

Feature Engineering in Modern Predictive Modeling: Techniques, Automation Frameworks and Emerging Trends

Jyoti Panchariya¹, Surendra Kumar Yadav^{2,*}, Shruti Mathur³

Abstract

Feature engineering plays a crucial role in developing reliable and high-performing predictive models. It involves transforming raw and unprocessed data into meaningful features that help machine learning algorithms achieve better accuracy and overall performance. Feature engineering is helpful in identifying hidden patterns, removing noise, and increasing the signal-to-noise ratio in data sets. Machine learning techniques have progressed remarkably over the years, and the effectiveness of these models largely depends on how informative and well-structured the extracted features are. This paper presents a comprehensive review of the traditional and advanced techniques in feature engineering for building predictive models. Techniques in Feature Engineering: Feature selection techniques, Feature extraction techniques, Feature construction techniques, Feature transformation techniques, Feature scaling techniques. Furthermore, new developments in feature engineering techniques such as Automated Feature Engineering (AutoFE), Deep Feature Synthesis, feature generation techniques in AutoML, etc., are reviewed. Moreover, recent developments in the domain of explainable artificial intelligence, such as feature importance techniques like SHAP values and permutation importance, are reviewed, considering the importance of the role played by explainability in machine learning systems to tackle key challenges such as high-dimensional data, feature redundancy, privacy, etc., in machine learning systems. Finally, new directions for future work are suggested, such as federated feature engineering, causal feature discovery, graph-based learning, etc. This review article aims to assist researchers and practitioners in building a systematic understanding of various feature engineering techniques.

Keywords: Graph neural networks, AutoML, feature tools, SHAP, federated learning, feature engineering

*Author for Correspondence

Surendra Kumar Yadav
E-mail: surendrakumar.yadav@jecrcu.edu.in

¹M. Tech. Scholar, Department of Computer Science and Engineering, JECRC University, Jaipur, Rajasthan, India.

²Professor, Department of Computer Science and Engineering, JECRC University, Jaipur, Rajasthan, India.

³Assistant Professor, Department of Computer Science and Engineering, JECRC University, Jaipur, Rajasthan, India.

Received Date: February 05, 2026

Accepted Date: June 09, 2026

Published Date: August 10, 2026

Citation: Jyoti Panchariya, Surendra Kumar Yadav, Shruti Mathur. Feature Engineering in Modern Predictive Modeling: Techniques, Automation Frameworks and Emerging Trends. Journal of Computer Technology & Applications. 2026; 17(2): 32–40p.

INTRODUCTION

Appropriate data representation is essential for predictive modelling. Despite the powerful inductive mechanisms offered by contemporary learners, such as ensembles and deep networks, feature quality continues to be the most significant factor influencing performance in the real world. Feature engineering is the process of converting unprocessed, noisy, or insufficient data into variables that improve the signal-to-noise ratio for learning algorithms. This review places recent automated and AI-driven methods alongside traditional statistical approaches, adhering to the taxonomy of feature engineering techniques (Figure 1).

Three observations serve as the basis for our review: (1) many tabular and domain-specific tasks still require handcrafted features; (2) automated approaches, such as Deep Feature Synthesis, reduce manual labor but present interpretability and scalability issues [1]; and (3) new deployment contexts, such as federated systems and graph-structured data, necessitate reconsideration of feature pipelines for privacy and relational inductive biases [2, 3].

HISTORY AND PURPOSE

Raw datasets frequently contain redundant variables, skewed distributions, high dimensionality, and missing values, all of which reduce the learning efficacy and increase the possibility of overfitting. In the past, domain experts created indicators and transformations specific to the context of the problem as part of the manual, expert-driven feature engineering phase of the analytics lifecycle. According to traditional textbooks and surveys, meticulous feature design frequently outperforms the haphazard application of complex models, particularly in noisy or low-data environments [4, 5]. Automating feature discovery (AutoFE) and incorporating feature engineering into end-to-end AutoML pipelines have been the subjects of recent research. Relational feature synthesis is operationalized by tools such as Featuretools, and AutoML systems use pipeline search and selection to suggest useful feature sets. However, there are trade-offs associated with automation; engineers must weigh interpretability, computational cost, and fairness considerations against predictive gains [1, 6].

