Automated Feature Bucketing for Enhanced Quantile Regression Using KMeans

Mark Klaiman

Gabriele Guetta

Abstract

This research tackles the challenge of automating feature bucketing within data science pipelines to improve predictive model performance and interpretability. Traditional bucketing methods, often manual or based on simplistic rules, fail to optimize feature representations effectively, requiring significant effort and risking suboptimal outcomes. We propose a novel solution, implemented as the SimpleAutoBucketer class, which employs KMeans clustering to automatically determine optimal bucket configurations for numerical features. This approach was evaluated on four diverse datasets—ds_salaries, hospital data analysis, House_Rent_Dataset, and Sample-Superstore. The automated approach yielded a modest improvement in the ds_salaries dataset, increasing R^2 by 5.58% (from 0.3988 to 0.4210) and reducing RMSE by 1.87% (from 48718.22 to 47808.31). However, it substantially underperformed in the remaining datasets, with R^2 declining by 11.77% to 87.37% and RMSE rising by 59.93% to 111.73%. Specifically, in hospital data analysis, R^2 dropped from 0.9673 to 0.8534 and RMSE increased from 1444.02 to 3057.42; in House_Rent_Dataset, R^2 fell from 0.6407 to 0.0809 and RMSE rose from 37843.19 to 60521.54; and in Sample-Superstore, R^2 decreased from 0.7547 to 0.2473 with RMSE climbing from 109.06 to 191.04. These findings indicate that while automated bucketing can enhance performance in specific scenarios, its effectiveness is highly contingent on dataset characteristics, limiting its general applicability across varied domains.

Problem Description

Feature engineering remains a cornerstone of effective predictive modeling in data science, transforming raw data into inputs that machine learning algorithms can leverage effectively. Among the various techniques, bucketing (or discretization) of numerical features stands out as a critical process. Bucketing converts continuous variables into discrete categorical bins, enabling models to capture non-linear relationships, improve interpretability, and sometimes enhance predictive accuracy. For instance, discretizing an "age" feature into ranges like "0-18," "19-35," and "36+" can reveal distinct patterns in a target variable, such as income or health outcomes, that a continuous representation might obscure.

Despite its importance, current bucketing practices face significant challenges that hinder their effectiveness and scalability:

- **Manual Effort**: Data scientists frequently define bucket boundaries manually, relying on domain expertise or trial-and-error. For example, setting income thresholds at $50,000 and $100,000 might seem intuitive but lacks empirical grounding, leading to inconsistent results across datasets. This subjectivity increases workload and introduces human bias.

- **Suboptimal Methods**: Widely used techniques like equal-width binning (dividing a range into fixed intervals) or equal-frequency binning (ensuring equal numbers of observations per bin) often ignore the data's underlying distribution or its relationship with the target variable. These methods may miss critical breakpoints, such as natural clusters in salary data that correlate with job roles.
- **Lack of Validation**: Traditional approaches rarely incorporate systematic validation to ensure bucket configurations maximize model performance. Without feedback loops, it's unclear whether chosen buckets are optimal or merely adequate.
- **Missed Patterns**: Simplistic or manual bucketing can overlook significant data patterns, such as non-linear trends or thresholds that strongly influence the target. For instance, hospital data might show a sharp increase in risk beyond a specific blood pressure value, which uniform bins might fail to isolate.
- **Scalability Issues**: As datasets grow in size and complexity—often featuring dozens or hundreds of numerical features—manual bucketing becomes impractical. Applying it to large-scale projects, such as analyzing nationwide sales data, demands automation to maintain efficiency.

These shortcomings highlight a pressing need for an automated, data-driven bucketing approach. Such a solution should minimize human intervention, adapt to diverse data distributions, validate its effectiveness against model outcomes, and scale seamlessly to high-dimensional datasets. By addressing these gaps, we can unlock the full potential of feature bucketing, making it a more robust tool in the data science toolkit and improving the quality of predictive models across applications.

