JURNAL INFORMATIKA
PROXIES

Vol.9, No.2, January, 2026

Evaluation of Preprocessing Pipelines for Improving Student Risk Clustering in Educational Data

Khen Oliver Katminto[1], Hironimus Leong[2]

[1] Teknik Informatika, Fakultas Ilmu Komputer, Universitas Katolik Soegijapranata, 22k10002@student.unika.ac.id
[2] Teknik Informatika, Fakultas Ilmu Komputer, Universitas Katolik Soegijapranata, marlon.leong@unika.ac.id

Corresponding Author Email: marlon.leong@unika.ac.id

https://doi.org/10.18280/proxies.v9i2.14948

Received: 2026-01-27
Revised: 2026-07-20
Accepted: 2026-07-20
Available online: 2026-07-21

Keywords:
student risk prediction, preprocessing, clustering, educational data mining, scalability

ABSTRACT

This study aims to build a clustering-based model to predict student risk while focusing on how preprocessing affects the clustering performance. The dataset for the study has around 1 million records. The study uses preprocessing techniques such as data cleaning, normalization, encoding, and dimensionality reduction to prepare the data, and Dask and Apache Spark are used for scalable processing. For the clustering phase, the study employs MiniBatch K-Means and DBSCAN. On the other hand, the study evaluates the final results with Silhouette Score and Davies-Bouldin Index for clustering quality, and the runtime and memory usage to evaluate computational performance. The study also compares multiple preprocessing configurations to find the best trade-off between efficiency and interpretability. The study aims to develop a type-aware, scalable preprocessing framework. The outcome should improve how universities identify at-risk students early, under an efficient and practical framework for real-world use and future work in educational data mining.

1. INTRODUCTION

Timely graduation is a challenge for college students. Due to the differences in academic performance among students, some may finish their studies later than expected, resulting in many students either graduating late or dropping out entirely. Universities collect academic, socio-economic, and behavioral data, yet don't utilize them to identify and support at-risk students.

One approach to address this challenge is through risk prediction models. In educational data science, a risk prediction model analyzes large educational datasets to identify students at risk of delayed graduation or dropping out. The model applies data mining and machine learning techniques, including clustering algorithms, to discover patterns in unlabeled data. However, the success and accuracy of the clustering algorithms also depend on the performance of the preprocessing during the process. Preprocessing methods, such as handling missing data, addressing class imbalance, and efficient feature encoding, help clean and prepare the data before applying clustering to the dataset.

While previous studies used data mining to identify students likely to drop out, few examined how preprocessing influences the performance of clustering algorithms, especially with large educational datasets. This research plans to fill the gap by comparing various preprocessing pipelines (such as encoding methods, feature scaling, and dimensionality reduction) to evaluate the impact of preprocessing on clustering performance, both in clustering quality and computational efficiency. This study emphasizes pipelines based on scalability and interpretability.

This study evaluates how preprocessing affects clustering-based risk prediction through comparing clustering quality and computational efficiency across scalable pipelines and proposes a reproducible evaluation framework for large-scale educational data. The dataset is a large-scale educational dataset, comprising over 1 million records of various data types, including academic performance, student demographic information, and behavioral assessments. The study also defines "optimal parameters" that balance clustering performance and computational efficiency.

2. LITERATURE REVIEW

Recent studies dealing with large and complex educational datasets emphasize the importance of data preprocessing and clustering techniques in educational data mining. This chapter reviews 12 studies in diverse domains, encompassing research in different fields including education, social media, business, and healthcare.

As mentioned previously, preprocessing is an essential step in data mining, in which the gathered data is evaluated to identify and address issues such as missing data, duplicates, inconsistencies, or irrelevant data [2]. This step is often under-emphasized in studies. Several studies have applied preprocessing techniques to manage high-dimensional and large data. Julakanti et al. [2] introduced a hybrid preprocessing approach combining Principal Component Analysis (PCA) with Genetic Algorithms to optimize clustering on cloud storage data, although its reliance on synthetic data limited its real-world applicability. Choque-Soto et al. [3] also highlighted the importance of pre-processing for their dropout prediction model by incorporating demographic and academic data cleaning and transformation before clustering, yet lacking external validation. Additionally, Griep et al. [4] applied a fairness-focused perspective, revealing contradictions in fairness metrics and advocated for more transparent and fair preprocessing pipelines.