TAXONOMY OF FEATURE ENGINEERING TECHNIQUES

We categorize methods into five main groups: scaling, transformation/encoding, extraction, construction, and selection. Practical heuristics and algorithmic subfamilies are included in each group Figure 2.

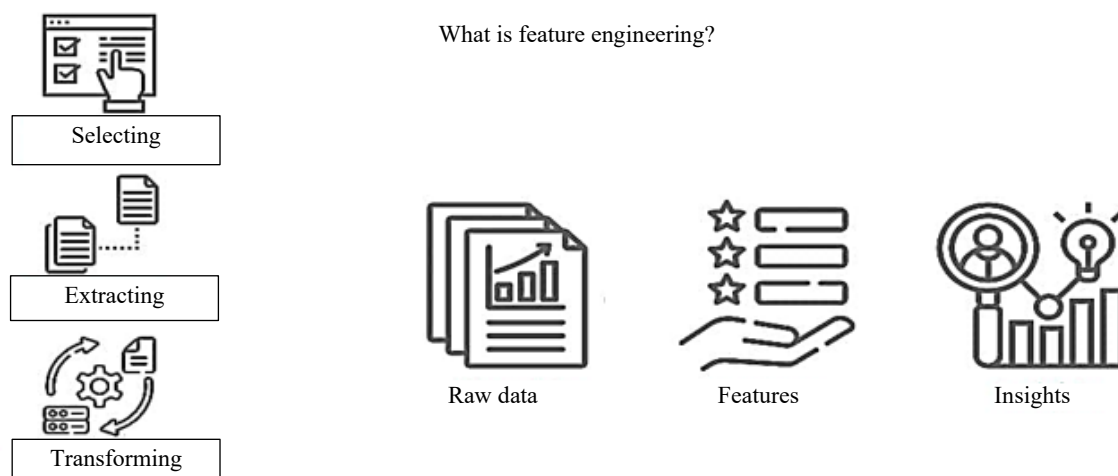


Figure 1. The general workflow of feature engineering.

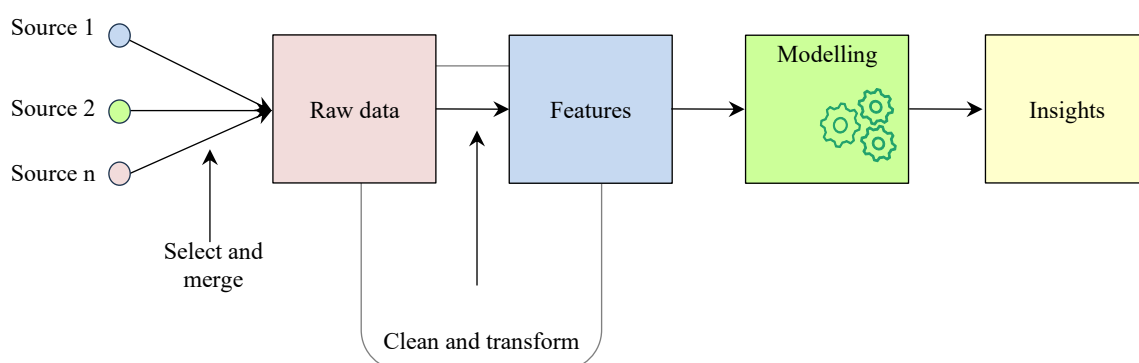


Figure 2. The taxonomy of feature engineering techniques includes multiple categories.