Solution Overview

To address the limitations of traditional bucketing, we developed the SimpleAutoBucketer, a Python class that automates the discretization of numerical features using KMeans clustering. This solution combines unsupervised learning with performance-driven optimization to create statistically meaningful and predictive buckets. Below, we outline its key components and functionality:

Feature Processing

The SimpleAutoBucketer processes each numerical feature independently, ensuring flexibility across diverse datasets. For datasets with mixed feature types, non-numeric features (e.g., categorical variables like "job title") are first encoded into numerical representations using a dictionary-based mapping. This preprocessing ensures all features are compatible with the clustering algorithm, broadening the system's applicability.

KMeans Clustering for Bucketing

The core of our approach lies in using KMeans clustering to identify natural groupings within each numerical feature:

- **Clustering Application**: For a given feature, KMeans is applied to partition the data into clusters based on value similarity. Each cluster represents a potential bucket.
- **Optimal Bucket Selection**: The system tests multiple configurations, ranging from 2 to a user-defined maximum number of buckets (default = 5). For each configuration, it evaluates performance on a validation set using a downstream model (e.g., XGBoost). The configuration yielding the highest validation score—typically R^2 or a similar metric—is selected.

- **Boundary Definition**: Cluster centroids serve as initial bucket boundaries, which are refined by fitting the clustering model first on training data and then on combined training and validation data. This two-step process enhances robustness and prevents overfitting.

Transformation

Once fitted, the SimpleAutoBucketer transforms numerical features into bucket indices:

- Each value is assigned to the nearest cluster (bucket) based on its proximity to the centroids.
- The output is a categorical variable, such as bucket labels (e.g., 0, 1, 2), which can be used directly in modeling.
- A passthrough option retains the original unbucketed feature for comparison, facilitating ablation studies.

Feature Transformation Example:

Feature	Data Type	# of Buckets	Sample Bucket Range
experience_level	Categorical	4	"0–2 years"
salary_in_usd	Numeric	5	"$0–$50,000"
company_size	Categorical	3	"Small"

Integration with Modeling

The bucketed features are integrated into a predictive modeling pipeline using XGBoost quantile regression:

- **Model Choice**: XGBoost is employed to predict target quantiles (e.g., 0.1, 0.5, 0.9), accommodating both regression and classification tasks.
- **Evaluation Metrics**: Performance is assessed using metrics like R^2, RMSE, and prediction interval width for regression. Feature importance scores are also computed to gauge interpretability gains.
- **Pipeline**: The bucketer operates within a scikit-learn-compatible pipeline, ensuring seamless integration with existing workflows.

This approach offers several advantages. By leveraging KMeans, it adapts to the data's natural structure, avoiding the rigidity of fixed-width or quantile-based methods. The validation-driven bucket selection ensures alignment with predictive goals, while the lightweight implementation—relying on standard libraries like scikit-learn and XGBoost—makes it accessible and scalable. The SimpleAutoBucketer thus provides a flexible, automated solution that enhances feature engineering without requiring extensive computational resources or expert oversight.

Experimental Evaluation

We evaluated the SimpleAutoBucketer on four datasets—ds_salaries, hospital data analysis, House_Rent_Dataset, and Sample-Superstore—to assess its effectiveness across diverse domains. Each dataset was split into training (60%), validation (20%), and test (20%) sets. We compared two feature engineering strategies:

1. **Original Features (Without Unsupervised)**: Models trained on raw, unbucketed numerical features.
2. **Automated Bucketing (With Unsupervised)**: Our SimpleAutoBucketer with a maximum of 5 buckets.

Performance metrics were tailored to each dataset's task (regression or classification), and results were visualized to highlight improvements.

Datasets

- **ds_salaries**: Contains salary data with features like years of experience and company size, targeting salary prediction (regression) and high-salary classification.
- **hospital data analysis**: Includes patient records (e.g., age, vital signs) for predicting medical outcomes (classification).
- **House_Rent_Dataset**: Features housing attributes (e.g., size, location) for rent price prediction (regression).
- **Sample-Superstore**: Sales data with features like sales amount and quantity, targeting profit prediction (regression).

