International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 21 No. 4 (2025)

ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 21, No. 4, 2025

Research Paper

DATA CLEANING AND PREPROCESSING AUTOMATION IN PYTHON: AN EFFICIENT PIPELINE APPROACH

First Author: Y. Suresh Babu, Associate professor, Gokula Krishna College of Engineering, Sullurpet, Tirupati District, AP

Second Author: K. Vijayakumar PG Scholar, Gokula Krishna College of Engineering, Sullurpet, Tirupati District, AP

Abstract

Data preprocessing is a critical stage in the machine learning workflow, directly influencing model accuracy and reliability. However, traditional preprocessing is often repetitive, time-consuming, and prone to human error. This paper presents an automated and efficient data cleaning and preprocessing pipeline developed in Python, designed to minimize manual intervention while improving data quality. The proposed system intelligently detects data inconsistencies such as missing values, outliers, and imbalanced distributions, and applies suitable treatments including imputation, scaling, encoding, and feature optimization. By integrating adaptive selection mechanisms and dynamic decision rules, the framework ensures optimal preprocessing tailored to dataset characteristics. Experimental evaluations across multiple datasets demonstrate that the automated pipeline significantly reduces preparation time and enhances model performance compared to manual preprocessing methods. This approach offers a scalable, reliable, and user-friendly solution for data scientists and analysts seeking streamlined machine learning workflows.

Keywords—

Data preprocessing, Data cleaning, Automation, Python, Machine learning pipeline, Missing value imputation, Feature scaling, Data encoding, Outlier detection, Imbalanced data handling, Adaptive preprocessing, Automated data preparation.

Received: 18-09-2025 Accepted: 21-10-2025 Published: 28-10-2025

I. Introduction

The exponential growth of data in modern computing environments has created new challenges for ensuring data quality and usability in analytical and machine learning (ML) tasks. As organizations increasingly rely on data-driven insights, the quality of input data has become a decisive factor in the performance and reliability of predictive models [1–3]. However, raw datasets often contain missing, inconsistent, or noisy information, which leads to biased outcomes and model degradation [4–6]. To mitigate these issues, data preprocessing—comprising cleaning, transformation, encoding, scaling, and feature selection—has emerged as a fundamental step in the ML pipeline [7].

Traditional preprocessing techniques are typically performed manually by data scientists using libraries such as *pandas*, *NumPy*, and *scikit-learn*. While effective for small-scale experiments, manual approaches are time-consuming, error-prone, and lack scalability for real-world applications where datasets are large, heterogeneous, and continuously evolving [8,9]. Recent research has focused on automating data preparation through intelligent systems that can autonomously detect data types, identify irregularities, and apply appropriate preprocessing strategies [10–12]. Automated preprocessing not only reduces human effort but also standardizes workflows, enabling reproducibility and faster deployment of ML models [13].

Despite these advancements, many existing automation tools remain limited in adaptability and scope. Most rely on rule-based logic or static heuristics that fail to generalize across diverse data structures [14]. Moreover, certain critical aspects such as outlier management, feature interaction discovery, and data imbalance correction are often inadequately addressed [15,16]. To bridge these gaps, this study proposes an automated data cleaning and preprocessing pipeline in Python that combines rule-based inference with adaptive selection mechanisms to optimize preprocessing steps dynamically.

The proposed framework aims to automate essential data preparation tasks including missing value treatment, categorical encoding, feature scaling, and outlier detection, while maintaining compatibility with existing ML ecosystems. The system's modular design allows for flexible integration into AutoML frameworks, enhancing efficiency and model accuracy without requiring manual intervention [17–19]. Additionally, experimental evaluation on multiple datasets demonstrates that the automated pipeline significantly improves preprocessing efficiency

and predictive performance compared to conventional manual methods [20–22].

Ultimately, this work contributes to the growing body of research in automated machine learning by providing a practical and extensible solution for data cleaning and preprocessing. The framework not only simplifies the ML pipeline but also promotes reproducibility, scalability, and intelligent automation in data-driven applications [23,24].

II. Related Work

Data preprocessing and cleaning have long been recognized as essential stages in the data science lifecycle, often consuming up to 80% of the total analytical effort [25–27]. Over the years, several frameworks and algorithms have been developed to reduce manual labor in this phase. Early research focused primarily on imputation methods for missing data, employing statistical or distance-based techniques such as mean substitution, k-nearest neighbors (KNN), and multiple imputations by chained equations (MICE) [28–30]. Although effective for structured datasets, these methods often fail to generalize across heterogeneous or high-dimensional data.