Feature Selection

Feature selection lowers dimensionality by retaining the most informative predictors and eliminating redundant or noisy variables. Typically, selection techniques are divided into three categories: filter, wrapper, and embedded. Filter methods assess features separately from the learning algorithm using criteria such as mutual information, chi-square for categorical associations, correlation thresholds, or variance filters. These techniques scale to high dimensions and are computationally inexpensive; however, they might overlook interactions between features [7].

Wrapper techniques (such as forward/backward selection and recursive feature elimination) search for subsets of features assessed by a selected predictive model. Although they are more computationally demanding and prone to overfitting small datasets, wrappers can capture interaction effects [8]. Selection is incorporated into model training through embedded methods: tree-based models offer importance scores, L1-regularized models (LASSO) reduce coefficients to zero, and more recent techniques incorporate sparsity-inducing priors into neural architectures [9]. Efficiency and performance are balanced using embedded techniques.

Feature Extraction and Representation Learning

New variables that compactly represent the original data are generated through feature extraction. Principal Component Analysis (PCA), a linear technique that projects data onto orthogonal axes that capture the maximum variance, has long been used for decorrelation and dimensionality reduction [10]. In lower dimensions, supervised variants such as Linear Discriminant Analysis (LDA) seek to maximize class separability [11]. Autoencoders, variational autoencoders, and manifold learning techniques are examples of non-linear learned representations that are especially useful for images, signals, and intricate tabular interactions. These representations learn compact encodings [12]. In domains with a large amount of labelled data, deep models trained end-to-end can reduce the need for manual pre-processing by incorporating feature extraction into representation learning [13].

Feature Construction

Using domain knowledge and arithmetic/functional transformations, feature construction generates derived variables such as polynomial features, interaction terms, ratios, time-window aggregations (counts, rolling means), and domain-specific indices (e.g., BMI in healthcare). Frequently, the constructed features offer comprehensible signals that downstream models can easily exploit [14]. Algorithmic feature engineering toolkits frequently employ automated construction algorithms that list potential transformations and assign scores to them (for example, based on information gain or validation performance) [1].

Feature Transformation and Encoding

The scales and feature distributions were standardized through transformations. Common procedures include normalization (min-max), standardization (z-score), logarithmic and power transforms for skewed data, and Box-Cox transformations when necessary. It is essential to encode categorical variables: There are specific trade-offs between cardinality and model compatibility for one-hot encoding, ordinal encoding, target (mean) encoding, and embedding-based encoding [15]. In contrast to naive one-hot schemes, target encoding and learned embeddings (from supervised neural networks or entity embedding techniques) reduce dimensionality by compressing high-cardinality categories into dense, informative vectors [16].

Feature Scaling and Normalization

When numerical features have similar scales, many gradient-based learners (such as logistic regression and neural networks) perform better. Optimization stability and convergence are enhanced by scaling methods (e.g., min-max or standardization). Scaling is less important for tree-based models, but it is still helpful for distance-based techniques or regularized linear components [17].

AUTOMATED FEATURE ENGINEERING (AUTOFE)

Automated Feature Engineering seeks to minimize human labor by automatically creating, choosing, and validating candidate features. By stacking aggregation and transformation primitives across entity relationships, Deep Feature Synthesis (DFS), which is implemented in Featuretools, generates relational features [1]. Although DFS has made it possible to generate features quickly in business settings, it must be carefully validated to prevent overfitting and feature leakage. AutoML toolkits expand automation to include feature processing, pipeline searching, and hyperparameter optimization. According to benchmarks, AutoML systems (H2O AutoML, AutoGluon, and TPOT) can significantly reduce the amount of human time required to complete many tabular tasks while approaching expert-crafted pipelines [18]. Recent studies and surveys have assessed the reproducibility and usability of AutoFE, pointing out that automated features frequently require manual curation to guarantee interpretability and business relevance [4, 19].