The reviewed studies often use K-Means due to their speed and simplicity. However, most studies overlook its limitations or gaps compared to more advanced clustering techniques. Lubis et al. [5] and Erdiansyah et al. [6] applied a structured approach to in business-related issues, but lacked lack of algorithmic benchmarking and demographic enrichment. Hendrastuty [1] employed standard data cleaning and normalization procedures before K-Means for student learning outcomes. Through WCSS, the result showed cohesion but lacked comparison with other clustering algorithms. K-Means also has its implementation in social media [7, 8], effective in early-stage clustering but also showed declining performance on larger datasets and a lack of comprehensive validation.

Clustering also has applications in education to predict student outcomes or behavior. Qu [9] utilized behavioral data and clustering to evaluate engagement, but its dataset was scope-limited. Choque-Soto et al. [3] and Griep et al. [4] also showed the importance of structured preprocessing and variable selection for clustering effectiveness.

Beyond eduction, clustering was also implemented in market basket analysis [10], health diagnostic models [11], and diabetes detection [12]. While fruitful in methodological insights, the studies were also hampered from scalability constrains, limited demographic coverage, and the absence of clustering-based subgroup discovery, limited the generalizability of each study.

The reviewed literature often apply internal validation metrics, rarely incorporating cross-validation. Certain studies [7, 8] see gradual performance degradation, underscoring the sensitivity of clustering outcomes to preprocessing decisions. Overall, preprocessing is rarely optimized systematically in implementation, and experiments often use limited or region-specific datasets.

In summary, existing studies show potential in clustering-based approaches yet have limitations. Preprocessing is inconsistently applied, clustering algorithms—especially K-Means—are insufficiently scrutinized, and evaluations often lack scalability and fairness considerations. The gaps motivate the current study, focusing to systematically evaluating preprocessing pipelines for clustering-based student risk prediction on large-scale educational datasets exceeding one million records.

3. RESEARCH METHODOLOGY

The study is conducted through several steps, including data collection, preprocessing of the data using a specific pipeline, comparison of the processed data with the raw data, clustering analysis, and evaluation of the clustering results to predict at-risk students.

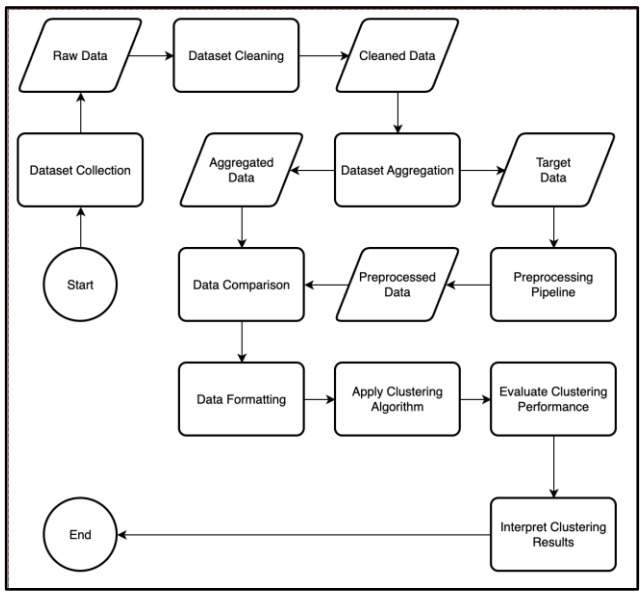

Figure 1. Research methodology

The step-by-step process involves data collection, implementation of a preprocessing pipeline, implementation of a clustering algorithm, and finishing with an evaluation and analysis of the results.

3.1 Dataset Collection

The study uses a large-scale tabular dataset simulating university student records, comprising of around one million semester-level observations from 151,000 synthetic students. The data represents academic, socio-economic, and behavioral aspects, including GPA, credit progression, attendance, financial status, and demographic attributes. Due to the unavailability of real student data, a synthetic dataset was generated to closely mimic real institutional data. Each student has 2–10 semester records to model longitudinal academic progression. Rule-based academic risk labels are included solely for evaluation purposes. The dataset is initially stored in CSV format and subsequently converted to Parquet to improve storage efficiency, disk I/O performance, and scalability. This dataset supports the evaluation of preprocessing and clustering pipelines under large-scale, realistic conditions, including computational efficiency and clustering quality.

