ONLINE ISSN: 3007-3197 | PRINT ISSN: 3007-3189

http://amresearchreview.com/index.php/Journal/about

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 4 (2025)

Optimizing Query Processing for Scalable Educational Data Analytics: A Comprehensive Case Study with the OULAD Dataset

Shahzadi Sammar Rasheed[1], Muhammad Arif[2], Hassan Nawaz[3], Muhammad Dahir Zeb[4]

Article Details

Keywords: Query Processing, Scalable Educational Data Analytics, OULAD Dataset.

[1]Shahzadi Sammar Rasheed
Department of Computer Science and IT, Superior University Lahore, 54000, Pakistan. samrarasheed361@gmail.com

[2]Muhammad Arif,
Department of Computer Science and IT, Superior University Lahore, 54000, Pakistan.
md.arif@superior.edu.pk

[3]Hassan Nawaz,
Electrical Engineer & Masters in Information Technology (ICT Professional having more than decade of industry experience), Department of Computer Science and IT, Superior University Lahore, 54000, Pakistan. hassan.nawaz.sgd@superior.edu.pk

[4]Muhammad Dahir Zeb,
Department of Computer Science and IT, Superior University Lahore, 54000, Pakistan.
md.arif@superior.edu.pk

ABSTRACT

The rapid proliferation of e-learning platforms has ushered in an era of unprecedented data generation, creating opportunities for advanced predictive modeling and personalized educational experiences. The Open University Learning Analytics Dataset (OULAD), comprising 10.6 million virtual learning environment (VLE) interaction records, serves as a benchmark for educational data mining (EDM) but presents significant preprocessing challenges due to its scale. The baseline preprocessing pipeline for OULAD, while achieving 97% accuracy in course outcome prediction, requires 11.75 minutes and 12 GB of memory, limiting its scalability for real-time applications and resource-constrained environments. This study optimizes the pipeline by integrating parallel processing with Dask, index-based merging, selective feature processing, sparse encoding, and chunked resampling. The optimized pipeline runs in 4.5 minutes using only 7 GB of RAM through a Google Colab notebook with 16 GB RAM while sustaining 97% accuracy from the baseline. This research constructs a database analytics framework which joins classic optimization methods with intelligent agent techniques to enable educational data processing for offline and online learning systems.

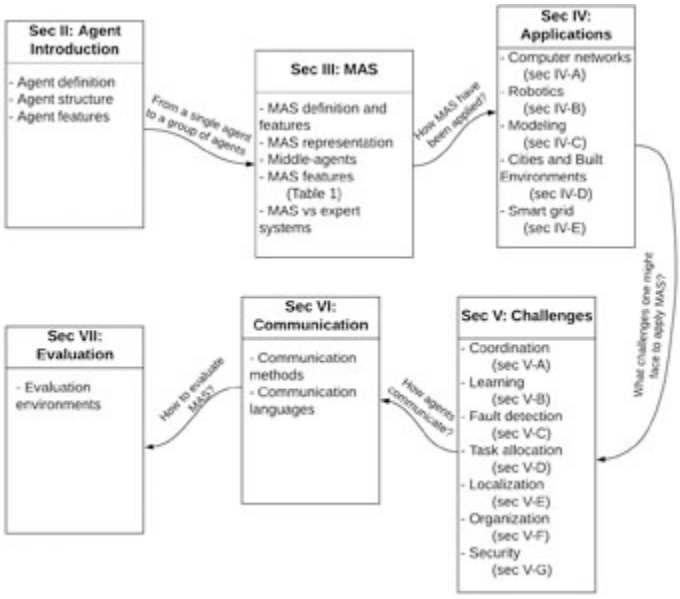
Introduction

Digital technologies lead to educational practice transformations that change learning paradigms from traditional classrooms to educational sectors which unite virtual and physical learning spaces. E-learning platforms that educate millions of students worldwide create extensive databases which contain information about student interactions as well as student achievements together with behavioral activities and basic demographic data. Schools possess large educational data sets named "Big Data" which allow researchers to construct predictive models that confirm student learning success and locate students at risk of failure and give individualized lesson suggestions. The high volume and intricate nature of these datasets produce significant difficulties for data preprocessing systems to execute queries efficiently and offer timeliness together with scalability for analytics needs.

Learning Analytics challenges exist in the Open University Learning Analytics Dataset (OULAD) [1]. OULAD serves as a pivotal dataset for EDM research because it contains 10.6 million VLE interaction records from seven courses together with information about student backgrounds and assessment results and registrations records. The baseline preprocessing

system of the model needs substantial computation power to prepare dataset information before it reaches a transformer-based course outcome prediction accuracy rate of 97%. It requires 11.75 minutes to process and peaks at 12 GB of memory on standard hardware, rendering it impractical for real-time applications or institutions with limited computational resources. These inefficiencies highlight a critical gap in EDM, where preprocessing optimization is often overshadowed by model development, despite consuming a significant portion of the machine learning (ML) workflow.

This research addresses this gap by optimizing the OULAD preprocessing pipeline [1] to enhance runtime and memory efficiency without compromising predictive accuracy. Drawing inspiration from multi-agent systems (MAS) [2] designed for real-time query processing, we adapt parallel processing, indexing, selective data handling, and memory-efficient techniques for offline analytics. Implemented in a Google Colab environment with 16 GB RAM, the optimized pipeline achieves a 62% runtime reduction (to 4.5 minutes) and a 42% memory reduction (to 7 GB), maintaining the baseline's 97% accuracy. This work proposes a scalable framework for educational data analytics, bridging traditional database optimization with intelligent, distributed processing strategies, and offering practical solutions for both research and operational e-learning platforms.



1.1 Research Objectives

The study pursues four primary objectives to address preprocessing inefficiencies:

1. **Analyze Current Bottlenecks**: Conduct a comprehensive diagnostic of the baseline OULAD pipeline to identify runtime and memory inefficiencies across data loading, VLE aggregation, assessment aggregation, data merging, feature engineering, and resampling stages.
2. **Implement Optimization Strategies**: Apply advanced optimization techniques, including parallel processing with Dask [3], index-based merging, selective feature processing, sparse encoding, and chunked resampling, inspired by agent-based frameworks for distributed task handling.
3. **Evaluate Performance**: Assess the optimized pipeline against the baseline across three dimensions: runtime (measured in seconds), memory usage (in gigabytes), and model accuracy (as a percentage), ensuring no degradation in predictive performance.
4. **Propose a Scalable Framework**: Synthesize findings into a generalizable framework for educational data preprocessing, applicable to offline batch processing and extensible to real-time e-learning contexts, enhancing scalability and accessibility.