EXPLAINABILITY AND FEATURE IMPORTANCE

In regulated fields like healthcare and finance, feature effect interpretation is crucial. SHAP (Shapley Additive exPlanations) is a common tool for feature importance and model explanation that exPlanations offers a unified framework for local and global feature attributions based on cooperative game theory [20]. Permutation importance and LIME are complementary techniques for global sensitivity analysis and local explanation, respectively. Explainability tools aid in validating the engineered features. It is important to use fairness and robustness checks in conjunction with explanations because features with high SHAP importance may still encode undesired biases or proxy-sensitive attributes [21].

EVALUATION METRICS AND PROTOCOLS

Robust experimental protocols are required to evaluate the engineered features. Common techniques include careful temporal validation for time-series tasks, nested cross-validation when model selection is involved, and holdout validation. The problem framing should be reflected in the metrics (accuracy, F1-score for classification, RMSE, and MAE for regression). Measures of effect size, permutation importance, and feature ablation studies quantify contribution. When implementing pipelines in production, it is essential to assess the computational cost (memory and feature construction time).

APPLICATIONS AND CASE STUDIES

Predictive systems across domains are based on feature extraction. In marketing, customer lifetime value and recency-frequency-monetary features support churn modelling; in finance, crafted features from transaction histories and temporal aggregations power fraud detection; and in healthcare, engineered features from EHR time series (trended vitals and lab-derived indices) enhance prognostic models. Frequency-domain features and sensor-level aggregations are frequently used in industrial applications for predictive maintenance [22, 23]. Feature extraction methods that consider relational inductive biases are beneficial for graph-structured data. In networked domains, graph neural networks (GNNs) offer an alternative to manual relational feature construction by performing implicit feature learning over node and edge attributes [24].

CHALLENGES AND OPEN PROBLEMS

Despite these advancements, feature engineering still faces the following challenges: (1) scalability for large, streaming datasets; (2) privacy requirements in cross-organizational settings, which drive federated feature engineering approaches; (3) preserving interpretability when automated systems produce intricate composite features; and (4) robust causal feature selection that prevents learning spurious correlations [3, 25]. Federated automated feature engineering is an emerging field that addresses privacy-preserving feature synthesis across distributed data holders. Methods for horizontal, vertical, and hybrid federated feature generation with secure aggregation mechanisms were proposed in early studies [3].

FUTURE DIRECTIONS

Incorporating causal discovery tools into feature pipelines to rank causally significant variables, enhance the explainability of AutoFE outputs, streamline feature pipelines in the face of resource limitations (memory, latency), and expand AutoFE to federated and continual learning environments are important avenues for future research. For low-data problems, graph-based and generative model approaches may open up new classes of relational and synthetic features (Figure 3).

COMPARATIVE ANALYSIS OF FEATURE ENGINEERING TECHNIQUES

Feature engineering approaches vary considerably with regard to complexity, interpretability, computational cost, and effect on the performance of machine learning models. The choice of the best approach for feature engineering may depend on the characteristics of the data and the machine learning algorithm used for predictive modeling. A comparative analysis of the approaches provides a better understanding of the effectiveness of the feature engineering process.

Feature selection approaches are regarded as the simplest approach to enhancing the performance of machine learning models. These approaches involve selecting the most important features and filtering out redundant and irrelevant features from the dataset. This feature selection process reduces the complexity of the machine learning models and prevents overfitting. Filter methods such as correlation analysis, mutual information, and chi-square tests are popular because of their simplicity and computational speed. However, these methods only consider features individually and may not account for interactions between features.

Wrapper methods improve the limitations of filter methods by considering the combination of features for predictive models. Methods such as recursive feature elimination and forward selection can account for the interactions between features and may result in better predictive performance than filter methods. However, the computational cost of wrapper methods may be high for datasets with hundreds or thousands of features.