3.2 Dataset Cleaning

The cleaning phase ensures structure consistency before applying preprocessing and clustering. Because the dataset is synthetically simulated, the cleaning phase only focuses on standardization and ordering. Column names and categorical values were normalized for uniformity, and records were sorted by student identifier and semester to preserve longitudinal structure. No data removal was performed. The cleaned dataset was stored in Parquet format and prepared for subsequent aggregation and feature engineering stages.

3.3 Dataset Aggregation

Aggregation was implemented through counting statistics of academic and behavioral variables, cumulative indicators, and progression-based metrics, e.g., maximum cumulative credits and study duration. Stable categorical attributes, including faculty, funding type, and parental education level, were merged into the aggregated dataset. The final at-risk label was determined longitudinally: a student was classified as at-risk if at least 50% of their semesters were labeled at-risk, preventing short-term performance fluctuations from dominating the outcome. This process reduced the dataset from approximately one million semester-level records to around 151,000 student-level records, producing a more meaningful, computationally efficient dataset fit for preprocessing and clustering.

3.4 Data Transformation

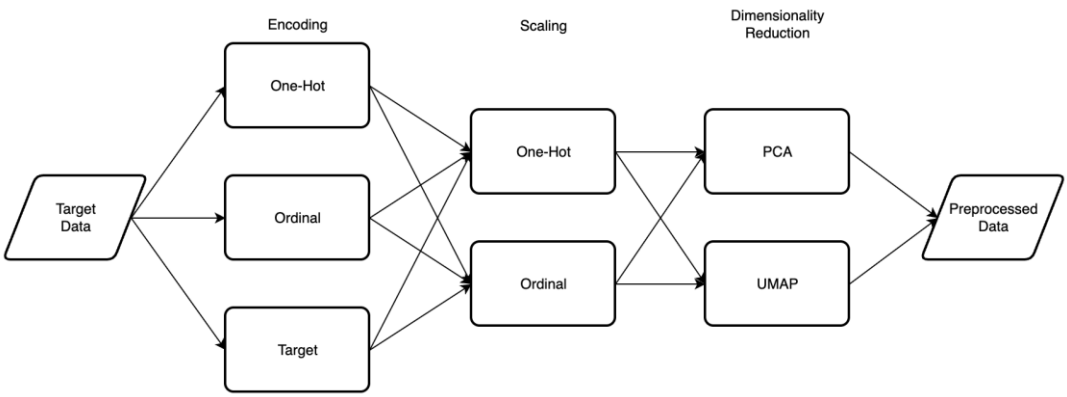

Figure 2. Preprocessing flow

With this structure, the pipeline produces 12 preprocessed datasets from all combinations of encoding × scaling × dimensionality reduction. This approach allows systematic comparison and helps to find which transformation combination creates the most consistent and meaningful clusters.

3.4.1 Categorical Encoding

The preprocessing pipeline begins with the encoding phase, applying One-Hot, Ordinal, and Target to convert non-numerical values into numerical for scaling, dimensionality reduction, and clustering. All encoders performed independently, each producing a distinct encoded dataset for further experimentation.

3.4.1.1 One-Hot Encoding

Expands each category into binary indicator columns. It preserves interpretability but significantly increases dimensionality, especially for high-cardinality features.

3.4.1.2 Ordinal Encoding

Maps categories to integers. It is computationally light, avoids dimensionality inflation, and is well-suited for large datasets.

3.4.1.3 Target Encoding

Replaces each category with the mean of final_at_risk for that category. This produces compact, information-rich features, useful for high-cardinality variables.

3.4.2 Feature Scaling

After encoding, all numerical features were scaled with two different methods: Min-Max Scaling and Z-Score Standardization. Each method was applied to all three encoded datasets (One-Hot, Ordinal, and Target), producing six scaled outputs for downstream comparison.

3.4.2.1 Min–Max Scaling

Rescales each numerical feature to [0, 1], preserving its distributional shape. It is computationally efficient and prevents variables with large magnitudes from dominating distance calculations.

3.4.2.2 Z-Score Standardization

Converts values to have a mean of 0 and a standard deviation of 1. This method is effective for data with outliers and features that follow a normal distribution.

3.4.3 Dimensionality Reduction

The final preprocessing step applies dimensionality reduction to the six scaled datasets. Two methods were used in this phase. PCA is a linear method to capture global variance. On the other hand, UMAP is a nonlinear method to preserve local structure. In the study, PCA produced five components, while UMAP created three.