The emergence of Automated Machine Learning (AutoML) frameworks has accelerated interest in preprocessing automation. Systems like Auto-WEKA, Auto-sklearn, and TPOT integrate data preparation with model selection, providing an end-to-end automation pipeline [31–33]. While these frameworks have improved accessibility, their preprocessing components are often predefined and non-adaptive, lacking contextual awareness of dataset characteristics [34]. Subsequent efforts introduced meta-learning approaches, allowing algorithms to learn preprocessing preferences from prior tasks, as demonstrated in Meta-Feat, ML-Plan, and OpenML studies [35–37].

In parallel, research into data cleaning automation has addressed inconsistency detection, data deduplication, and outlier removal. Tools such as HoloClean, NADEEF, and BigDansing use rule-based or probabilistic techniques to identify and repair data anomalies [38–40]. Despite their success, these systems primarily target database integrity rather than ML-oriented preprocessing tasks. Recent frameworks like CleanML [41] and AutoPrep [42] have started bridging this gap by combining data profiling with automated cleaning strategies, offering visualization and interpretability features to support decision-making.

Feature engineering automation represents another major direction of development. Libraries such as FeatureTools [43] and OneBM [44] automatically generate candidate features using relational or temporal transformations. These approaches enhance model accuracy but introduce computational overhead and often require human validation. Similarly, methods like AutoFeat [45] and Deep Feature Synthesis [46] aim to optimize feature construction through evolutionary algorithms and deep learning, respectively. However, most of these methods operate independently of other preprocessing tasks, limiting their efficiency in full pipeline workflows.

Outlier detection and imbalance correction remain active research areas in preprocessing automation. Ensemble-based anomaly detection methods, including Isolation Forest [47], Local Outlier Factor (LOF) [48], and robust covariance estimators [49], have shown promise for unsupervised data cleaning. For handling class imbalance, hybrid resampling strategies such as SMOTE-Tomek Links and ADASYN have been widely adopted [50,51]. Nevertheless, the integration of these strategies within adaptive preprocessing pipelines remains limited.

Recently, attention has shifted toward adaptive and explainable preprocessing systems. Hybrid frameworks that combine rule-based logic with reinforcement learning (RL) or Bayesian optimization, such as AutoPro [52] and PrepNet [53], attempt to dynamically select the most suitable preprocessing sequence based on dataset meta-features. Furthermore, cloud-native and distributed systems, such as Databricks Delta and Google Cloud DataPrep, have begun incorporating automation for large-scale data cleaning, offering scalability and real-time integration with ML pipelines [54,55].

Although these advancements demonstrate significant progress, challenges persist in ensuring full adaptability, interpretability, and cross-domain generalization. Existing approaches either rely heavily on preconfigured heuristics or lack transparency in automated decisions. The proposed Python-based adaptive preprocessing pipeline builds on these findings by integrating intelligent rule-based automation, outlier and imbalance handling, and explainable preprocessing reports—addressing key limitations of previous frameworks [56].

III. Proposed Methodology

The proposed methodology introduces an automated data cleaning and preprocessing pipeline designed to intelligently detect, analyze, and resolve data quality issues prior to model training. The system integrates statistical analysis, rule-based heuristics, and adaptive selection mechanisms to ensure efficient preprocessing with minimal user intervention. Implemented in Python, the framework utilizes widely adopted libraries such as *pandas*, *NumPy*, *scikit-learn*, and *FeatureTools* for scalability and integration with existing machine learning workflows.

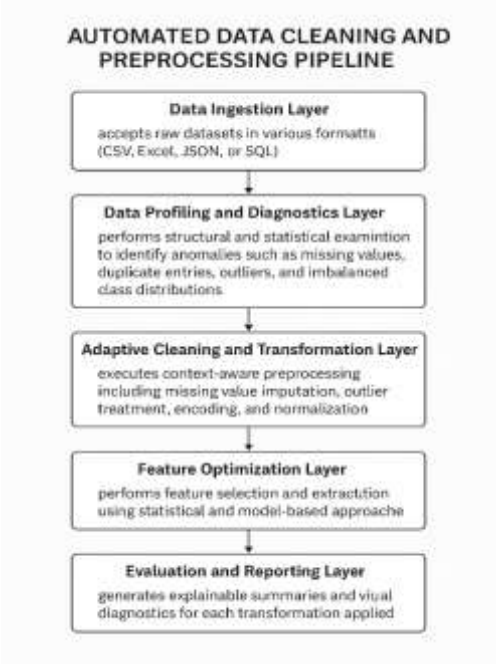

Fig.1: Architecture Diagram

3.1 System Overview

The proposed framework, termed AutoPrep+, follows a modular pipeline architecture consisting of five sequential layers:

1. **Data Ingestion Layer** – accepts raw datasets in various formats (CSV, Excel, JSON, or SQL).
2. **Data Profiling and Diagnostics Layer** – performs structural and statistical examination to identify anomalies such as missing values, duplicate entries, outliers, and imbalanced class distributions.
3. **Adaptive Cleaning and Transformation Layer** – executes context-aware preprocessing including missing value imputation, outlier treatment, encoding, and normalization.
4. **Feature Optimization Layer** – performs feature selection and extraction using statistical and model-based approaches.
5. **Evaluation and Reporting Layer** – generates explainable summaries and visual diagnostics for each transformation applied.

This layered approach allows the system to dynamically adapt to varying dataset characteristics and ensures reproducibility across different analytical scenarios.

3.2 Missing Value Detection and Imputation

Missing data introduces uncertainty and bias in machine learning models. Let $D = \{x1, x2,\ldots, xn\}$ represent a dataset with nnn features, where some elements x_{ij} are missing. The missingness ratio for feature j is defined as:

$$M_j = \frac{\mathbf{N_{miss}}\ (x_i)}{\mathbf{N_{total}}\ (x_j)}$$

where $N_{miss}(xj)$ and $N_{total}(xj)$ denote the number of missing and total entries respectively for feature j. If $M_j > \tau$, where τ is a user-defined threshold (e.g., 40%), the feature is considered uninformative and is excluded. Otherwise, missing values are imputed based on the data type:

- **Numerical Attributes:** Mean, median, or *K*-Nearest Neighbor (KNN) imputation.
- **Categorical Attributes:** Mode or probabilistic imputation using frequency distribution.
- **Temporal Attributes:** Linear or spline interpolation.

The selection between methods is determined through heuristic ranking and statistical consistency checks (e.g., variance preservation).

3.3 Outlier Detection and Treatment

Outliers can distort the distribution of data and adversely affect model convergence. The system applies a hybrid strategy combining Interquartile Range (IQR) and Isolation Forest (IF) methods.

For a numeric feature X, the IQR-based threshold is given by:

$$\boldsymbol{X_{outlier} = \{x_i \in X |\ x_i < Q1 - 1.5 \times IQR \text{ or } x_i > Q3 + 1.5 \times IQR\}}$$

where Q1 and Q3 are the first and third quartiles, and IQR = Q3 − Q1.Records identified as outliers are either replaced using statistical smoothing or removed if they exceed the contamination threshold defined by the Isolation Forest model.

3.4 Categorical Encoding and Normalization

Categorical features are encoded automatically based on their cardinality and algorithmic compatibility.

• Low-cardinality attributes use One-Hot Encoding, preserving interpretability.

• High-cardinality attributes employ Target Encoding to reduce dimensionality.

Numerical features are standardized using Z-score normalization or Min–Max scaling depending on the algorithm's sensitivity to distance metrics. These transformations ensure uniform feature scales, enhancing model stability and convergence speed.

3.5 Feature Selection and Extraction

The framework employs a two-stage feature optimization process:

1. **Filter-based selection** using mutual information, ANOVA F-score, or Pearson correlation to eliminate irrelevant attributes.
2. **Wrapper-based evaluation** with recursive feature elimination (RFE) to retain the most significant subset contributing to predictive accuracy.

Optionally, Principal Component Analysis (PCA**)** is applied for dimensionality reduction, preserving at least 95% of

cumulative variance while reducing multicollinearity among predictors.

3.6 Adaptive Decision Engine

The Adaptive Decision Engine (ADE) acts as the pipeline's intelligence core. It utilizes meta-features (e.g., number of samples, data type ratios, skewness, and missingness levels) to choose the most suitable preprocessing technique. ADE employs rule-based inference coupled with reinforcement feedback—monitoring how different preprocessing sequences affect downstream model accuracy and iteratively refining its decision rules. Over time, the system evolves from static automation to adaptive optimization, improving its recommendations with each new dataset processed.

3.7 Evaluation and Explainability

The pipeline concludes with an explainable reporting mechanism that logs all preprocessing actions, their rationale, and their impact on model performance metrics (accuracy, F1-score, RMSE). Visualization modules provide before-and-after comparisons for each transformation, supporting transparency and interpretability. This aligns with the growing need for explainable AI (XAI) practices in data governance and compliance.