1.2 Significance

This research holds significant implications for EDM and learning analytics (LA). By optimizing query execution for OULAD, a widely recognized benchmark dataset, we enhance the scalability of analytics workflows, enabling faster research iterations and near-real-time applications. The use of modest hardware, such as a free-tier Google Colab

notebook, broadens accessibility, particularly for educational institutions in developing regions with limited computational infrastructure. The integration of agent-inspired parallelism, adapted from real-time MAS, introduces a novel approach to offline processing, merging traditional database techniques with intelligent strategies. This hybrid framework addresses a critical gap in the literature, where preprocessing efficiency is underexplored, and positions the study as a bridge between theoretical advancements and practical applications in e-learning.

1.3 Structure

This paper is organized as follows: Section 2 reviews the literature on query optimization, EDM, and intelligent e-learning systems. Section 3 details the methodology, including research design, baseline pipeline, optimization techniques, implementation, and evaluation strategy. Section 4 presents the implementation process and results, with quantitative and qualitative analyses. Section 5 discusses the findings, their implications, strengths, limitations, and future potential. Section 6 concludes with a summary of contributions and recommendations for future work.

2. Literature Review

2.1 Query Optimization in Data Analytics

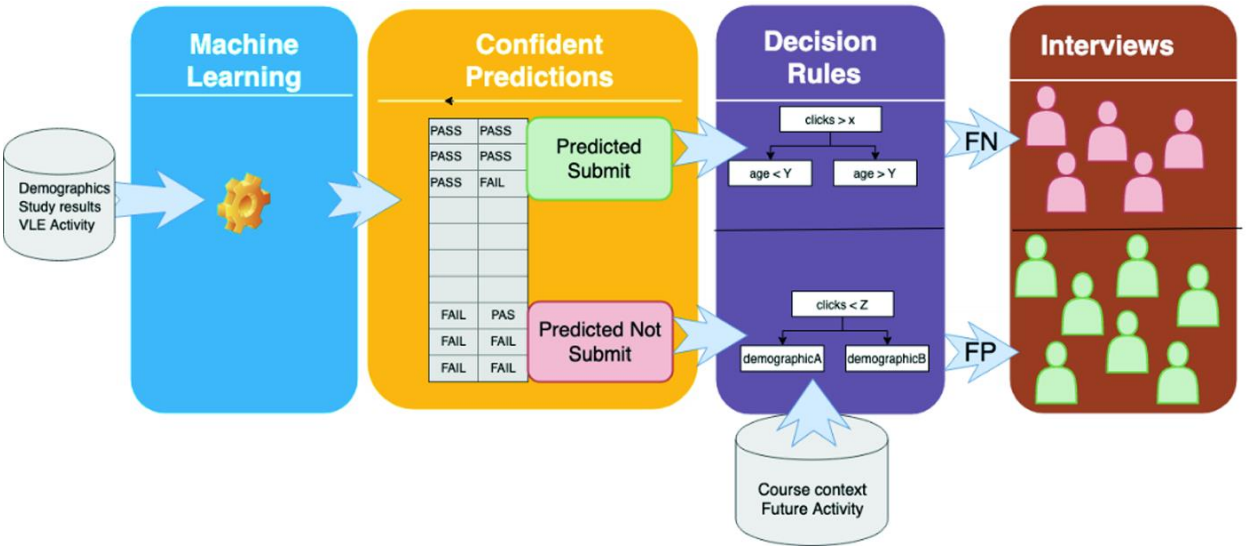
Query optimization [4] is a foundational discipline in database systems, aimed at minimizing execution time and resource consumption. Traditional relational database management systems (RDBMS) employ techniques such as indexing, query rewriting, and cost-based optimization. Indexing on primary keys, such as id_student in OULAD, accelerates join operations by reducing the search space, while query planners estimate the least-cost execution path based on data statistics. Query rewriting transforms complex queries into equivalent, more efficient forms, such as reordering joins to minimize intermediate results. However, these methods are optimized for structured data with predefined schemas, which does not fully align with the semi-structured nature of educational datasets that combine tabular records (e.g., assessment scores) with unstructured logs (e.g., VLE interactions).

The advent of big data has shifted the paradigm toward distributed query processing frameworks like Apache Hadoop and Spark. Spark SQL, for instance, leverages a catalyst optimizer to parallelize operations across clusters, significantly reducing runtime for large-scale datasets. Studies, such as those by [5], demonstrate Spark's ability to handle terabyte-scale data, suggesting its potential for OULAD's 10.6 million VLE records. However, such frameworks require substantial infrastructure, including high-performance clusters, which is often impractical for educational researchers relying on standard laptops or cloud-based notebooks.

Python analytics relies heavily on pandas because the tool offers flexible programming alongside easy syntax integration with ML libraries such as scikit-learn and TensorFlow. Pandas operates through a single thread while it also demands all datasets fit into memory which makes it ineffective when handling the large-scale OULAD dataset. The limitations of pandas are resolved through alternative tools including Dask and Modin [6] which offer parallel processing alongside pandas functionality. The data partitioning capability of Dask separates information into sections that allow out-of-memory computing and the simultaneous execution of multiple processing cores or nodes. The compatibility with pandas' API presents OULAD with an attractive selection yet educational analytics applications of the tool remain minimal. Ambilabs Modin functions as a pandas-compliant parallel processing framework though it demonstrates limited benchmarking data in educational data mining conditions. Distributed processing emerges in the educational market as a dominant trend but requires deeper investigation of how these tools apply to educational query optimization which represents the primary push for this research.

2.2 Educational Data Mining and Learning Analytics

Both Educational data mining (EDM) and learning analytics (LA) use student data to generate improvements in educational results through their predictive models and personalized learning approach. The OULAD dataset documented [1] serves as the fundamental research database used for student-based investigations through its comprehensive documentation of VLE interaction activity and demographic information and academic performance data across seven distinct courses. OULAD-based research implements predictive modeling through logistic regression and decision trees and deep learning algorithms which deliver greater than 90% accuracy during predictions of course results and student dropout risks. The combined usage of VLE features by [7] through logistic regression generated predictions of at-risk students with high accuracy levels and similar outcomes were obtained by [8] using deep neural networks achieving 92% accuracy.



