Recursive Feature Engineering for Automated Machine Learning Optimization

Max Kuzmin

March 2025

Every machine learning pipeline consists of several key steps, Each one of them is crucial for better model performance. While many of these steps already have automated solutions, one of the most important steps, feature engineering, is still largely manual and heuristic driven, and the existing automated solutions are not yet good enough to replace human expertise. In this project, I propose an automated recursive feature engineering framework that generates, evaluates, and selects optimal feature transformations dynamically. My project suggests a recursive algorithm to create polynomial, logarithmic, exponential and interaction-based features, while continuously evaluating their contribution to model performance. The proposed approach was tested on multiple datasets covering different domains, including housing prices, Energy Consumption, and weather history. The experimental results showed a consistent decrease in Root Mean Squared Error (RMSE) across all datasets. For instance, on the House Prices dataset, my Recursive Feature Engineering demonstrated an improvement of approximately 7%.

1 Introduction and Problem Description

Feature engineering is one of the crucial steps in machine learning pipelines. It is the process of transforming raw data into meaningful features that increase the predictive power of machine learning models and significantly impact their performance. Feature engineering involves creating new features, modifying existing ones, and selecting the most relevant features to improve model accuracy. This process is very important because the right set of features can not only significantly improve model performance, but also help the model generalize better to unseen examples, compensate for limited data, and uncover patterns that would otherwise remain hidden in

raw data. However, feature engineering is a challenging and time-consuming task that requires domain expertise and manual intervention. It involves exploring different transformations, interactions, and combinations of features to identify the most informative ones, a process that is often done in trial and error manner. Unfortunetly, automating feature engineering is a difficult task. Due to the large search space of possible feature transformations and the strong dependency on the specific characteristics of the dataset, constructing a generalized system that will perform a good feature engineering on every given set, is a hard work.

The key challenges include:

- large number of potential feature transformations
- non-linear and dataset-dependent relationships between features
- Balancing computational efficiency with the need to generate meaningful feature transformations
- Identifying the most useful transformations for a given dataset.
- Avoiding the curse of dimensionality
- Ensuring that newly generated features provide additional predictive value

2 Solution Overview

My proposed solution, **Recursive Feature Engineering (RFE)**, automates the feature engineering process by iteratively applying transformations and evaluating their effectiveness using the regression model.
The Recursive Feature Engineering framework gets a dataset as input, and the target variable to predict. Initially, the framework preprocesses the data by handling missing and unwanted values (NaNs, inf, etc.) that can harm the process and lead to an error. Categorical features are ignored in this process, as they require separate treatment. This is a base preprocessing that is done to ensure stable working of the RFE, as further more coplex and domain specific preprocessing is not the goal of my project, and its the responsibility of the user.
Once the preprocessing is complete, the framwork iteratively applies various mathematical transformations on the existing features to generate new,

more complex features. These transformations include polynomial expansions, logarithmic transformations and interaction terms, all to capture the relationships between the features. Each at a time, the newly generated feature is evaluated based on its impact on model performance, using a predefined metric. The features that contribute positively to the model performance are kept, while those that introduce noise or redundancy are discarded. The selected features are then used to generate further feature transformations in a recursive manner. The process continues until a stopping criterion is met, such as no further improvement in model performance or predefined number of iterations. The final output of RFE is an optimized dataset containing only the most informative features, enhancing model accuracy.

The core steps of my approach include:

1. **Creation:** a configuration of the tool created staticly, given -
    - A model to optimize.
    - Maximum number of features to generate.
    - Threshold for the feature importance.
2. **Acticvation:** The tool is activated with a given dataset and target variable.
3. **Random selection:** The program randomly chooses the transformation to apply to the features.
4. **Order:** The order of the features is determined by the mutual correlation* for transformations of two features, and feature importance for transformations of one feature.
5. **Feature Transformation:** The chosen transformation is applied on the features at the top of the order, and the new feature is added to the dataset
6. **Feature Evaluation:** The model is trained on the dataset with the new feature, and RMSE is calculated. If the RMSE is lower than the previous one, the feature is kept, otherwise, it is discarded and the process continues to the next feature in the order.
7. **Feature Selection:** The new set of features is evaluated based on their importance, and the features that are below the threshold are discarded.

8. **Recursive Iteration:** The process repeats, generating new features based on the previously selected dataset, including the newly generated feature.

9. **Stopping Criterion:** The process terminates when additional transformations no longer yield performance improvements or if the maximum number of features is reached.

* I chose mutual correlation as a measure of the order of the features, as during my research I found a connection between high mutual correlation and the improvement in the model performance after the transformation.

My approach ensures that the most impactful features are identified and incorporated while eliminating low value features. By dynamically pruning and expanding the feature set, RFE optimizes model performance without manual intervention.

3 Experimental Evaluation

I tested the Recursive Feature Engineering framework on multiple datasets to evaluate its effectiveness in improving model performance. The experiments were conducted using a RandomForestRegressor model, and the evaluation metric used was Root Mean Squared Error (RMSE).
Each experiment followed the same procedure:

1. Training a baseline model with only basic feature engineering, such as filling missing values and infinities

2. Applying our recursive feature engineering algorithm.

3. Comparing model performance before and after feature augmentation.

3.1 Datasets Used

The five standouts datasets used in the experiments are:

- **Dataset 1:** Paris House Prices dataset.
- **Dataset 2:** Flood Probability dataset.
- **Dataset 3:** Weather History dataset.
- **Dataset 4:** House Price dataset.
- **Dataset 5:** Energy Consumption dataset.