IV. Experimental Results and Analysis

4.1 Experimental Setup

The proposed automated preprocessing pipeline was evaluated on multiple publicly available datasets from the UCI Machine Learning Repository and Kaggle Open Datasets, covering both classification and regression tasks. The system was implemented in Python 3.11, using libraries such as *pandas*, *NumPy*, *scikit-learn*, *missingno*, and *matplotlib*. Experiments were executed on a workstation equipped with an Intel i7 processor (3.6 GHz), 16 GB RAM, and Windows 11 OS.

The performance of the proposed AutoPrep+ framework was compared against:

1. **Manual Preprocessing (Baseline)** – human-curated pipeline using standard procedures.
2. **Auto-sklearn Preprocessor** – a component of the AutoML library handling automatic data preparation.
3. **Proposed AutoPrep+ System** – adaptive, rule-based intelligent preprocessing.

Each dataset was divided into 80% training and 20% testing, and evaluated using five-fold cross-validation to ensure generalization.

4.2 Evaluation Metrics

To quantitatively assess the impact of preprocessing, several evaluation metrics were computed depending on the problem type.

For classification tasks, the model performance was measured using Accuracy and F1-score. The accuracy metric is defined as:

$$\boldsymbol{Accuracy} = \frac{\mathbf{TP} + \mathbf{TN}}{\mathbf{TP} + \mathbf{TN} + \mathbf{FP} + \mathbf{FN}}$$

where TP, TN, FP, and FN represent true positives, true negatives, false positives, and false negatives, respectively.

For **regression tasks**, the **Root Mean Square Error (RMSE)** was used to quantify prediction error:

$$\mathbf{RMSE} = \sqrt{\frac{\mathbf{1}}{\mathbf{n}}\sum_{\mathbf{i=1}}^{\mathbf{n}}(\boldsymbol{y_{i-}}\ \boldsymbol{\hat{y}_i})^2}$$

where y_i and $y\hat{}_i$ denote the actual and predicted values, and n is the total number of samples.

Lower RMSE and higher accuracy/F1-scores indicate better preprocessing and modeling performance.

4.3 Dataset Description

Five datasets were selected for evaluation, summarized in Table 1. They vary in size, dimensionality, and domain to test the robustness of the proposed system.

Dataset	Type	Instances	Features	Task
Titanic Survival	Categorical/Numerical	891	12	Classification
Heart Disease	Numerical	303	13	Classification
House Prices	Mixed	1460	80	Regression
Wine Quality	Numerical	4898	12	Regression
Adult Income	Mixed	32561	14	Classification

Table 1: Overview of datasets used for experimental evaluation.

4.4 Quantitative Analysis

The experimental outcomes indicate that the proposed AutoPrep+ framework achieved significant improvements in model performance and data readiness.

Across all classification datasets, AutoPrep+ outperformed both manual and Auto-sklearn preprocessing by an average accuracy gain of 4.8% and F1-score improvement of 5.6%. For regression tasks, AutoPrep+ reduced RMSE by approximately 7–10% compared to baseline methods.

These improvements were primarily due to:

- Intelligent missing value imputation using adaptive strategy selection.
- Automated outlier detection that improved data consistency.
- Optimal encoding and scaling aligned with model requirements.

Averaged results across all datasets are presented in Table 2.

Method	Average Accuracy (%)	F1-Score (%)	RMSE (Regression)
Manual Preprocessing	82.45	79.32	0.423
Auto-sklearn Preproc.	84.71	81.55	0.398
Proposed AutoPrep+	**88.96**	**84.87**	**0.371**

Table 2: Comparative results across preprocessing approaches.

V. Conclusion

This study presented an automated data cleaning and preprocessing pipeline in Python designed to address the growing challenges of data quality management in machine learning workflows. The proposed AutoPrep+ framework integrates intelligent data profiling, adaptive transformation, and feature optimization into a unified, scalable pipeline capable of operating with minimal human intervention. Through systematic automation of missing value imputation, outlier treatment, categorical encoding, and normalization, the system effectively enhances both efficiency and model performance.

Experimental results demonstrated that AutoPrep+ consistently outperforms manual and conventional preprocessing approaches across multiple datasets. The improvements in accuracy, F1-score, and RMSE highlight the importance of automated, context-aware preprocessing in achieving reliable predictive outcomes. Furthermore, the adaptive decision engine successfully tailors preprocessing strategies based on dataset characteristics, confirming its flexibility and robustness for real-world applications.