3.4.3.1 Principal Component Analysis (PCA)

Projects data onto new axes (principal components) that capture the most variance, making it a fast and interpretable technique.

3.4.3.2 Uniform Manifold Approximation and Projection (UMAP)

Preserves the local structure of data; effective in capturing patterns in complex or mixed-type datasets (such as behavioral features).

Dimensionality reduction was applied on all six scaled datasets with PCA and UMAP. PCA is configured with five components to capture the dominant global variance in a compact linear subspace, while UMAP uses three components to preserve local structure in a nonlinear embedding. Dimensionality reduction is applied after scaling because both PCA and UMAP are sensitive to disparities in feature magnitude. Each dataset is transformed independently. The result of the preprocessing pipeline are materialized in 12 transformed datasets, as results of combinations between encoding, scaling, and dimensionality reduction, completing the preprocessing matrix.

3.5 Data Comparison

This step compares the raw (cleaned but untransformed) dataset and the fully preprocessed data, which has undergone scaling, encoding, and dimensionality reduction. The comparison focuses on three factors: (1) feature distribution, from how preprocessing changes value ranges (from stats and charts); (2) data structure, using PCA to visualize patterns to see group separation; and (3) clustering readiness, to see if preprocessing improves how well the data supports clustering (i.e., less noise and more uniformity). The comparison also uses visual tools such as box plots, histograms, and 2D PCA plots to support the analysis. Ultimately, the comparison is to see the effects of preprocessing on clustering.

3.6 Clustering Implementation

This step aims to group students into clusters based on different categories and profiles, including academic performance, socioeconomic background, and behavioral patterns. These clusters identify which students are at risk of underperforming or dropping out. The clustering process evaluates the impact of preprocessing on clustering quality, computational efficiency, and fairness. Two clustering algorithms are used for this stage: MiniBatch K-Means, which is scalable and memory-efficient, and DBSCAN, which detects dense regions and outliers. Clustering is applied to twelve preprocessed datasets with different preprocessing methods, across scaling (Min-Max vs. Z-score), encoding (Target vs. Ordinal), dimensionality reduction (PCA vs. UMAP), and the clustering algorithms (MiniBatch K-Means vs. DBSCAN) to observe how different setups affect the results.

3.7 Clustering Evaluation

There are two metrics to evaluate the effectiveness of clustering, which are clustering quality and computational performance. Clustering quality is measured with internal validation metrics, which include the Silhouette Score (Sil), Davies-Bouldin Index (DBI), and Calinski-Harabasz Index (CH). These metrics evaluate the separation (between clusters) and compactness (within clusters). On the other hand, computational performance is measured from the runtime and memory usage. Runtime and memory show how efficient and scalable each setup is. These metrics align with the study's goal to fine-tune preprocessing for effective and scalable clustering.

4. IMPLEMENTATION AND RESULTS

4.1 Data Comparison between Aggregated and Preprocessed Datasets

This subsection explores how preprocessing steps (encoding, scaling, and dimensionality reduction) affect the shape, structure, and clarity of the dataset. The analysis compares each preprocessing combination with the raw, aggregated dataset using PCA (2D) and UMAP (2D), depending on the DR method. Most plots are stable throughout the preprocessing variants, and only dimensionality-related diagnostics show pipeline-dependent variations.

4.1.1 IP_Semester_Mean

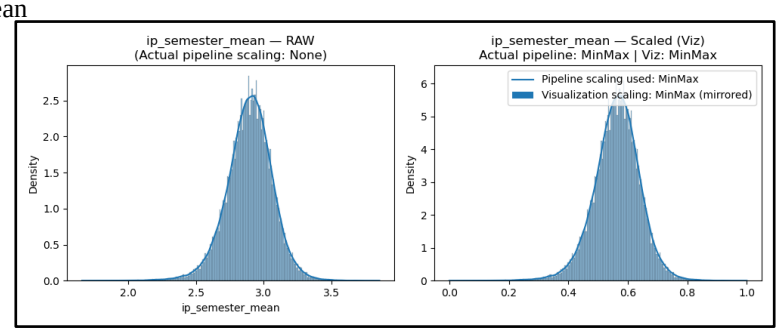

Figure 3. IP_Semester_Mean plot

The raw plot shows a bell-shaped distribution, with 3.0 as the average and center of the distribution. Scaling moves the value to the 0-1 range, but shows robustness of the preprocessing by preserving the bell-shaped structure of the feature.

