Comparing Manual and Automated Feature Engineering: A Kaggle Case Study

Cameron Rader, Suneil Patel, Smit Patel
Department of Computer Science, University of Tennessee, Knoxville
Email: {crader6, spate200, spate149}@vols.utk.edu

Abstract—**Background: Feature engineering transforms raw data into representations that improve machine learning performance, interpretability, and computational efficiency. This study systematically compares expert-driven manual selection with three automated techniques—PCA, autoencoder compression, and tree-based importance ranking—on a standard benchmark. The goal is to provide practitioners with clear guidance on method selection based on dataset characteristics, resource constraints, and interpretability requirements.**

Methods: We preprocess the Kaggle Titanic dataset (891 records, 12 attributes) via imputation, encoding, and scaling, then apply four pipelines: (i) manual feature selection based on domain knowledge and statistical tests, (ii) PCA retaining $\geq 90\%$ explained variance to reduce dimensionality, (iii) a PyTorch autoencoder compressing to a 5-dim latent space to capture non-linear patterns, and (iv) Random Forest Gini-based importance selecting the top eight features for transparency. Each pipeline feeds a Random Forest classifier evaluated by stratified 5-fold cross-validation on accuracy, precision, recall, F1-score, and wall-clock training time to balance predictive power and computational cost.

Results: Manual selection achieves the highest accuracy (0.8137) but has the longest training time (1.81 s), highlighting the cost of expert curation. PCA and autoencoder pipelines reduce training to 0.45 s with moderate accuracy declines (0.7733 and 0.7756, respectively), demonstrating efficiency gains. Importance-based ranking recovers much of manual performance (0.8036) with minimal compute (0.42 s), underscoring its role as a hybrid approach.

Conclusions: Expert-informed manual features remain a robust baseline for small structured datasets where interpretability and accuracy are paramount. Automated methods offer significant speedups and can serve as first-pass or complement manual pipelines. A hybrid approach combining manual insight with importance ranking delivers an optimal trade-off among accuracy, interpretability, and efficiency, especially under resource constraints.

I. Introduction

A. Motivation

Feature engineering is the process of creating or selecting input variables that capture the underlying signal relevant to a predictive task. In domains such as finance, healthcare, and risk management, the quality of these features directly impacts model accuracy, generalization to new data, and the capacity to explain and justify decisions. Poorly chosen features can introduce noise, increase model complexity, and lead to overfitting, while effective features can simplify models, accelerate training, and provide actionable insights. Given the growing emphasis on model accountability and efficiency, understanding the trade-offs between manual and automated feature engineering is essential for both researchers and practitioners.

B. Related Work and Research Gap

Manual feature engineering has a long tradition in structured data analysis, with domain experts crafting variables based on business knowledge and statistical observations [1]. Automated dimensionality reduction techniques, such as Principal Component Analysis (PCA) [2], transform features into orthogonal components that retain maximum variance but often lose semantic interpretation. Neural autoencoders [3], [4] extend this concept to learn non-linear compressed representations via reconstruction objectives, enabling more flexible dimensionality reduction at the cost of interpretability. Tree-based importance measures, derived from Random Forests and related ensembles, offer automatic feature ranking while preserving original feature semantics [5]. However, existing literature typically focuses on pairwise comparisons or specific domains, and few studies have systematically benchmarked all four approaches on a small, tabular dataset with mixed data types under identical evaluation criteria.

C. Research Questions

We address the following questions to fill this gap:

1) **Predictive Performance**: How does each feature engineering pipeline affect classification accuracy, precision, recall, and F1-score?
2) **Computational Efficiency**: What are the relative training times and resource demands of manual versus automated methods?
3) **Interpretability**: Which approaches retain feature transparency, and how does this impact potential deployment in high-stake environments?

II. Methods

A. Dataset and Preprocessing

We utilize the Kaggle Titanic ”train.csv” dataset, consisting of 891 passenger records with 12 initial attributes including demographics, ticket information, and survival outcome. The preprocessing pipeline comprises:

1) **Imputation:** Missing values for `Age` (19%) are replaced by the median age to minimize distortion by outliers; missing `Embarked` entries (0.2%) receive the most frequent port of embarkation to maintain distributional integrity.
2) **Encoding:** Categorical variables (`Sex`, `Embarked`) undergo one-hot encoding with one level dropped to prevent multicollinearity, resulting in dummy variables that represent passenger characteristics in a numeric format.
3) **Scaling:** Continuous features (`Pclass`, `Age`, `SibSp`, `Parch`, `Fare`) and generated dummies are standardized to zero mean and unit variance, ensuring comparable scales across all inputs and enhancing convergence for some algorithms.

This preprocessing produces a consistent 10-dimensional feature space for all downstream pipelines.

B. Feature Engineering Techniques

We implement and compare the following four pipelines on the preprocessed features:

1) Manual Selection: Eight features are selected using domain knowledge and statistical analyses: `Pclass`, `Age`, `SibSp`, `Parch`, `Fare`, `Sex_male`, `Embarked_Q`, and `Embarked_S`. We evaluated feature distributions, survival rate stratifications, and Pearson correlations to inform this choice, emphasizing variables with the strongest univariate predictive power.

2) Principal Component Analysis (PCA): PCA is applied to the full 10-dimensional dataset. We retain the first five principal components that explain at least 90% of total variance, as determined by a cumulative variance scree plot and the Kaiser criterion. The resulting orthogonal components serve as compressed, uncorrelated features, facilitating dimensionality reduction while preserving information.

3) Autoencoder: A feedforward autoencoder is implemented in PyTorch with architecture: Input (10) → Dense(64, ReLU) → Dense(5, ReLU) → Dense(64, ReLU) → Output (10, linear). The network is trained for 50 epochs on mean squared error (MSE) reconstruction loss using the Adam optimizer (learning rate 1e3), with an 80/20 train/validation split to monitor convergence and prevent overfitting. The 5-dimension bottleneck outputs become the new feature set, capturing non-linear relationships among inputs.