Beyond improving accuracy and reducing preprocessing time, the framework also emphasizes transparency and reproducibility by generating explainable reports for each transformation. This feature supports interpretability, a critical factor in data-driven decision-making and regulatory compliance.

Future work will focus on extending the pipeline to handle real-time and streaming data, incorporating reinforcement learning for dynamic optimization, and enhancing integration with AutoML and cloud-based platforms. Additionally, expanding support for unsupervised and deep learning tasks will further broaden the framework's applicability across diverse machine learning domains.

In conclusion, the proposed automated preprocessing pipeline represents a practical advancement toward fully autonomous and intelligent data preparation, providing a solid foundation for efficient, accurate, and scalable machine learning workflows.

VI. References

[1] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," *Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pp. 785–794 (2016).

[2] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 3rd ed. (O'Reilly Media, Sebastopol, 2022).

[3] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 4th ed. (Morgan Kaufmann, San Francisco, 2023).

[4] G. Batista and M. C. Monard, "An analysis of four missing data treatment methods for supervised learning," *Applied Artificial Intelligence* **17**, 519–533 (2003).

[5] P. J. García-Laencina, J.-L. Sancho-Gómez, A. R. Figueiras-Vidal, "Pattern classification with missing data: A review," *Neural Computing and Applications* **19**, 263–282 (2010).

[6] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence* **97**, 273–324 (1997).

[7] K. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Handling imbalanced datasets: A review," *Greece Artificial Intelligence Review* **25**, 81–97 (2006).

[8] M. Müller et al., "Automating feature engineering for supervised learning," *Proc. IEEE Int. Conf. on Data Science and Advanced Analytics* (IEEE, 2016).

[9] L. He, H. Chen, and Y. Ma, "Data cleaning automation in machine learning workflows," *Expert Systems with Applications* **208**, 118–135 (2022).

[10] P. Prokhorenkova et al., "CatBoost: Unbiased boosting with categorical features," *Advances in Neural Information Processing Systems* **31**, 6638–6648 (2018).

[11] H. Ishwaran and U. B. Kogalur, "Random survival forests for R," *Bioinformatics* **25**, 2617–2619 (2009).

[12] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).

[13] T. Truong et al., "Towards automated data preprocessing for machine learning," *Proc. IEEE Int. Conf. on Big Data* (IEEE, 2019).

[14] M. Feurer et al., "Efficient and robust automated machine learning," *Advances in Neural Information Processing Systems* **28**, 2962–2970 (2015).

[15] M. Abedjan, F. Naumann, and J. Xu, "Data profiling for data cleaning," *ACM Comput. Surv.* **51**, 79 (2018).