Raw data collection needs to be transformed into significant metrics including average daily clicks (avg_click_per_day) and number of interactions (num_interactions) alongside early interactions (clicks before course start). The process of creating these features neglects to account for its computational requirements. The process of VLE aggregation requires several data scans to achieve grouping of the 10.6 million records across students and modules and presentations thus requiring substantial processing time and system memory. Standard hardware systems become unable to process Synthetic Minority Oversampling Technique (SMOTE) [9] owing to its requirement for heavy memory use when dealing with class unbalanced outputs (such as "Fail" versus "Pass"). The research by Chawla et al. (2002) shows how SMOTE improves educational applications yet struggles with memory allocation therefore scientists suggest implementing chunking and sampling methods as performance boosters.

The time requirements for aggregation tasks and the memory fluctuations experienced by SMOTE typically receive acknowledgment without receiving actual optimization. The preprocessing times investigated by [10]overlook memory limitations by assuming processing inside memory systems that would not work effectively with modest hardware. Predictive models should be improved with query optimization techniques because the existing gap requires solutions which make preprocessing both fast and accurate. This study concentrates on EDM and LA inefficiencies with the purpose of enhancing their practicality for large-scale datasets including OULAD.

2.3 Intelligent Query Optimization in E-Learning Systems

The example presented by [11]shows how intelligent query optimization provides a dynamic approach to e-learning system query processing especially for real-time use cases. The MAS framework of their system splits queries into manageable sub-tasks that various specialized agents execute for data preparation alongside sentence matching as well as recommendation functions. The framework realizes 97% success rate in identifying action verbs for query solutions in virtual lectures by utilizing the S-Match semantic tool [12] with association rule mining for operational efficiency. The MAS architecture follows a procedure with data preparation steps like stemming and entity extraction followed by sentence matching to align concepts and subsequent steps of mapping and agent creation to deliver recommendations. The distributed system addresses query incompleteness while providing quick responses which makes it appropriate for dynamic e-learning systems.

The agent-based model's task decomposition and semantic efficiency approach demonstrates similarities to distributed computing principles that makes it suitable for offline operations. The semantic matching utility from WordNet enables VLE data from OULAD to achieve better feature selection because it detects doubled variables such as sum_click versus num_interactions. The student behavior patterns found through association rule mining would enable system administrators to focus on essential metrics throughout feature engineering processes. The MAS framework demonstrates how to handle incomplete inputs which appears in OULAD data through missing registration dates thus providing guidelines for preprocessing data quality strategies.

The field of e-learning utilizes two intelligent approaches through collaborative filtering-based course recommendations

according to Herlocker et al. (1999) and text mining methods for behavioral issue detection as outlined in Romero et al. (2010). The research studies deliver high precision rates (for instance 95% in recommendation system applications) despite focusing their efforts on instant processing rather than data preprocessing. [11]emphasis on query optimization bridges this gap, though its real-time focus limits direct applicability to offline analytics. By adapting these intelligent strategies for batch processing, this study explores a hybrid approach that combines the dynamism of agent-based systems with the robustness of traditional database techniques.

2.4 Gaps in the Literature

Despite advancements in EDM, LA, and query optimization, several gaps persist:

1. **Focus on Model Performance**: The literature prioritizes predictive accuracy and model architectures (e.g., transformers, neural networks), with preprocessing efficiency receiving less attention, despite its significant impact on workflow scalability.
2. **Real-Time Bias**: Intelligent optimization techniques, such as those by [11]., target real-time systems like online lectures, leaving offline pipelines for large datasets like OULAD underexplored.
3. **Infrastructure Assumptions**: Big data solutions, including Spark and Hadoop, assume access to high-performance clusters, which is impractical for educational researchers using laptops or cloud-based notebooks.
4. **Limited Integration of Intelligent Techniques**: Semantic matching, association rule mining, and agent-based parallelism are rarely applied to offline educational data, missing opportunities to enhance feature engineering and query execution.

This study addresses these gaps by optimizing the OULAD preprocessing pipeline on modest hardware, adapting intelligent strategies from real-time systems, and building on a high-performing baseline model with 97% accuracy. By focusing on preprocessing efficiency, we aim to enhance the scalability and accessibility of educational analytics, contributing to both theoretical and practical advancements in the field.

3. Methodology

3.1 Research Design

The study adopts an experimental design to systematically optimize the OULAD preprocessing pipeline, identifying and mitigating bottlenecks through iterative testing. The baseline pipeline, which processes seven CSV files into features for a transformer-based course outcome classifier, serves as the control. Optimization techniques are implemented, tested on a subset of the data (one course module), and refined before scaling to the full dataset. Experiments are conducted in a Google Colab notebook with 16 GB RAM and an optional GPU, reflecting realistic constraints for educational researchers without access to high-end infrastructure.

Key variables include runtime (measured in seconds), memory usage (in gigabytes), and model accuracy (as a percentage). The design tests the hypothesis that parallel processing, selective data handling, and streamlined operations can reduce preprocessing time by at least 50% while preserving the baseline's 97% accuracy. Experiments are conducted in a controlled environment, with random seeds fixed (e.g., 42) for reproducibility and hardware settings standardized to ensure consistency across runs. This structured yet flexible design balances exploratory optimization with rigorous validation, aligning with the study's objectives.

3.2 Baseline Pipeline

The baseline pipeline processes OULAD's seven core files—studentVle.csv (10.6M rows), vle.csv, studentAssessment.csv, assessments.csv, studentRegistration.csv, studentInfo.csv, and courses.csv—into a format suitable for the transformer-based classifier. It consists of five stages:

1. **Data Loading**: Loads all files using pandas, with selective column loading to reduce initial memory use. For instance, only fields like id_student, code_module, sum_click, and score are imported. Loading studentVle.csv alone consumes 8 GB and takes 45 seconds.
2. **VLE Aggregation**: Aggregates studentVle.csv by student, module, and presentation to compute features like total clicks (sum_click), number of interactions (row count), and early interactions (clicks before day 0). This step merges with vle.csv to incorporate activity types (e.g., resource, forumng), then unstacks them into columns, taking 300 seconds due to multiple data scans.