3.2 Results and Analysis

Dataset	Baseline RMSE	RFE RMSE	RFE Improvement (%)
Paris House Prices	4002.878	3917.226	2.139%
Flood Probability	0.02589	0.02503	3.295%
Weather History	0.0549	0.365	2.099%
House Price	19827.827	18510.230	6.645%
Energy Consumption	514.046	481.231	6.383%

Table 1: Comparison of RMSE scores before applying Recursive Feature Engineering, after applying Recursive Feature Engineering and RFE Improvement in percentage

It can be seen that across all tested datasets, Recursive Feature Engineering consistently reduced the RMSE compared to the baseline model. the percentage of the improvement ranges from 2 to 7 percent, making it a solid tool for improving the accuracy of models. However, the extent of improvement varied depending heavily on the dataset characteristics.
In the House Prices dataset, RFE significantly lowered RMSE by 7%, showing a good ability of compensating for limited data, as the House Prices dataset have a relatively small number of samples and features compared to the complexity of the target variable. On the other hand, on the weather History dataset the improvement was less significant, leading to only 2% improvement, as the features in the data sets already highly informative and less redundant.
Overall, the improvement achieved by RFE is not extremely high, but it still demonstrates its potential as a useful tool for enhancing model performance, especially when dealing with datasets that may not already have well-engineered features. In theory, RFE's ability to explore and optimize feature interactions should lead to significant improvements in model accuracy, mainly thanks to -

1. **consistent evaluation:** RFE iteratively explores feature combinations, evaluates their importance, and selects the most helpful features. This process helps the model identify the most impactful feature interactions and avoid redundant or uninformative ones, improving model accuracy over time.

2. **Dynamic Feature Optimization:** The recursive feature generation allows RFE to adapt dynamically to the dataset's characteristics. As

it examines feature interactions in a recursive manner, the program captures a large variability of transformations, capturing complex relationships within the data that might not be initially apparent.

3. **Reduction of Redundancy:** By evaluating the contribution of each feature and its interactions, RFE effectively reduces redundancy. This means that features that do not add unique information are discarded, resulting in a more concise and effective set of features that improve model efficiency and accuracy.

4. **Compatibility with Different Models:** RFE can be applied to any machine learning model, such as decision trees, random forests, or linear regressions, making it a versatile and adaptable tool. This versatility ensures that RFE can be effectively used in various domains, enhancing model performance across different types of predictive tasks.

After showing consistent improvement in RMSE and analyzing its effectiveness across various datasets, it is clear that RFE works and outperforms the baseline model.

4 Related Work

Several existing tools and techniques attempt to automate the process of feature engineering -

- **Featuretools** - An open-source library that automates feature generation using a technique called Deep Feature Synthesis

- **DIFER** - A feature engineering method that integrates gradient-based optimization to discover informative features.

- **AutoFeat** - A machine learning-based feature engineering framework that generates polynomial and interaction-based features.

The existing works provided valuable inspiration for my Recursive Feature Engineering project, as they focus on automating feature engineering in multiple ways and approaches. Some of them even had easy-to-understand articles about the main ideas and concepts, from which I adopted a similar theme and workflow. However, my approach to the problem distinguishes itself by including a recursive generation of complex features, based on the already generated ones, and a constant evaluation of the new features to

ensure improvement. Unlike other automated feature engineering methods that primarily rely on predefined transformations or evolutionary algorithms, my project dynamically selects and applies transformations based on randomness, statistical significance, feature interactions, and their contribution to model performance. The recursive approach of my project allows the set of avalible transformations to be endless, with no need for manual creation, repeatability and high place complexity.

5 Future improvements

Future improvement could include:

- Extending the framework to handle categorical features - encoading them to numerical values, and applying transformations that suitble for them.
- Integrating additional transformation types - such as trigonometric functions, three way interactions, and more non-linear transformations.
- Other metrics - for now the only metric used is RMSE, but other metrics could be used to ensure the model is optimized for the specific task.
- Optimizing the recursive selection process for efficiency - the current implementation gives the same probability to each one of the transformations, leading to it being not optimized for large datasets, and could be improved be Heuristics to reduce the computational cost.

6 Conclusion

In this project, I proposed a Recursive Feature Engineering (RFE) framework designed to automate the process of generating and selecting features for machine learning models. Through an iterative process of generating polynomial, logarithmic, exponential, and interaction-based features, and evaluating their impact on model performance, the framework effectively enhances predictive accuracy while reducing the need for manual intervention. As seen in the experimental results, RFE consistently reduced the Root Mean Squared Error (RMSE) and improved the model's predictive power. While the improvements were modest in some cases, they confirmed the validity of the approach. Unfortunately, the comparison of RFE to existing

automated feature engineering techniques revealed that while RFE provides improvements in certain scenarios, its effectiveness is heavily dependent on the dataset characteristics, and other methods may outperform RFE, both in model performance and runtime complexity.

Key lessons I learned from this project include the importance of balancing the complexity of feature generation with the need for computational efficiency. While generating many new features can potentially enhance model performance, it also introduces the challenge of managing computational resources and avoiding overfitting. Additionally, it became clear that the effectiveness of feature engineering methods like RFE is highly dataset dependent. In some cases, the dataset already contained well-engineered features, which led to only marginal improvements, while in others, the transformations introduced by the recursive process significantly enhanced the model.

Overall, this project not only offers a practical tool for automated feature engineering but also provides a deeper understanding of how recursive feature generation and selection can improve machine learning models. The approach presented here can be adapted to various domains and serve as a stepping stone for more advanced automated machine learning frameworks.