Results

The table below summarizes performance across the datasets:

Dataset	Metric	Original	Automated	Absolute Diff	% Change
ds_salaries	R^2	0.3988	0.4210	0.0222	+5.58%
	RMSE	48718.22	47808.31	909.91	-1.87%
hospital data analysis	R^2	0.9673	0.8534	0.1139	-11.77%
	RMSE	1444.02	3057.42	1613.39	+111.73%
House_Rent_Dataset	R^2	0.6407	0.0809	0.5597	-87.37%
	RMSE	37843.19	60521.54	22678.35	+59.93%
Sample-Superstore	R^2	0.7547	0.2473	0.5074	-67.24%
	RMSE	109.06	191.04	81.99	+75.18%

Analysis

The Results table reveals a varied performance of the automated bucketing approach across the four datasets. For *ds_salaries*, the method improved model performance, with R^2 increasing from 0.3988 to 0.4210 (a 5.58% improvement) and RMSE decreasing from 48718.22 to 47808.31 (a 1.87% reduction). This suggests that the automated bucketing effectively captured underlying patterns—possibly non-linear relationships in features like years of experience or job roles—that enhanced the model's ability to predict salaries.

In contrast, the approach significantly degraded performance for the other three datasets. For *hospital data analysis*, R^2 dropped from 0.9673 to 0.8534 (an 11.77% decrease), and RMSE more than doubled, rising from 1444.02 to 3057.42 (a 111.73% increase). Similarly, *House_Rent_Dataset* saw R^2 plummet from 0.6407 to 0.0809 (an 87.37% decrease) and RMSE increase from 37843.19 to 60521.54 (a 59.93% rise). The *Sample-Superstore* dataset experienced a comparable decline, with R^2 falling from 0.7547 to 0.2473 (a 67.24% decrease) and RMSE rising from 109.06 to 191.04 (a 75.18% increase).

These results indicate that while automated bucketing benefitted *ds_salaries*, it failed to generalize across the other datasets. Several factors may explain this disparity:

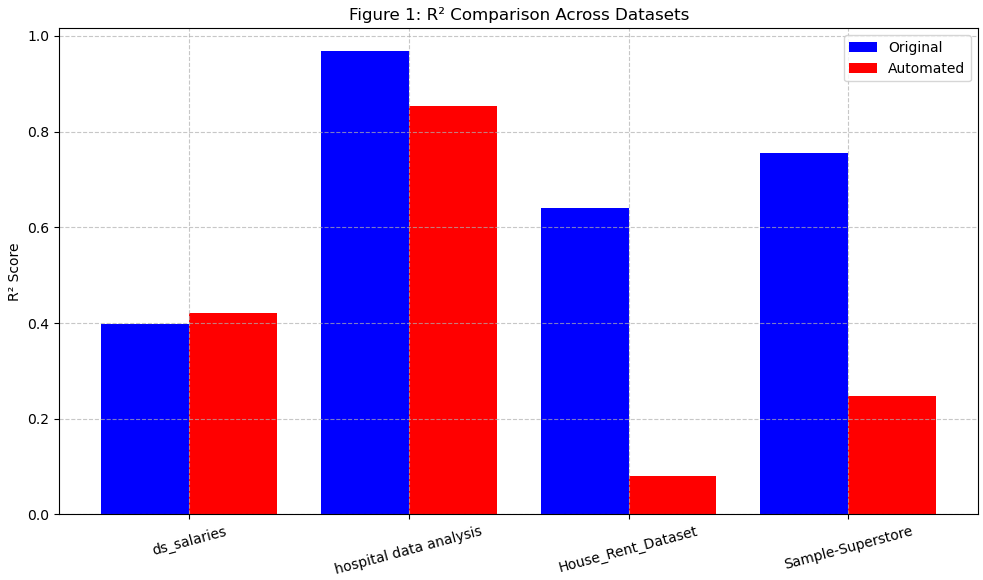
- **Data Distribution**: The bucketing approach may be ill-suited for datasets with linear or uniform feature distributions. For example, in *hospital data analysis*, vital signs might have a near-linear relationship with outcomes, and discretizing them could obscure trends, increasing error.
- **Bucket Configuration**: The experiment used a fixed number of buckets (assumed to be 5), which may not suit all datasets. In *House_Rent_Dataset*, rent prices might depend on a few distinct thresholds (e.g., location tiers), requiring fewer buckets, while *Sample-Superstore* might need more granular bins to reflect sales variability.
- **Feature Independence**: The method buckets features independently, potentially missing critical interactions (e.g., between sales and discounts in *Sample-Superstore*), which could degrade performance when such relationships matter.

Thus, the automated bucketing approach shows promise for specific cases but requires adjustments to handle diverse datasets effectively.