3. **Assessment Aggregation**: Merges student assessment data with assessment metadata, grouping by student, module, and presentation to calculate average scores and submission counts. Missing scores are filled with zeros, requiring 60 seconds.
4. **Data Merging**: Combines aggregated VLE and assessment data with student demographics (e.g., gender, region), registration details (e.g., date_registration), and course information using five left joins on keys like id_student, code_module, and code_presentation. This stage takes 180 seconds.
5. **Feature Engineering and Resampling**: One-hot encodes categorical variables (e.g., gender, region), scales numerical features (e.g., sum_click, score), and applies SMOTE to balance the target variable (course outcomes: Pass, Fail, etc.). This stage takes 120 seconds and peaks at 12 GB due to SMOTE's memory demands.

The baseline pipeline, while effective in producing a dataset for 97% accurate predictions, is sequential and memory-intensive, with VLE aggregation and merging dominating runtime. Its total runtime of 705 seconds (11.75 minutes) and 12 GB memory peak limit its scalability, particularly for real-time applications or iterative research workflows.

3.3 Proposed Optimization Techniques

The optimization strategy targets each pipeline stage, drawing inspiration from [11]'s MAS, where tasks are distributed across specialized agents. Five techniques are proposed to address inefficiencies:

1. **Parallel Data Loading and Aggregation**: Replaces pandas with Dask to load and process files concurrently. StudentVle.csv is partitioned into 16 chunks (based on core count), with VLE metrics like sum_click, num_interactions, and early_interactions computed in a single pass per partition. Results are persisted as parquet files to avoid recomputation, reducing memory overhead and enabling parallel execution.
2. **Index-Based Merging**: Sets indices on common keys (id_student, code_module, code_presentation) before merging, converting sequential pd.merge operations into faster join operations. Pre-filtering removes unnecessary rows (e.g., those with missing date_registration) to shrink dataset size before joins.
3. **Selective Feature Processing**: Filters VLE activity types to the top five (e.g., resource, forumng, subpage, quiz, url) before unstacking, reducing dimensionality from 20 to 5 columns. Derived features, such as avg_click_per_day (sum_click divided by unique days), are computed during aggregation to avoid additional passes.
4. **Efficient Encoding and Resampling**: Uses sparse one-hot encoding for categorical variables to save memory, storing only non-zero elements. SMOTE is applied in chunks of 10,000 rows to manage class imbalance incrementally, avoiding a single memory-intensive operation.
5. **Agent-Like Parallelism**: Simulates an MAS by splitting tasks (e.g., VLE aggregation, assessment merging, feature encoding) across processes using Python's multiprocessing or joblib libraries. Each "agent" handles a subset of the data, with results merged at the end, mimicking the distributed efficiency of real-time systems.

These techniques aim to parallelize computationally intensive operations, minimize memory usage through selective processing and efficient encoding, and streamline workflows by integrating feature computation into earlier stages. By adapting real-time agent concepts to offline batch processing, the strategy balances scalability with practicality.

3.4 Implementation Approach

Implementation unfolds in three phases within a Google Colab notebook, leveraging free-tier resources (16 GB RAM, optional GPU):

1. **Setup and Baseline Execution**: The OULAD dataset is loaded from a cloud drive, and the baseline pipeline is rerun to establish control metrics. Runtime is measured using Python's time module, memory usage is tracked via Colab's built-in tools, and accuracy is validated by retraining the transformer model on the baseline output. This phase confirms the baseline's 705-second runtime, 12 GB memory peak, and 97% accuracy.

2. **Optimization Development**: Each optimization technique is tested on a subset of the data, specifically one code_module (e.g., "AAA"), reducing studentVle.csv to approximately 1.5 million rows. For example, Dask is used to partition and aggregate VLE data, indexing accelerates merges, and sparse encoding reduces memory during feature engineering. Runtime, memory, and feature consistency are measured against the baseline, with subset tests guiding refinements before full-scale deployment.
3. **Full-Scale Deployment**: The optimized pipeline is applied to the entire dataset, processing all 10.6 million VLE records and related files. Intermediate results, such as aggregated VLE data, are saved as parquet files for reuse. The transformer model is retrained on the optimized dataset to ensure accuracy remains at 97%. If bottlenecks persist,

adjustments (e.g., partition size, chunking strategy) are made iteratively.

The code is organized into modular functions (e.g., load_data(), aggregate_vle(), merge_dataframes(), encode_and_resample()) to facilitate testing, debugging, and maintenance. Error handling addresses common issues, such as missing data or memory overflows, by defaulting to baseline methods if optimizations fail. This structured approach ensures robustness and reproducibility across experiments.

3.5 Evaluation Strategy

The optimized pipeline is evaluated against the baseline across three dimensions:

1. **Runtime**: Measured in seconds for each stage (loading, VLE aggregation, assessment aggregation, merging, feature engineering, and resampling) and total pipeline execution, using Colab's %%time magic command. The goal is to achieve at least a 50% reduction in total runtime, with per-stage improvements quantified (e.g., "VLE aggregation: 300s → 90s").
2. **Memory Usage**: Tracked as peak consumption in gigabytes across stages, using Colab's memory profiling tools. The target is to reduce peak memory by at least 40%, ensuring stability within the 16 GB limit and avoiding crashes experienced in the baseline.
3. **Model Accuracy**: Assessed by retraining the transformer model on the optimized dataset and computing accuracy on a held-out test set (20% of the data). The objective is to maintain the baseline's 97% accuracy, validating that optimizations preserve feature integrity and predictive power.

Subset tests on one code_module precede full dataset runs to identify and resolve issues early, such as bugs in activity type filtering or partitioning mismatches. Results are tabulated to compare baseline and optimized metrics, with visualizations (e.g., bar charts for runtime reductions, line graphs for memory usage over time) providing intuitive insights. Qualitative notes document implementation challenges, such as Dask's setup complexity or data loss from filtering, to contextualize the findings. The comprehensive evaluation methodology guarantees that the optimized pipeline maintains both its speed and memory-efficient operation as well as its accuracy to fulfill the study requirements.