Embedded methods combine the benefits of filter and wrapper methods by considering feature selection during the training process of the models. Methods such as LASSO regression involve the addition of penalties to the model during training, which may result in the coefficients of less important features being zero. Similarly, models such as Random Forest and Gradient Boosting consider the importance of features during the training process and provide importance values for the features. Feature extraction methods are distinct from feature selection methods in that they change the original variables. Principal Component Analysis (PCA) is one of the most frequently employed feature extraction methods. PCA minimizes dimensionality by mapping data to principal components that maximize variance. Although PCA increases the efficiency of computations and reduces noise, it also compromises interpretability because the new features are linear combinations of the original variables. Feature extraction methods using deep learning have also become popular recently. Neural networks, especially CNN and autoencoders, are widely employed for feature extraction. These networks are capable of learning hierarchical representations from the data. They can also learn complex nonlinear relationships between variables that may be overlooked by traditional feature engineering methods.

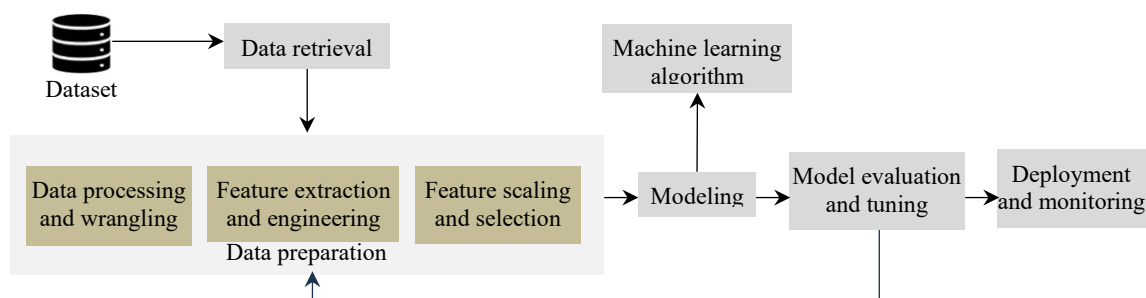


Figure 3. The future research directions are summarized.

Another important aspect of the methodologies of feature engineering is the importance of feature construction, as it is also significant. Feature extraction methodologies are based on mathematical operations, whereas feature construction methodologies are based on domain knowledge to develop new features based on the relationship features.

Some of the effective methodologies used in feature engineering are interaction terms, ratios, and time series aggregation, as they can be effective in increasing the accuracy of the predictions. These features are also more interpretable, as they are meaningful.

Another effective feature engineering methodology to reduce manual labor is Automated Feature Engineering (AutoFE), which is effective in generating features in relational datasets where complex relationships exist between multiple entities. Featuretools is a package used to generate features using various transformation and aggregation operations. However, thousands of features are generated using automated systems, and there is a greater chance of overfitting.

Overall, it can be concluded that all feature engineering techniques have unique advantages and disadvantages, and the best combination of techniques is required to achieve optimal performance.

IMPACT OF FEATURE ENGINEERING ON MACHINE LEARNING MODEL PERFORMANCE

The performance of machine learning models is highly dependent on the quality and relevance of input features. Regardless of how advanced a machine learning algorithm is, it will not be able to deliver good performance in terms of accuracy if the input features do not provide sufficient information.

Feature engineering is a crucial step in the transformation of raw data into a more structured form that can be easily utilized by machine learning algorithms.

Feature engineering is highly significant in traditional statistical models, such as logistic and linear regression. The key idea behind these models is to assume linear relationships between the input features and output variables. Therefore, polynomial relationships and interaction terms are often required to account for non-linear relationships.

Machine learning algorithms, such as Random Forest, Decision Trees, and Gradient Boosting, are more robust when handling raw data representations. These algorithms can automatically identify non-linear patterns and interactions in the data. However, these machine learning algorithms can be significantly enhanced using appropriate feature engineering. For example, aggregating features for temporal data or appropriate transformations for categorical data can substantially improve the accuracy of these algorithms.