4) Tree-based Importance Ranking: We fit a Random Forest classifier (100 trees, Gini impurity) on all standardized inputs. Gini-based importance scores are extracted for each feature, quantifying its contribution to impurity reduction across the ensemble. The top eight features by importance are selected, preserving their original semantic meaning for transparent downstream modeling.

C. Model Training and Evaluation

Each pipeline's feature set is used to train a Random Forest classifier (100 trees, `random_state`=42). We perform stratified 5-fold cross-validation to ensure consistent class distributions across folds. Performance metrics include:

- **Accuracy**: overall correct classification rate, reflecting general model quality.
- **Precision**: fraction of predicted positives that are true positives, measuring false positive control.
- **Recall**: fraction of actual positives correctly identified, measuring false negative control.
- **F1-Score**: harmonic mean of precision and recall, balancing both aspects.
- **Training Time**: wall-clock time required to train the model on each fold, capturing computational cost.

All experiments are conducted on a 16GB RAM CPU environment running Python 3.9.

III. Results

A. Quantitative Summary

Table I reports mean performance metrics and training times across cross-validation folds. Manual selection yields the highest accuracy (0.8137) but requires the most compute time. PCA and autoencoder reduce training time by approximately 75% while incurring a modest accuracy drop of 4 percentage points. Importance-based ranking recovers most manual performance (0.8036) under the same time reduction, demonstrating its effectiveness as a hybrid strategy.

TABLE I: Cross-validated performance and training time

Method	Accuracy	Precision	Recall	F1-Score	Time (s)
Manual	0.8137	0.7720	0.7336	0.7501	1.8124
PCA	0.7733	0.7076	0.6987	0.7025	0.4653
Autoenc.	0.7756	0.7032	0.7162	0.7092	0.4244
Imp(sel)	0.8036	0.7575	0.7191	0.7365	0.4190

B. Visual Comparison

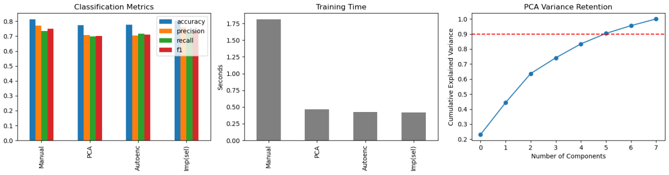

Fig. 1: (Left) accuracy, precision, recall, F1; (Middle) training time; (Right) PCA cumulative variance.

C. Interpretation

Expert-driven manual selection offers the highest accuracy, validating the value of domain expertise in crafting features. However, it requires the greatest computational resources. PCA and autoencoders both achieve substantial reductions in training time—critical for rapid iteration or deployment in

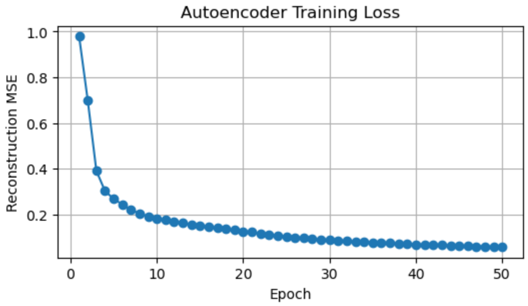

Fig. 2: Autoencoder reconstruction loss over 50 epochs.

constrained environments—while maintaining reasonable predictive performance. Tree-based importance ranking emerges as an effective hybrid, nearly matching manual accuracy while preserving feature interpretability and minimizing computational cost.

IV. Limitations and Validity

A. *Internal Validity*

Our study uses fixed hyperparameters across pipelines, which may favor methods more robust to default settings. Employing nested cross-validation or automated hyperparameter optimization would strengthen internal validity.

B. *External Validity*

The Titanic dataset is a small, structured benchmark. Results may differ on larger or unstructured datasets (e.g., images, text) where feature distributions and complexity vary significantly.

C. *Construct Validity*

We equated interpretability with the use of original feature semantics; this simplification may not capture the full human understanding of transformed features. Incorporating user studies or model-agnostic explanation tools (e.g., SHAP) could provide a more nuanced assessment.

D. *Future Work*

Future research could investigate hybrid pipelines that combine expert feature seeds with iterative importance pruning, integrate explainable AI frameworks for deeper interpretability analysis, and benchmark across diverse datasets with statistical significance testing to confirm generalizability.

V. Conclusion

Our comprehensive comparison demonstrates that while manual feature engineering remains a powerful baseline—particularly when interpretability and accuracy are crucial—automated methods offer significant efficiency gains. A hybrid strategy, leveraging initial expert insights followed by data-driven importance selection, provides an optimal balance of performance, transparency, and computational cost.

Acknowledgment

We thank the University of Tennessee's ISAAC cluster for computational resources.

References

[1] L. G. Nyúl and J. K. Udupa, ”On standardizing the MR image intensity scale,” *Magn. Reson. Med.*, vol. 42, pp. 1072–1081, 1999.

[2] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. Springer, 2002.

[3] G. E. Hinton and R. R. Salakhutdinov, ”Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[4] Y. Bengio et al., ”Greedy layer-wise training of deep networks,” in *Proc. NIPS*, 2007, pp. 153–160.

[5] S. M. Lundberg and S.-I. Lee, ”A unified approach to interpreting model predictions,” in *NeurIPS*, 2017.

[6] Kaggle, ”Titanic - Machine Learning from Disaster,” Kaggle, 2012. [Online]. Available: https://www.kaggle.com/c/titanic/data. [Accessed: May 6, 2025].