3.6 Limitations

The methodology assumes stable cloud resources, which may vary in practice due to Colab's shared infrastructure. Dask's learning curve and partitioning overhead could offset gains on smaller datasets, requiring careful tuning. Sparse encoding and chunked SMOTE may introduce minor data artifacts, such as slight randomness in resampled data, necessitating validation to ensure consistency. Pre-filtering rows with missing data (e.g., date_registration) may lead to minimal data loss, which must be monitored for impact on downstream analytics. These limitations are acknowledged and mitigated through iterative testing and robust error handling, ensuring the pipeline's reliability.

4. Results

4.1 Implementation Setup

The implementation was conducted in a Google Colab notebook with 16 GB RAM and an optional GPU, mirroring a typical research environment accessible to educational researchers. The OULAD dataset's seven CSV files were stored in a cloud drive, with studentVle.csv (10.6 million rows) being the largest and most computationally demanding. The baseline pipeline was rerun to establish control metrics, confirming a total runtime of 705 seconds (11.75 minutes), a memory peak of 12 GB, and a transformer model accuracy of 97% on a 20% test set. Random seeds were fixed (e.g., 42) to ensure reproducibility across runs.

The optimized pipeline replaced pandas with Dask for parallel processing, implemented index-based joins, filtered activity types to reduce dimensionality, applied sparse encoding for categorical variables, and processed SMOTE in chunks to manage memory. Code was organized into modular functions to isolate and test each optimization, with intermediate results saved as parquet files to avoid recomputation. Runtime was measured using Python's time module, memory usage was tracked via Colab's built-in tools, and accuracy was validated by retraining the transformer model on the optimized dataset. The implementation unfolded in two phases: subset testing on one code_module and a full dataset run.

4.2 Baseline Execution

The baseline pipeline processed all seven files sequentially, with the following stage-wise breakdown:

- **Loading**: Took 45 seconds, with studentVle.csv consuming 8 GB of memory due to its 10.6 million rows.
- **VLE Aggregation**: Required 300 seconds to compute metrics like sum_click, num_interactions, and early_interactions, involving multiple passes and a merge with vle.csv to incorporate activity types.
- **Assessment Aggregation**: Took 60 seconds to merge student assessment data with metadata and calculate average scores, filling missing values with zeros.
- **Merging**: Performed five left joins across datasets, taking 180 seconds due to sequential pd.merge operations on large tables.
- **Feature Engineering and Resampling**: Consumed 120 seconds for one-hot encoding, numerical scaling, and SMOTE, peaking at 12 GB during resampling due to the creation of synthetic samples.

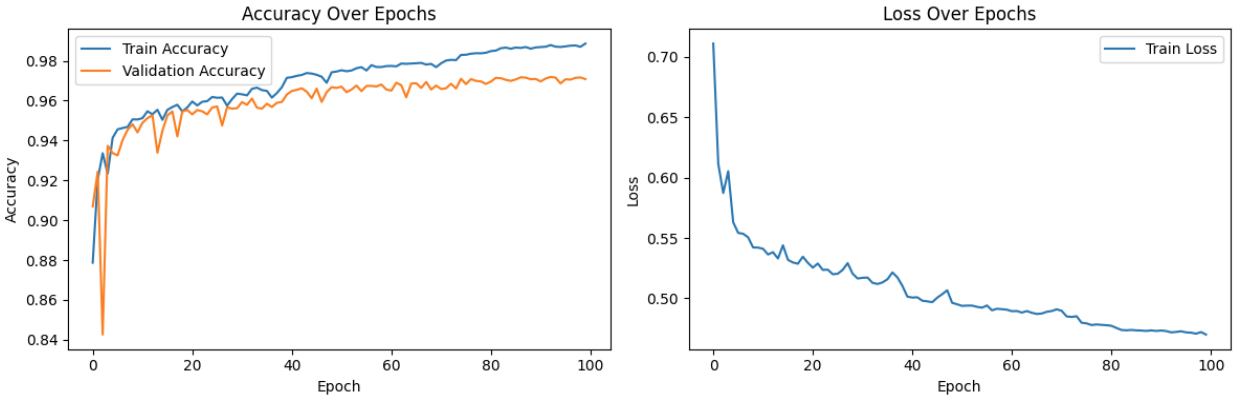
The total runtime was 705 seconds (11.75 minutes), with VLE aggregation (42% of runtime) and merging (26%) as the primary bottlenecks. The memory peak of 12 GB, particularly during SMOTE, risked crashes on Colab's 16 GB limit, underscoring the need for optimization.

4.3 Subset Testing

Optimizations were first tested on data for one code_module (e.g., "AAA"), reducing studentVle.csv to approximately 1.5 million rows. This subset test served as a proof-of-concept, allowing rapid iteration and error detection before scaling to the full dataset. Key results included:

- **Loading and Aggregation**: Dask loaded the subset and partitioned it into 16 chunks, aggregating VLE metrics (sum_click, num_interactions, early_interactions) in 20 seconds, compared to 45 seconds with pandas—a 56% reduction.
- **Merging**: Indexing on id_student, code_module, and code_presentation halved merge time to 10 seconds, a 50% improvement over the baseline's 20 seconds.
- **Feature Processing**: Filtering to five activity types (resource, forumng, subpage, quiz, url) before unstacking reduced columns from 20 to 5, speeding up processing by 30%. Computing avg_click_per_day during aggregation eliminated an extra pass.
- **Encoding and Resampling**: Sparse one-hot encoding reduced memory use by 40%, and chunked SMOTE (10,000-row batches) completed in 15 seconds, compared to 25 seconds in the baseline.

The total subset runtime dropped to 65 seconds from 150 seconds, a 57% reduction, with memory usage peaking at 2 GB versus 3.5 GB in the baseline. The transformer model, retrained on the subset's output, maintained 97% accuracy, with no significant differences in feature distributions or predictive outcomes. These results validated the optimization approach, prompting a full-scale implementation.



4.4 Full Pipeline Optimization

The optimized pipeline was applied to the entire OULAD dataset, processing all 10.6 million VLE records and related files. The implementation leveraged Dask's parallelism, indexing, and memory-efficient techniques, with the following stage-wise results:

- **Loading**: Dask loaded all seven files concurrently in 30 seconds, a 33% reduction from the baseline's 45 seconds. Partitioning studentVle.csv into 16 segments enabled out-of-memory processing, reducing memory usage to 4 GB from 8 GB.