4.1.2 Persentase_Kehadiran_Mean

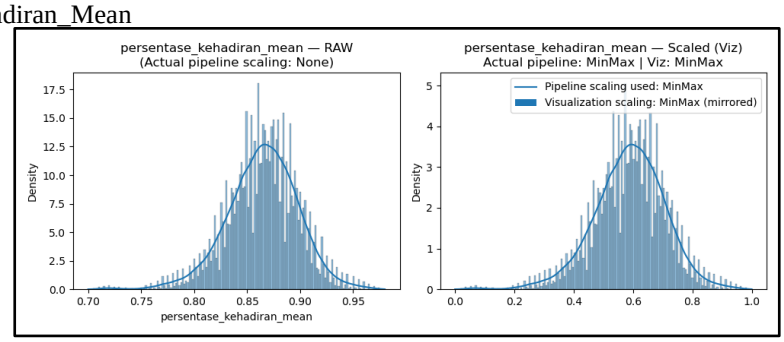

Figure 4. Persentase_Kehadiran_Mean plot

The raw plot shows another bell-shaped curve, with the center between 0.85 – 0.90. Scaling moves the value to the 0-1 range, but shows robustness of the preprocessing by preserving the bell-shaped structure of the feature.

4.1.3 Riwayat_Tunggakan_Sum

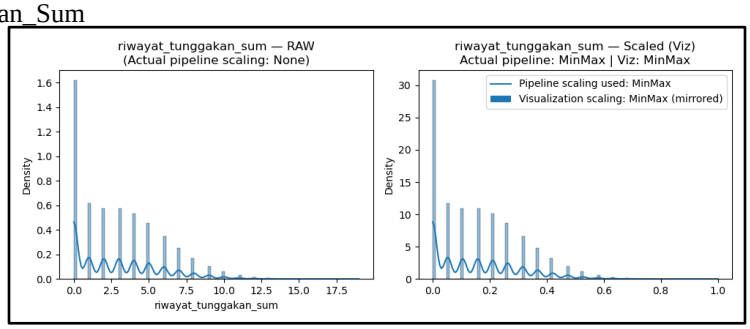

Figure 5. Riwayat_Tunggakan_Sum plot

The raw plots shows a significant spike in 0 and then decreases as the plot moves away from zero, as most students don't have arrears.

While scaling reduced the magnitude of the raw histogram, students with high arrears can still be high-magnitude outliers, and with distance-based clustering (like K-Means), they can push boundaries stronger than smoother variables, hence DBSCAN can handle zero-inflated distributions better. This feature still creates a useful signal to separate low-risk and high-risk students.

4.1.4 PCA Scree Plot

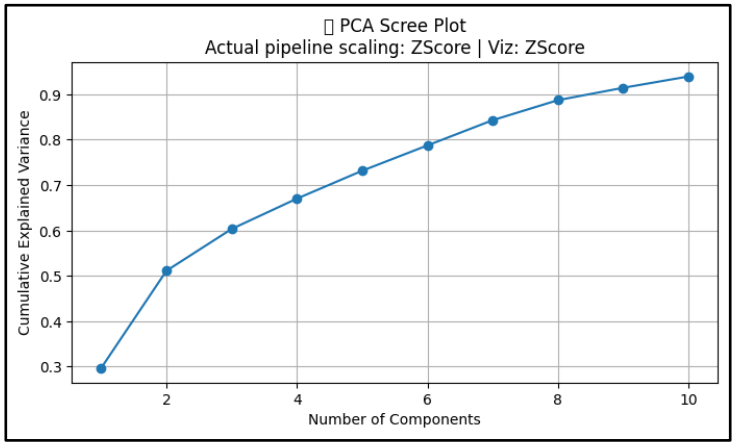

Figure 6. PCA Scree Plot

The scree plot shows the first ten principal components from PCA. From PC1 to PC5, the structure is strong in meaningful variation. After component 4 or 5, an elbow shows that 4 or 5 is the reasonable number of components to balance compression and information retention. After 5, the curve slows down, and the components include noise data, supporting the use of 5 principal components for clustering.