In most real-world datasets, there is noisy, incomplete, or imbalanced data. Feature engineering can be used to overcome these challenges by enhancing the signal-to-noise ratio of the dataset. Normalization, scaling, and transformation of features ensure that all features in the dataset are represented consistently, irrespective of the magnitude of the data. This is particularly important for gradient-based optimization algorithms used in neural networks and logistic-regression models. Feature engineering is also important when working with high-dimensional data. In fields such as genomics, text mining, and sensor data analysis, there may be thousands of variables in the data set. This may cause overfitting and poor generalization performance. Techniques such as PCA, autoencoders, and feature selection methods are employed to reduce dimensionality and select the most important features, while removing redundant variables.

Recent breakthroughs in automated machine learning (AutoML) have incorporated feature engineering into the process of automated model development. AutoML systems can automatically

search through various feature transformations, feature selection methods, and model architectures to determine the most promising combinations. Although these systems have reduced the need for human intervention in the process, human expertise may be important for interpreting the automatically created features and determining their relevance to the domain.

Another key aspect of feature engineering is model interpretability. This is particularly important in critical domains such as healthcare, finance and risk prediction. In such cases, stakeholders need to have an easily interpretable model. Feature engineering helps improve model interpretability. For example, in healthcare, feature engineering can be used to build a model in which some features are risk scores. This helps domain experts understand the predictions made by machine learning models. Moreover, artificial intelligence techniques, such as SHAP values and permutation importance, can be used to understand which features contribute to a model's predictions. This helps researchers validate whether feature engineering has a positive effect on a model's performance or whether there are any biases. Feature engineering remains one of the most influential parts of machine learning.

INTEGRATION OF FEATURE ENGINEERING WITH MODERN AI SYSTEMS

In today's artificial intelligence systems, complex data sources are used in the form of images, text, graphs, etc. Therefore, feature engineering methods must be updated to deal with these complex data sources in complex computational environments. The integration of traditional feature engineering methods with advanced AI systems is a major research area in predictive modeling.

In deep learning systems, feature engineering is a major aspect integrated into the system architecture. In convolutional neural network systems, features are learned from input image data using convolution operations. Recurrent neural network systems are used to predict features from sequential data in the form of time-series data. The transformer model learns features from the input text data by considering long-range dependencies in the input data. These systems do not require any external feature preprocessing operations, which is still a major requirement. Another emerging area in which feature engineering plays an important role in machine learning models is in graph-based models. Many real-world datasets, such as social networks, financial transaction graphs, and biological interaction graphs, are naturally represented as graphs. In graph neural networks, the model learns the representation of the graph using neighboring nodes. In these models, node characteristics and structural and relational information are considered important engineered features for better predictive accuracy.

Another emerging trend in machine learning and data science is the development and deployment of federated learning systems for privacy-preserving machine learning. In federated learning systems, data are distributed across multiple organizations or devices, and raw data cannot be aggregated because of data privacy constraints. In federated learning systems, feature engineering must be performed in a distributed manner while preserving data secrecy. Federated feature selection, federated secure aggregation, and federated feature transformation are some of the methods currently being explored for federated learning systems. In addition, real-time predictive systems require effective feature pipelines that can process streaming data efficiently. Online feature engineering is a technique for building features that can process streaming data in real time. This technique includes real-time aggregation, incremental features, and features constructed based on time.

The combination of feature engineering and the latest advancements in AI technologies underscores the continued importance of data representation in machine learning research. Although machine learning algorithms continue to improve dramatically, the process of designing effective features still plays a fundamental role in developing effective predictive systems.

CONCLUSION

This aspect remains important for feature engineering and the construction of prediction models. This is because domain knowledge, validation, and interpretability will still be important, albeit with less

manual effort owing to the introduction of automation. This review provides guidelines for practitioners to build their feature pipelines using traditional feature selection/extraction and AutoFE approaches.