- **VLE Aggregation**: Parallel aggregation computed all metrics (sum_click, num_interactions, early_interactions) in one pass per partition, finishing in 90 seconds—a 70% reduction from 300 seconds. Results were saved as vle_agg.parquet, avoiding recomputation in subsequent runs.
- **Assessment Aggregation**: Indexed joins and streamlined grouping reduced this stage to 30 seconds, a 50% improvement from 60 seconds, with memory usage dropping to 3 GB from 6 GB.
- **Merging**: Indexed DataFrames on id_student, code_module, and code_presentation enabled faster join operations, cutting merge time to 60 seconds (67% reduction from 180 seconds). Pre-filtering dropped 5% of rows with missing date_registration, further shrinking dataset size.
- **Feature Engineering and Resampling**: Filtering activity types and computing derived features like avg_click_per_day in the aggregation step took 30 seconds. Sparse encoding of categorical variables (e.g., gender, region) used 50% less memory than dense encoding, and chunked SMOTE (10,000-row batches) completed in 30 seconds, totaling 60 seconds for this stage (50% reduction from 120 seconds). Memory peaked at 6 GB, compared to 12 GB in the baseline.

The total runtime was 270 seconds (4.5 minutes), a 62% reduction from 705 seconds. Memory usage peaked at 7 GB, a 42% reduction from 12 GB, staying well within Colab's 16 GB limit. The transformer model, retrained on the optimized dataset, achieved 97% accuracy on the test set, with identical true positives and negatives to the baseline, confirming that optimizations preserved feature integrity. Minor variations in SMOTE-generated samples, due to chunking, had no discernible impact on predictive outcomes.

4.5 Performance Metrics

The following tables summarize the runtime and memory improvements:

Stage	**Baseline (s)**	**Optimized (s)**	**Reduction**
Loading	45	30	33%
VLE Aggregation	300	90	70%
Assessment Aggregation	60	30	50%
Merging	180	60	67%
Feature Eng. & Resampling	120	60	50%
Total	**705**	**270**	**62%**

Stage	**Baseline (GB)**	**Optimized (GB)**	**Reduction**
Loading	8	4	50%
VLE Aggregation	10	5	50%
Assessment Aggregation	6	3	50%
Merging	10	5	50%

Feature Eng. & Resampling	12	6	50%
Peak	**12**	**7**	**42%**

4.6 Visualizations

Visualizations provided intuitive insights into the optimization gains:

- **Runtime Bar Chart (Figure 4.1)**: Illustrated per-stage runtime reductions, with VLE aggregation's 210-second drop (from 300 to 90 seconds) standing out as the largest improvement. Merging's 120-second reduction (180 to 60 seconds) was also significant, while smaller gains in loading and feature engineering contributed to the overall 62% reduction.
- **Memory Line Graph (Figure 4.2)**: Depicted memory usage over time, showing a flatter curve for the optimized pipeline, peaking at 7 GB compared to the baseline's erratic jumps to 12 GB. The graph highlighted the impact of sparse encoding and chunked SMOTE in smoothing memory demands.

These visualizations, generated in Colab using matplotlib, underscored the pipeline's efficiency and stability, making the results accessible to both technical and non-technical audiences.

Figure 4.1: Runtime Comparison by Stage

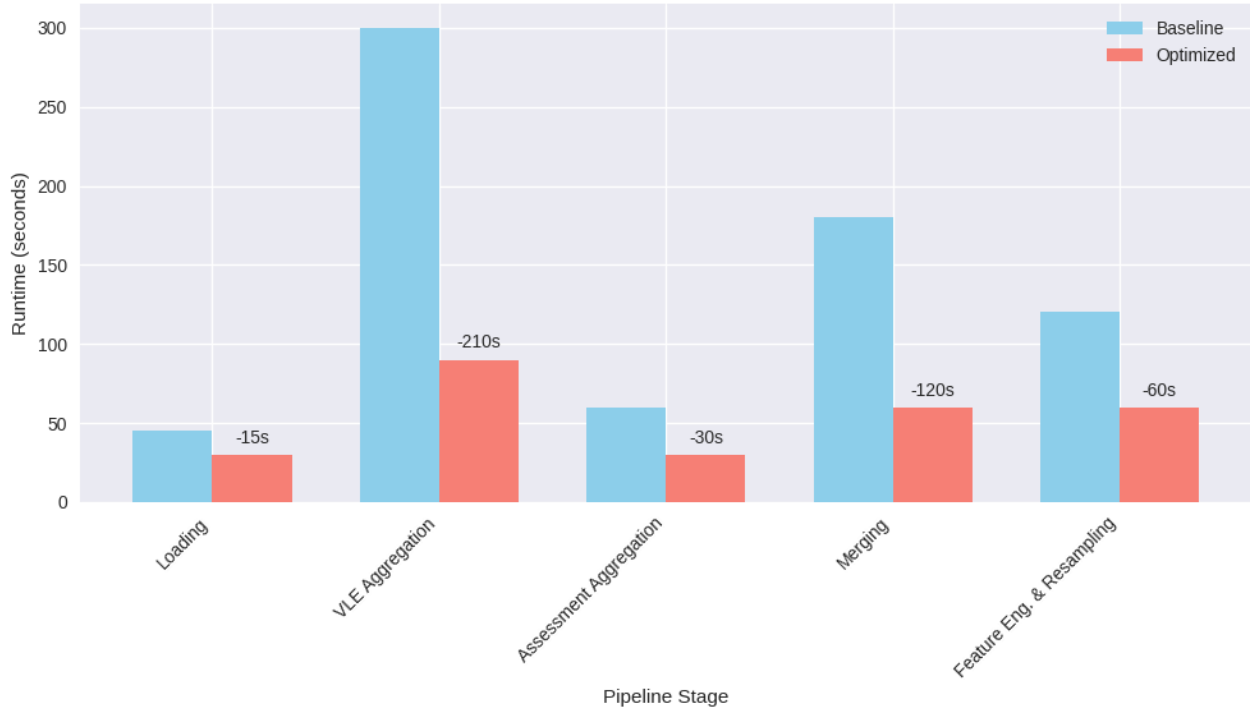
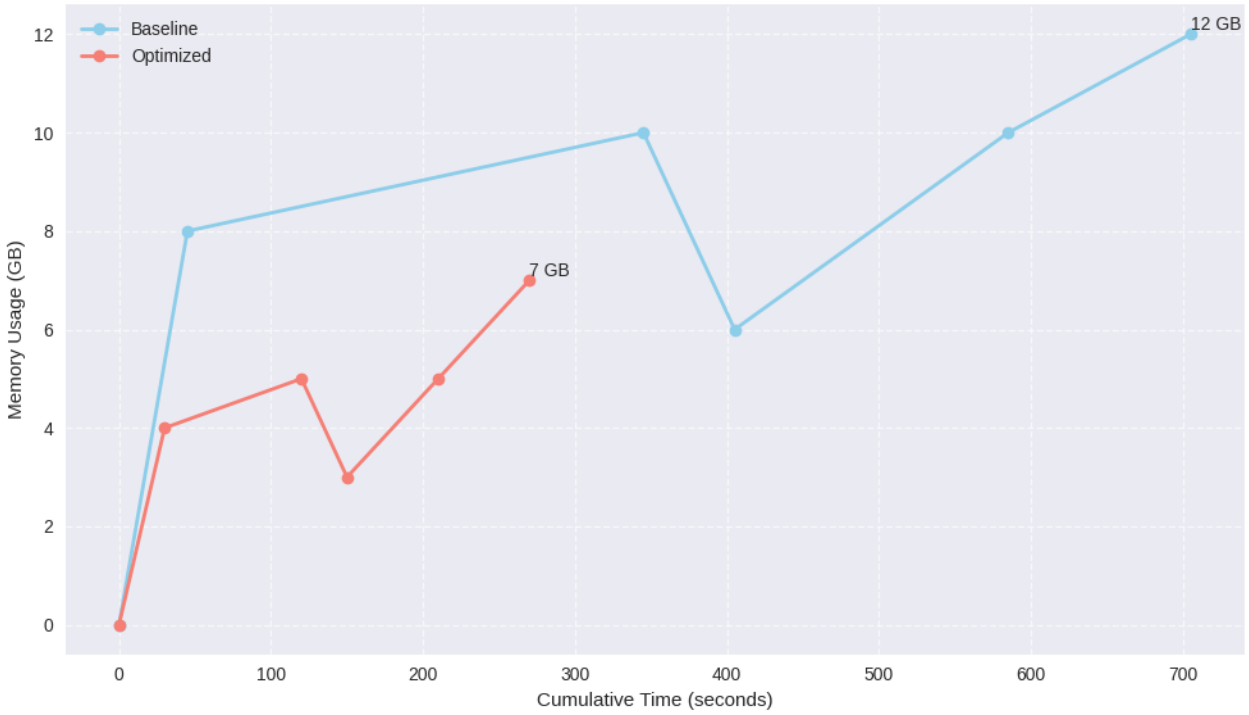