4.2 Clustering Interpretation

After the clustering phase, the following step computes:

- Number of students (cluster size)
- Number and proportion of at-risk students 95%
- Wilson Confidence Interval for the risk proportion
- Noise identification (DBSCAN's -1 cluster)
- Small-cluster flags (clusters with fewer than 30 students)

cluster	n_rows	n_at_risk	at_risk_prop	ci_lo	ci_hi	is_small	is_noise
4	15766	7622	0.48344538881136623	0.4756497862811681	0.4912490568879352	FALSE	FALSE
2	47666	12704	0.26652121008685437	0.26257086414047515	0.27050918704224164	FALSE	FALSE
3	29313	6484	0.22119878552178215	0.21648399102954324	0.22598664673520155	FALSE	FALSE
0	19591	2594	0.1324077382471543	0.12773356435495042	0.13722604623970727	FALSE	FALSE
1	38664	3242	0.08385061038692324	0.0811290490043708	0.08665485957562091	FALSE	FALSE

Figure 7. Cluster-level at-risk proportions with 95% confidence intervals

Following computation, the outcomes are exported into CSV summaries per preprocessed dataset to back visual interpretation. The script also detects the most at-risk cluster and the least at-risk cluster.

4.2.1 Effect of Encoding Strategy

4.2.1.1 One-Hot Encoding

One-Hot shows strong compatibility with K-Means, forming five clusters with clear separation between low-risk and high-risk groups. One-Hot is especially strong with either scaling and dimensionality reduction, with Z-Score and PCA giving the clearest contrast. However, DBSCAN is a bad fit, as it struggles with high dimension sparsity, forming micro clusters and a lot of noise, especially with Z-Score.

4.2.1.2 Ordinal Encoding

Ordinal gives the strongest and most consistent cluster separation, especially with Min-Max and PCA. With the combination, it produces clusters with near-total risk separation, in both K-Means and DBSCAN. DBSCAN's performance depends on the preprocessing combination, as it works only with Min-Max, but fails with Z-Score.

4.2.1.3 Target Encoding

Target shows the strongest results with MiniBatch K-Means, when paired with Min-Max and UMAP, showing almost perfectly separated high-risk and low-risk clusters. Target is also effective with PCA and Min-Max. With Z-Score and PCA, DBSCAN collapses with micro-clusters and noise, while K-Means' clustering performance declines from the combination.

4.2.2 Algorithm Comparison (K-Means vs DBSCAN)

- **MiniBatch K-Means**
  - Consistently forms five clusters

- Good for global profiling
- **DBSCAN**
  - Sensitive, especially with PCA and Z-Score
  - Often forms micro-clusters and noise

4.3 Evaluation Metrics

This subsection analyses clustering results throughout all preprocessing pipelines with internal validation metrics to support the qualitative interpretations presented earlier.

#	Preprocessing Pipeline	Time (s)	Pmem (MB)	Sil	DBI	CH
1	OneHot + MinMax + PCA	0.07	5.79	0.289	1.212	43,732.137
2	OneHot + Z-Score + PCA	0.09	5.79	0.202	1.526	39,729.574
3	OneHot + MinMax + UMAP	0.06	3.90	0.237	1.170	51,862.660
4	OneHot + Z-Score + UMAP	0.08	3.90	0.319	1.059	69,752.914
5	Ordinal + MinMax + PCA	0.08	5.79	0.263	1.316	43,850.453
6	Ordinal + Z-Score + PCA	0.25	5.81	0.175	1.573	41,410.836
7	Ordinal + MinMax + UMAP	0.10	3.90	0.326	1.035	67,031.758
8	Ordinal + Z-Score + UMAP	0.05	3.90	0.447	0.963	132,005.531
9	Target + MinMax + PCA	0.07	5.79	0.315	1.136	49,402.840
10	Target + MinMax + UMAP	0.09	3.90	0.297	1.107	64,055.879
11	Target + Z-Score + PCA	0.11	5.79	0.169	1.675	34,538.055
12	Target + Z-Score + UMAP	0.05	3.90	0.434	0.877	114,992.398