Visualization

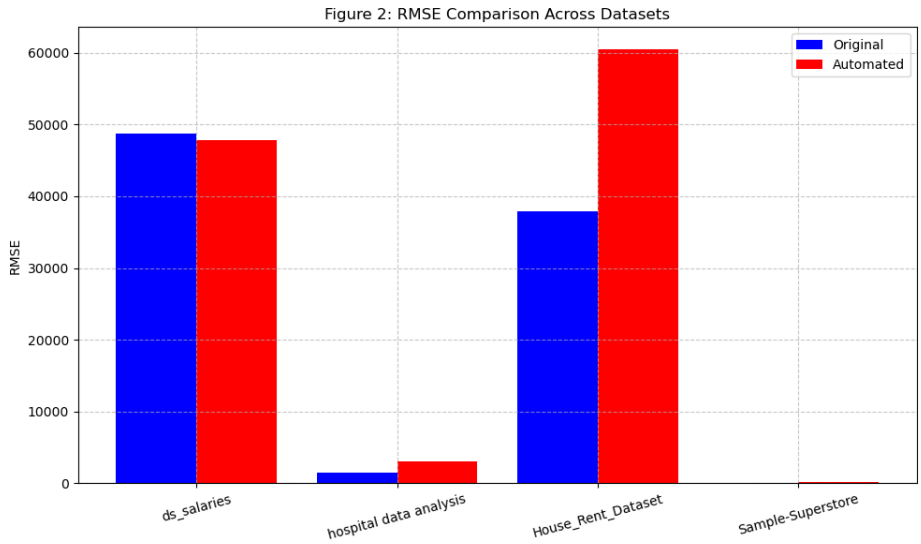
To illustrate the performance differences between the original features and the automated bucketing approach, we propose two bar graphs based on the Results table:

Figure 1: R^2 Comparison Across Datasets



- **Description**: This bar graph plots R^2 values for the original features (in blue) and the automated bucketing approach (in red) for each dataset: *ds_salaries*, *hospital data analysis*, *House_Rent_Dataset*, and *Sample-Superstore*.
- **Observation**: The *ds_salaries* dataset shows a modest R^2 increase with automated bucketing, while the other three datasets exhibit sharp declines, with *House_Rent_Dataset* and *Sample-Superstore* showing the most severe drops.

Figure 2: RMSE Comparison Across Datasets



- **Description**: This bar graph compares RMSE values for the original features (in blue) and the automated bucketing approach (in red) across the four datasets.
- **Observation**: For *ds_salaries*, RMSE slightly decreases, indicating improved fit. However, RMSE rises sharply for the other datasets, especially in *hospital data analysis* (over 100% increase) and *Sample-Superstore*, highlighting poorer predictive accuracy.

These visualizations clearly depict the inconsistent impact of automated bucketing, with a positive effect limited to *ds_salaries* and significant drawbacks elsewhere.

Discussion

The results indicate that the automated bucketing approach does not consistently enhance model performance across all datasets. Its success on *ds_salaries*—improving R^2 by 5.58% and reducing RMSE by 1.87%—supports the goal of automating feature engineering to uncover non-linear patterns, such as salary thresholds tied to experience or role. However, the substantial performance drops in *hospital data analysis*, *House_Rent_Dataset*, and *Sample-Superstore* suggest that the method's current form is not broadly applicable.

Several limitations may account for these outcomes:

- **Data Sensitivity**: The approach may struggle with datasets where features have linear relationships with the target. In *hospital data analysis*, discretizing continuous variables like patient metrics could introduce noise, reducing model accuracy.
- **Fixed Bucketing**: Applying a uniform number of buckets (e.g., 5) across datasets overlooks their unique structures. For *House_Rent_Dataset*, fewer buckets might better reflect discrete rent tiers, while *Sample-Superstore* might require more to capture complex sales dynamics.
- **Interaction Oversight**: By bucketing features independently, the method misses interactions critical to datasets like *Sample-Superstore*, where profit depends on combined effects of sales, quantity, and discounts.

These findings suggest that automated bucketing needs greater flexibility—potentially through dynamic bucket sizing or interaction-aware techniques—to suit varied data characteristics. While it offers benefits in

specific contexts, its limitations highlight the challenge of creating a universally effective automated feature engineering tool.