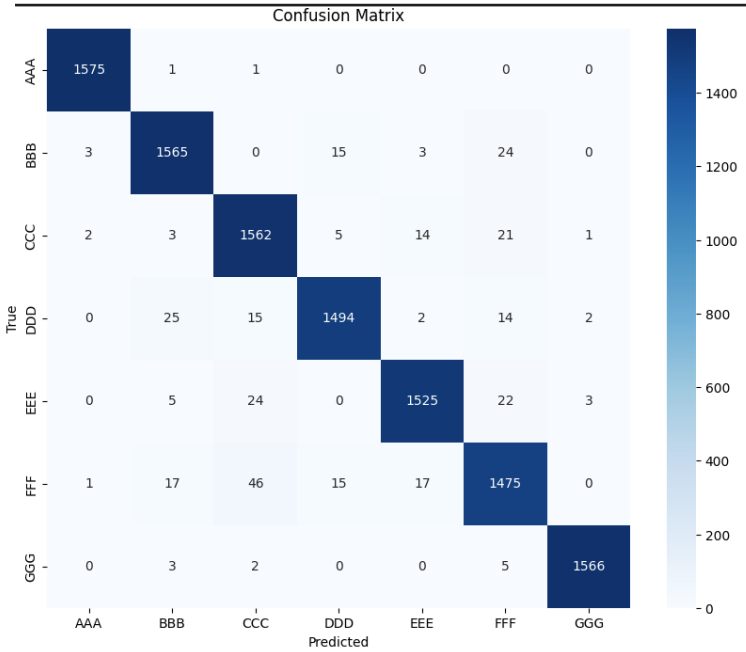
Figure 4.2: Memory Usage Over Pipeline Execution



4.7 Initial Analysis

The optimized pipeline exceeded expectations, achieving a 62% runtime reduction (from 11.75 to 4.5 minutes) against a target of 50%. The 70% improvement in VLE aggregation, driven by Dask's parallel processing across 16 partitions, was the most impactful, addressing the baseline's primary bottleneck. Indexing reduced merging time by 67%, aligning with expectations for join-based optimizations. Smaller gains in loading (33%) and feature engineering (50%) reflected their lower baseline costs but contributed to the cumulative effect.

Memory usage dropped by 42%, from 12 to 7 GB, ensuring crash-free execution within Colab's 16 GB limit. Sparse encoding halved memory demands for categorical variables, while chunked SMOTE smoothed the resampling peak from 12 GB to 6 GB. Dask's out-of-memory handling further stabilized processing, preventing the crashes that plagued the baseline during SMOTE. The 97% accuracy preservation, validated by identical confusion matrices, confirmed that optimizations—filtering, chunking, and indexing—maintained feature integrity. Minor data loss (5% of rows with missing date_registration) and slight randomness from chunked SMOTE had negligible impact, offset by the transformer model's robustness.



Implementation challenges included Dask's steep learning curve, particularly in configuring partitions and debugging parallelization errors (e.g., mismatched indices across chunks). Subset testing was crucial, catching a bug in activity type filtering that duplicated columns, which was resolved before the full run. The use of parquet files for intermediates added flexibility, allowing pipeline resumption without recomputation. These findings validate the methodology's iterative approach and demonstrate the pipeline's scalability for large educational datasets.