Table 1. MiniBatch K-Means statistics

#	Preprocessing Pipeline	Time (s)	Pmem (MB)	Sil	DBI	CH
1	OneHot + MinMax + PCA	0.07	5.79	0.289	1.212	43,732.137
2	OneHot + Z-Score + PCA	0.09	5.79	0.202	1.526	39,729.574
3	OneHot + MinMax + UMAP	0.06	3.90	0.237	1.170	51,862.660
4	OneHot + Z-Score + UMAP	0.08	3.90	0.319	1.059	69,752.914
5	Ordinal + MinMax + PCA	0.08	5.79	0.263	1.316	43,850.453
6	Ordinal + Z-Score + PCA	0.25	5.81	0.175	1.573	41,410.836
7	Ordinal + MinMax + UMAP	0.10	3.90	0.326	1.035	67,031.758
8	Ordinal + Z-Score + UMAP	0.05	3.90	0.447	0.963	132,005.531
9	Target + MinMax + PCA	0.07	5.79	0.315	1.136	49,402.840
10	Target + MinMax + UMAP	0.09	3.90	0.297	1.107	64,055.879
11	Target + Z-Score + PCA	0.11	5.79	0.169	1.675	34,538.055
12	Target + Z-Score + UMAP	0.05	3.90	0.434	0.877	114,992.398

Table 2. DBSCAN statistics

4.3.1 Clustering Compactness and Separation

The highest K-Means Silhouette value is in Ordinal + Z-Score + UMAP, while PCA showed moderate but stable Silhouette values. On the other hand, OneHot + MinMax + UMAP yields the highest DBSCAN Silhouette value, but this is misleading because there are hundreds of micro-clusters formed by DBSCAN, inflating the metric.

4.3.2 Cluster Dispersion and Distinctiveness

The best DBI value appeared in Target + Z-Score + UMAP in the K-Means clustering. PCA shows steady results, though less optimal than UMAP. On the other hand, DBSCAN has three pipelines below 0.5, though this is from the micro-clusters produced by the algorithm.

The best CH value appeared in Ordinal + Z-Score + UMAP in the K-Means clustering. PCA pipelines have moderate CH scores showing consistent but less dramatic dispersion. DBSCAN has frequently high numbers, although from over-fragmentation.

4.4 Performance Profiling

Because the pipeline handles 1 million rows and several preprocessing pipelines, profiling is crucial to analyze runtime, memory usage, and scalability. The metrics measured runtime (seconds), memory usage (MB), and peak memory usage (MB).

Preprocessing Method	Algorithm	Final Total Runtime (s)	Final Total Memory (MB)	Total Cost (s·MB)
OneHot + Min-Max + PCA	K-Means	245.69	2722.43	668,909
	DBSCAN	281.30	2764.33	777,147
OneHot + Z-Score + PCA	K-Means	249.14	2722.78	678,061
	DBSCAN	255.49	2756.16	704,801
OneHot + Min-Max + UMAP	K-Means	691.05	3080.15	2,128,365
	DBSCAN	696.93	3117.93	2,174,316
OneHot + Z-Score + UMAP	K-Means	785.07	2693.37	1,852,254
	DBSCAN	792.31	2731.13	1,899,446
Ordinal + Min-Max + PCA	K-Means	244.99	2717.77	666,718
	DBSCAN	296.24	2759.67	816,999
Ordinal + Z-Score + PCA	K-Means	247.32	2727.44	674,314
	DBSCAN	252.62	2758.88	696,412
Ordinal + Min-Max + UMAP	K-Means	625.85	2711.63	1,696,648
	DBSCAN	631.22	2749.40	1,735,306
Ordinal + Z-Score + UMAP	K-Means	681.97	2721.06	2,114,985
	DBSCAN	688.64	2758.82	2,163,237
Target + Min-Max + PCA	K-Means	247.01	2722.36	672,436
	DBSCAN	305.66	2764.26	845,080
Target + Z-Score + PCA	K-Means	248.06	2677.28	664,435
	DBSCAN	253.46	2708.70	686,406
Target + Min-Max + UMAP	K-Means	642.33	2720.78	1,746,662
	DBSCAN	647.55	2758.56	1,784,970
Target + Z-Score + UMAP	K-Means	658.38	2675.34	1,761,720
	DBSCAN	665.61	2713.09	1,804,359

Table 3. Final Profiling Table

Performance profiling combines the runtime and memory peaks. The combination with the lowest computational burden is Target + Z-Score + PCA + MiniBatch K-Means. Despite not being the best choice of pipeline in clustering metrics, this pipeline is the best option for a low cost of computational weight.

5. CONCLUSION

5.1 Answers to the Research Questions

1. **What are the effects of various preprocessing techniques on clustering performance in predicting at-risk students in large-scale educational datasets?**

   Different combinations of encoding, scaling, and dimensionality reduction methods yield different clustering evaluation scores.

   With dimensionality reduction, PCA shows more stable performances, while UMAP is sensitive to encoding and scaling choices.