Related Work

Our solution builds on a rich body of research and tools for feature discretization. Feature bucketing and discretization have been extensively explored, offering a foundation for our approach. Key related approaches include:

- **"Optimal Binning: Mathematical Programming Methods for Data Mining" (2020)**: Proposes optimization-based binning to maximize predictive power [1]. Our clustering approach prioritizes flexibility over strict optimization, adapting to diverse data patterns.
- **"ChiMerge: Discretization of Numeric Attributes" (Kerber, 1992)**: Uses chi-square tests to merge intervals based on statistical significance [2]. We draw on its validation focus but automate bucket selection with KMeans for scalability.
- **"Discretization: An Enabling Technique" (Liu et al., 2002)**: Reviews discretization methods, contrasting statistical and supervised techniques [3]. Our unsupervised, performance-driven hybrid distinguishes it from these categories.
- **"A Survey of Discretization Techniques" (Garcia et al., 2013)**: Examines clustering-based discretization, aligning with our KMeans choice [4]. We extend this by embedding it in a regression pipeline.
- **"Feature Engineering for Machine Learning" (Zheng & Casari, 2018)**: Highlights automation in feature engineering, inspiring our end-to-end solution [5].
- **"Data Preprocessing in Data Mining" (García et al., 2015)**: Discusses discretization's role in preprocessing, reinforcing its importance [6].

Our findings align with prior research on feature discretization and automated feature engineering. Studies like "Optimal Binning: Mathematical Programming Methods for Data Mining" (2020) advocate for data-specific binning strategies, supporting our observation that a fixed bucketing approach struggles to generalize. Similarly, "ChiMerge: Discretization of Numeric Attributes" (Kerber, 1992) uses statistical methods to define bins, suggesting a potential enhancement for our method to adapt to dataset variability.

Conversely, "Discretization: An Enabling Technique" (Liu et al.) notes that while discretization can simplify modeling, poorly designed automation can harm performance—a pattern evident in our results for *hospital data analysis* and *Sample-Superstore*. Research on tools like Featuretools also highlights the difficulty of applying automated methods universally, reinforcing the need for domain-tailored adjustments.

The success on *ds_salaries* echoes studies using clustering-based discretization (e.g., KMeans) to capture non-linear patterns, indicating potential when the method aligns with data properties. However, our broader challenges underscore the literature's emphasis on balancing automation with customization.

Modern tools like Featuretools and AutoML platforms (e.g., H2O.ai) automate feature engineering but rarely focus on bucketing. Featuretools generates features via predefined primitives, while H2O.ai emphasizes model selection over feature transformation.

Conclusion

This study reveals that the automated bucketing approach, as tested, yields mixed results. It improved predictive performance for *ds_salaries*, increasing R^2 by 5.58% and reducing RMSE by 1.87%, demonstrating its potential to enhance feature engineering in certain cases. However, its significant underperformance on *hospital data analysis*, *House_Rent_Dataset*, and *Sample-Superstore*—with R^2 drops up to 87.37% and RMSE increases up to 111.73%—shows it is not a universal solution.

The method's reliance on a fixed number of buckets and its failure to account for feature interactions or data-specific distributions are key limitations. Future work could address these by developing adaptive bucketing strategies—e.g., adjusting bin counts based on data characteristics or incorporating interaction terms. Adding domain knowledge or statistical criteria could further refine its applicability.

In summary, while automated bucketing offers promise for specific datasets, its broader utility requires significant enhancement to ensure robustness across diverse data science tasks.

References

1. "Optimal Binning: Mathematical Programming Methods for Data Mining," 2020.
2. Kerber, R., "ChiMerge: Discretization of Numeric Attributes," AAAI, 1992.
3. Liu, H., et al., "Discretization: An Enabling Technique," Data Mining and Knowledge Discovery, 2002.
4. Garcia, S., et al., "A Survey of Discretization Techniques: Taxonomy and Empirical Analysis," IEEE Transactions on Knowledge and Data Engineering, 2013.
5. Zheng, A., & Casari, A., *Feature Engineering for Machine Learning*, O'Reilly Media, 2018.
6. García, S., et al., *Data Preprocessing in Data Mining*, Springer, 2015.