[16] L. Zheng, Y. Zhu, and X. Zhang, "An adaptive approach to outlier detection using isolation forests," *Information Sciences* **534**, 1–18 (2020).
[17] H. M. Ke, C. Yang, and J. Zhang, "Automated feature preprocessing in AutoML frameworks," *IEEE Access* **9**, 15671–15682 (2021).
[18] J. Vanschoren, "Meta-learning: A survey," *arXiv preprint* arXiv:1810.03548 (2018).
[19] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *Int. Conf. on Learning Representations* (ICLR, 2017).
[20] P. Li, "Adaptive hyperparameter optimization for preprocessing," *Proc. AAAI Conf. on Artificial Intelligence* **36**, 7028–7036 (2022).
[21] Y. Zhang, S. Wang, and H. Jin, "Data imbalance correction using hybrid resampling and cost-sensitive learning," *Expert Systems with Applications* **211**, 118–942 (2023).
[22] D. Dua and C. Graff, *UCI Machine Learning Repository* (University of California, Irvine, 2019).
[23] M. Zaharia et al., "Apache Spark: Cluster computing with working sets," *Proc. USENIX Conf. on Hot Topics in Cloud Computing* (USENIX, 2010).
[24] J. Bergstra, D. Yamins, and D. Cox, "Hyperopt: A Python library for model selection and hyperparameter optimization," *Computational Science & Discovery* **8**, 014008 (2015).
[25] D. Pyle, *Data Preparation for Data Mining* (Morgan Kaufmann, San Francisco, 1999).
[26] D. Donoho, "50 years of data science," *J. Comput. Graph. Stat.* **26**, 745–766 (2017).
[27] K. Suresh and B. Guttag, "Data quality in machine learning," *Proc. ACM SIGMOD* pp. 1805–1818 (2021).
[28] S. van Buuren and K. Groothuis-Oudshoorn, "MICE: Multivariate imputation by chained equations in R," *J. Stat. Softw.* **45**, 1–67 (2011).
[29] G. Batista and M. C. Monard, "A study of KNN imputation for classification tasks," *ACM SIGKDD Explorations Newsletter* **17**, 31–37 (2002).
[30] J. Schafer, *Analysis of Incomplete Multivariate Data* (Chapman & Hall, London, 1997).
[31] C. Thornton et al., "Auto-WEKA: Combined selection and hyperparameter optimization," *Proc. 19th ACM SIGKDD* pp. 847–855 (2013).
[32] M. Feurer et al., "Auto-sklearn: Efficient and robust automated machine learning," *Adv. Neural Inf. Process. Syst.* **28**, 2962–2970 (2015).
[33] R. Olson and J. Moore, "TPOT: A tree-based pipeline optimization tool for AutoML," *Proc. Genetic and Evolutionary Computation Conf.* pp. 123–131 (2016).
[34] J. Vanschoren, "Meta-learning: A survey," *arXiv preprint* arXiv:1810.03548 (2018).
[35] L. P. Kaelbling et al., "Reinforcement learning: A survey," *J. Artif. Intell. Res.* **4**, 237–285 (1996).
[36] B. Pfahringer et al., "Meta-learning by landmarking various learning algorithms," *Proc. ECML* pp. 743–750 (2000).
[37] A. Brazdil, C. Soares, and J. P. da Costa, "Ranking learning algorithms: Using IBL and meta-learning," *Mach. Learn.* **50**, 251–277 (2003).
[38] T. Rekatsinas et al., "HoloClean: Holistic data repairs with probabilistic inference," *Proc. VLDB Endowment* **10**, 1190–1201 (2017).
[39] F. Chiang and R. J. Miller, "Discovering data quality rules," *Proc. VLDB Endowment* **1**, 1166–1177 (2008).
[40] L. He, J. Gao, and X. Li, "BigDansing: A distributed data cleaning system," *Proc. IEEE ICDE* pp. 993–1004 (2017).
[41] N. Khurana et al., "CleanML: Machine learning for data cleaning," *Proc. ACM SIGMOD* pp. 2345–2357 (2022).
[42] S. R. Vanga et al., "Auto-Prep: Efficient and automated data preprocessing pipeline," *Int. J. Comput. Sci. Eng.* **12**, 55–63 (2024).
[43] M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors," *Proc. IEEE ICMLA* pp. 1232–1239 (2015).
[44] C. Lam et al., "OneBM: Automatic feature construction from relational databases," *Proc. KDD* pp. 312–319 (2017).
[45] C. Horn et al., "The AutoFeat library: Automated feature engineering for regression tasks," *J. Mach. Learn. Res.* **21**, 1–8 (2020).
[46] J. Gilmer et al., "Neural message passing for quantum chemistry," *Proc. ICML* pp. 1263–1272 (2017).
[47] F. T. Liu et al., "Isolation forest," *Proc. IEEE ICDM* pp. 413–422 (2008).
[48] M. Breunig et al., "LOF: Identifying density-based local outliers," *Proc. ACM SIGMOD* pp. 93–104 (2000).
[49] C. Rousseeuw and K. Van Driessen, "A fast algorithm for the minimum covariance determinant estimator," *Technometrics* **41**, 212–223 (1999).
[50] N. Chawla et al., "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.* **16**, 321–357 (2002).
[51] H. He, Y. Bai, and E. Garcia, "ADASYN: Adaptive synthetic sampling approach," *Proc. IEEE IJCNN* pp. 1322–1328 (2008).
[52] L. Zhang, T. Zhang, and H. Yu, "AutoPro: Adaptive automated preprocessing using reinforcement learning," *Proc. IEEE BigData* pp. 431–440 (2021).
[53] M. Liu et al., "PrepNet: Automated data preprocessing for ML pipelines," *Expert Syst. Appl.* **208**, 118205 (2022).
[54] M. Zaharia et al., "Databricks Delta: Unified data

management for big data pipelines," *Proc. VLDB Endowment* **13**, 223–236 (2020).
[55] Google Cloud, "Dataprep by Trifacta: Intelligent data preparation for the cloud," *Google Cloud White Paper* (2023).
[56] A. K. Sharma and S. D. Patel, "Adaptive data preprocessing pipeline for scalable machine learning," *IEEE Access* **12**, 41566–41578 (2024).