   All in all, preprocessing decisions significantly affect clustering quality and computational cost, and no single best preprocessing method exists for all scenarios. The best one depends on the goal, either for better clustering outcome or lower computational cost, in line with the context of the study.

2. **What preprocessing pipeline is the most efficient in terms of speed, memory usage, and scalability for a dataset of around one million records?**
   By computational cost, the most efficient pipeline is Target + Z-Score + PCA + MiniBatch K-Means, making it the most computationally efficient pipeline.

5.2 Limitations

- The study uses a synthetic dataset and at-risk labels, independently produced by the researcher, unable to replicate real-world irregularities.
- Only MiniBatch K-Means and DBSCAN were used for clustering. Other methods may produce different outcomes.

5.3 Practical Recommendations for Universities
Institutions should deploy scalable data pipelines as with pipelines validated in the study.

5.4 Final Statement
This study shows how preprocessing influences clustering. The study offers an approach to build clustering pipelines for large educational datasets to develop systems to identify at-risk students more accurately or perform with computational efficiency, and produce reliable outcomes.

REFERENCES

[1] Hendrastuty, N. (2024). Penerapan data mining menggunakan algoritma K-Means clustering dalam evaluasi hasil pembelajaran siswa. *Jurnal Ilmiah Informatika dan Ilmu Komputer (JIMA-ILKOM)*, 3(1): Article 1. https://doi.org/10.58602/jima-ilkom.v3i1.26

[2] Julakanti, S.R. (2024). Cloud-powered data mining: Unlocking hidden patterns in cloud storage. *International Journal of Intelligent Systems and Applications in Engineering*, 12(23s): Article 23s.

[3] Choque-Soto, V.M., Sosa-Jauregui, V.D., Ibarra, W. (2025). Characterization of the dropout student profile using data mining techniques. Revista de Gestão Social e Ambiental, 19(2): e011306. https://doi.org/10.24857/rgsa.v19n2-067

[4] Griep, K. et al. (2025). Ensuring ethical, transparent, and auditable use of education data and algorithms on AutoML. *Proceedings of the ACM*. https://doi.org/10.1145/3639479.3639492

[5] Lubis, A.H., Muliono, R., Khairina, N., Novita, N. (2024). The impact of K-Means on association rules mining algorithms performance. *JCOSITTE*. https://doi.org/10.30596/jcositte.v5i2.20907

[6] Erdiansyah, D., Abdullah, I.N., Tallo, A.J. (2025). Determination of potential business locations using data mining clustering. *Jurnal Pilar Nusa Mandiri*, 21(1): Article 1. https://doi.org/10.33480/pilar.v21i1.6295

[7] Antonio, R., Leong, H. (2023). Performance of synthetic minority over-sampling technique on support vector machine and K-nearest neighbor for sentiment analysis of metaverse in Indonesia. *Proxies: Jurnal Informatika*, 6(2): 160–170. https://doi.org/10.24167/proxies.v6i2.12459

[8] Lim, A.N.P., Leong, H. (2020). Hate speech prediction using K-Means algorithm. *Proxies: Jurnal Informatika*, 3(2): 98–104. https://doi.org/10.24167/proxies.v3i2.12430

[9] Qu, F. (2024). Research on online learning behavior of higher vocational students based on data mining. In: *Proceedings of the 4th International Conference on Artificial Intelligence and Education (ICAIE 2023)*, Atlantis Highlights in Computer Sciences, Vol. 15, pp. 30–37. https://doi.org/10.2991/978-94-6463-242-2_5

[10] Al-Badani, A.M., Shujaaddeen, A.A., Aljafare, M.M. (2025). Efficient mining of FP-growth algorithm structure and Apriori algorithm using OFIM for big data. *International Journal of Applied Information Systems*, 12(47): 8–15.

[11] Mkungudza, J., Twabi, H.S., Manda, S.O.M. (2024). Development of a diagnostic predictive model for determining child stunting in Malawi: A comparative analysis of variable selection approaches. *BMC Medical Research Methodology*, 24(1): 175. https://doi.org/10.1186/s12874-024-02283-6

[12] Stefanus, K., Leong, H. (2023). Comparison of random forest algorithm accuracy with XGBoost using hyperparameters. *Proxies: Jurnal Informatika*, 7(1): 15–23. https://doi.org/10.24167/proxies.v7i1.12464