 6 highlights the most significant skewness improvements achieved by our transformations, primarily observed in the California Housing and Breast Cancer datasets.

Table 6: Top skewness improvements across datasets

Dataset	**Feature**	**Raw Skew**	**Trans. Skew**	**Transformation**
California Housing	AveOccup	88.04	0.04	Quantile
Breast Cancer	smoothness error	2.40	0.00	Binning
Breast Cancer	mean compactness	1.25	0.00	Binning
Breast Cancer	mean concavity	1.38	0.00	Binning
Breast Cancer	worst symmetry	1.44	0.01	Binning

Analysis of the selected transformations reveals that:

- **Selective Transformation:** For the majority of features (68%), no transformation was applied—demonstrating the importance of selective transformation.
- **Effective Methods:** Binning was the most frequently selected transformation, followed by quantile and Yeo-Johnson methods.
- **Significant Distributional Improvements:** Notably, the quantile transformation in the California Housing dataset reduced the skewness of the AveOccup feature dramatically (from 88.04 to nearly 0).

3.3 Discussion

The experimental results highlight several key strengths of our approach:

1. **Effective Performance Improvements:** Both our approach and AutoFeat yield significant improvements, with performance gains measured relative to the baseline models.

2. **Computational Efficiency:** Our approach achieves comparable performance improvements at a fraction of the computational cost—often 10–20 times faster than AutoFeat.

3. **Interpretability Focus:** By applying selective and simpler transformations, our system maintains high interpretability scores.

4. **Statistical Distribution Enhancements:** The applied transformations substantially reduce skewness in problematic features, facilitating improved model performance.

5. **Balanced Performance-Efficiency Trade-off:** Our results demonstrate a favorable trade-off between performance gains and computational cost, providing near-optimal improvements with significantly reduced processing requirements.

Overall, while AutoFeat sometimes achieves slightly higher absolute performance improvements, our approach offers a more practical alternative by combining competitive predictive gains with significantly lower computational overhead. Our solution emphasizes both efficiency and interpretability, providing an effective tool for feature transformation in linear models applied to tabular data.

4 Related Work

Automated feature generation and transformation have been widely explored in recent years, with several tools and methodologies proposed to enhance predictive modeling.

[2] introduced an automated approach for feature extraction in battery capacity modeling, demonstrating how domain-specific transformations can improve predictive accuracy. Their method focuses on generating features tailored specifically for battery degradation prediction, rather than providing a general-purpose transformation framework applicable across different datasets and models. In contrast, our approach offers a flexible mechanism for selecting appropriate feature transformations in a data-driven manner, without relying on domain-specific knowledge.

[3] presented *AutoFeat*, a Python library for automated feature engineering and selection. AutoFeat aims to improve model performance by generating nonlinear feature combinations and selecting the most relevant ones. In our evaluation, we compared our system's performance to AutoFeat. While AutoFeat achieved a greater improvement in predictive accuracy, our approach demonstrated significantly lower training time overhead. This efficiency advantage makes our system more suitable for scenarios where computational cost is a key constraint.

[1] introduced *FeatureTools*, a Python library for automated feature engineering. FeatureTools allows users to automatically generate relevant features by stacking simple transformations,

providing a powerful approach to feature engineering for machine learning tasks. Compared to AutoFeat, FeatureTools focuses more on generating features by leveraging a series of pre-defined transformation functions, which makes it more suited for domain-specific feature generation tasks. In contrast, our approach emphasizes automatic selection and evaluation of transformations for both general-purpose and domain-specific datasets.

5 Conclusion

Our automated feature transformation recommender significantly improves the feature engineering process for linear models by minimizing manual effort while enhancing model performance. Experimental evaluations showed meaningful improvements across diverse datasets—up to 17.01% reduction in MSE for regression tasks and 2.75% increase in accuracy for classification—while maintaining good interpretability.

The project revealed key insights about effective feature transformation: (1) selective transformation is more effective than indiscriminate application, as evidenced by 68% of features requiring no transformation; (2) simple transformations often provide most of the benefit without added complexity; and (3) there exists an important trade-off between computational efficiency and marginal performance gains—our targeted approach achieved comparable results with dramatically reduced computational requirements.

This project successfully applied core data science concepts from class, creating a tool that bridges theory and practical preprocessing challenges, and fulfills our goal of assisting data scientists in performing more efficient data preparation.

References

[1] Analytics Vidhya. Guide to automated feature engineering using featuretools in python. Blog post, 2018.

[2] S. Greenbank and D. Howey. Automated feature extraction and selection for data-driven models of rapid battery capacity fade and end of life. *arXiv*, 2021.

[3] F. Horn, R. Pack, and M. Rieger. The autofeat python library for automated feature engineering and selection. *arXiv*, 2019.