REFERENCES

1. Guyon I, Elisseeff A. An introduction to variable and feature selection. *Journal of machine learning research*. 2003;3(Mar):1157-82.
2. Kohavi R, John GH. Wrappers for feature subset selection. *Artificial intelligence*. 1997 Dec 1;97(1-2):273-324.
3. Breiman L. Random forests. *Machine learning*. 2001 Oct;45(1):5-32.
4. Bishop CM, Nasrabadi NM. *Pattern recognition and machine learning*. New York: springer; 2006 Aug 17.
5. Goodfellow I, Bengio Y, Courville A, Bengio Y. *Deep learning*. Cambridge: MIT press; 2016 Nov 18.
6. han, kamber, pei. *Data Mining Third Edition*. 2011. Available from: <https://myweb.sabanciuniv.edu/rdehkharghani/files/2016/02/The-Morgan-Kaufmann-Series-in-Data-Management-Systems-Jiawei-Han-Micheline-Kamber-Jian-Pei-Data-Mining.-Concepts-and-Techniques-3rd-Edition-Morgan-Kaufmann-2011.pdf>.
7. Kuhn M, Johnson K. *Feature engineering and selection: A practical approach for predictive models*. Chapman and Hall/CRC; 2019 Jul 25.
8. Shlens J. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*. 2014 Apr 3.
9. Lundberg SM, Lee SI. A unified approach to interpreting model predictions. *Advances in neural information processing systems*. 2017;30.
10. Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, Wang L, Li C, Sun M. Graph neural networks: A review of methods and applications. *AI open*. 2020 Jan 1;1:57-81.
11. Heaton JB, Polson NG, Witte JH. Deep learning in finance. *arXiv preprint arXiv:1602.06561*. 2016 Feb 21.
12. Chen T, Guestrin C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining 2016 Aug 13* (pp. 785-794).
13. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*. 2011 Nov 1;12:2825-30.
14. Bengio Y, Courville A, Vincent P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*. 2013 Mar 7;35(8):1798-828.
15. Aggarwal CC. *Data mining: the textbook*. New York: springer; 2015 Apr 13.
16. Khalid S, Khalil T, Nasreen S. A survey of feature selection and feature extraction techniques in machine learning. In *2014 science and information conference 2014 Aug 27* (pp. 372-378). IEEE.
17. Mumuni A, Mumuni F. Automated data processing and feature engineering for deep learning and big data applications: a survey. *Journal of Information and Intelligence*. 2025 Mar 1;3(2):113-53.
18. Overman T, Klabjan D. Federated automated feature engineering. *arXiv preprint arXiv:2412.04404*. 2024 Dec 5.
19. Hu K, Xiang L, Tang P, Qiu W. Feature Norm Regularized Federated Learning: Utilizing Data Disparities for Model Performance Gains. In *IJCAI 2024 Aug 3* (pp. 4136-4146).
20. Li H, Zhao Y, Mao Z, Qin Y, Xiao Z, Feng J, Gu Y, Ju W, Luo X, Zhang M. Graph neural networks in intelligent transportation systems: Advances, applications and trends. *arXiv preprint arXiv:2401.00713*. 2024 Jan 1.
21. Feurer M, Hutter F. Hyperparameter optimization. In *Automated machine learning: Methods, systems, challenges 2019 May 18* (pp. 3-33). Cham: Springer International Publishing.
22. Olson RS, Moore JH. TPOT: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning 2016 Dec 4* (pp. 66-74). PMLR.
23. Erickson N, Mueller J, Shirkov A, Zhang H, Larroy P, Li M, Smola A. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*. 2020 Mar 13.

24. McKinney W. Data structures for statistical computing in Python. *scipy*. 2010 Jun 28;445(1):51-6.
25. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M. {TensorFlow}: a system for {Large-Scale} machine learning. In 12th USENIX symposium on operating systems design and implementation (OSDI 16) 2016 (pp. 265-283).