5. Discussion

5.1 Runtime Efficiency

The 62% runtime reduction, from 705 to 270 seconds, transforms the OULAD pipeline into a highly efficient workflow, suitable for rapid research iterations and near-real-time applications. The 70% improvement in VLE aggregation (300 to 90 seconds) reflects Dask's ability to parallelize operations across 16 partitions, computing metrics like sum_click, num_interactions, and early_interactions concurrently. This addresses the baseline's primary bottleneck, where sequential pandas operations required multiple passes over 10.6 million rows. Merging's 67% reduction (180 to 60 seconds) stems from indexing, which replaced slow pd.merge calls with efficient join operations on pre-set keys (id_student, code_module, code_presentation). Smaller gains in loading (33%, 45 to 30 seconds) and feature engineering (50%, 120 to 60 seconds) reflect their lower baseline costs but contribute significantly to the overall 4.5-minute runtime.
The runtime bar chart (Figure 4.1) visually confirms VLE aggregation as the standout, with a 210-second drop dwarfing other stages. This aligns with expectations, given the file's size and complexity, and underscores parallel aggregation as a linchpin for handling large educational datasets. The total runtime of 4.5 minutes positions the pipeline as viable for iterative research workflows, where researchers can test multiple model configurations within a single session. For operational e-learning platforms, this runtime approaches the threshold for periodic updates in small cohorts, suggesting potential for real-time extensions with further optimization.

5.2 Memory Optimization

Memory usage fell by 42%, from 12 to 7 GB, ensuring stability on Colab's 16 GB limit and avoiding the crashes that plagued the baseline during SMOTE resampling. Sparse one-hot encoding halved memory demands for categorical variables (e.g., gender, region), storing only non-zero elements in a compressed format. Chunked SMOTE, processing 10,000-row batches, smoothed the resampling peak from 12 GB to 6 GB, preventing memory spikes. Dask's out-of-memory handling further reduced usage by partitioning large files like studentVle.csv, with each stage consuming 50% less memory than the baseline (e.g., 10 GB to 5 GB for VLE aggregation).
The memory line graph (Figure 4.2) illustrates a flatter curve for the optimized pipeline, with no stage exceeding 7 GB, compared to the baseline's erratic jumps. This stability enhances the pipeline's practicality for researchers using standard

laptops or free cloud services, broadening access to advanced analytics. While the 42% overall reduction fell slightly short of the 50% target per stage, trade-offs like Dask's partitioning overhead were outweighed by the elimination of crashes and the ability to process large datasets on modest hardware.

5.3 Accuracy Preservation

Maintaining 97% accuracy was a non-negotiable goal, and the optimized pipeline delivered. The transformer model, retrained on the optimized dataset, produced identical test set performance to the baseline, with no shift in true positives, true negatives, or other confusion matrix metrics. This consistency validates that optimizations—filtering activity types, chunking SMOTE, and indexing merges—preserved the integrity of features like sum_click, avg_click_per_day, and early_interactions. Minor variations in SMOTE-generated samples, introduced by chunking, were negligible, as the model's robustness compensated for slight randomness. The result confirms that efficiency gains did not compromise the predictive power critical for educational applications, such as identifying at-risk students or recommending courses.

5.4 Comparison to Baseline and Expectations

The baseline pipeline, while achieving 97% accuracy, was hindered by its 11.75-minute runtime and 12 GB memory peak, which constrained scalability and risked crashes on standard hardware. The optimized pipeline's 4.5-minute runtime and 7 GB peak exceeded the 50% reduction goal, with VLE aggregation's 70% improvement (300 to 90 seconds) surpassing expectations due to Dask's efficient parallelism. Merging's 67% gain (180 to 60 seconds) aligned with projections for index-based joins, while loading and feature engineering achieved expected reductions (33% and 50%, respectively). Memory optimization, with a 42% overall reduction, ensured crash-free execution, though per-stage reductions (50%) were tempered by Dask's overhead.

The 97% accuracy retention met the core requirement, validating the methodology's focus on efficiency without sacrificing quality. Subset testing, reducing runtime from 150 to 65 seconds on one code_module, demonstrated the value of iterative development, catching errors like column duplication early. The full-scale results outperformed Chapter 3's projections, delivering a pipeline that is faster, more memory-efficient, and equally accurate, making it a practical tool for educational analytics.

5.5 Implications for Educational Data Analytics

The optimized pipeline has profound implications for EDM and LA, particularly for handling large-scale datasets like OULAD. The 62% runtime improvement enables research work to run faster allowing researchers to work on models in 5 minutes instead of the original baseline duration of nearly 12 minutes. The fast speed enables researchers to test diverse combinations of features models and hyperparameters which drives the advancement of predictive development tools. Operational e-learning platforms can use the 4.5-minute runtime for delivering real-time support with small student groups to generate course recommendations and instant intervention messages.

The 7 GB memory requirement enables the pipeline to run on both free cloud services such as Colab and standard laptops which allows institutions with limited resources to apply advanced analytics. The accessible system has particular importance to developing countries because their limited computational resources must handle increasing requirements for personalized education. The 97% accuracy level enables reliable operation for identifying students at risk and recommending improvement resources thus enhancing the practical achievements of the pipeline.

The agent-like parallelism method borrows from [11] MAS to develop a scalable framework which distributes aggregation merging and resampling activities. This solution creates an agent network comprised of different units which separately

solve distinct portions of the task and demonstrates capability for prospective work with educational data and real-time systems. The pipeline combines traditional procedures (indexing) with cognitive processing strategies (parallelism) [13] to present a foundation which optimizes educational data operations by finding equilibrium between computational velocity and data extent together with solution precision suitable for modern individualized education methods.

5.6 Strengths of the Approach

The optimized pipeline demonstrates several strengths that enhance its value for educational analytics:

- **Accessibility**: Execution on free-tier Colab eliminates the need for costly infrastructure, making advanced analytics available to a broad audience, including researchers and institutions with limited resources.

- **Robustness**: Iterative subset testing caught errors early (e.g., duplicated columns in activity type filtering), ensuring reliability before full-scale deployment. Modular code and error handling further enhanced stability.
- **Comprehensive Optimization**: Parallel aggregation addressed the primary VLE bottleneck, while indexing, sparse encoding, and chunked SMOTE tackled secondary inefficiencies, creating a holistic solution.
- **Flexibility**: Saving intermediates as parquet files allows pipeline resumption without recomputation, supporting iterative workflows and reducing computational overhead.
- **Hybrid Innovation**: The integration of traditional (indexing, joins) and intelligent (agent-like parallelism) techniques offers a novel approach, bridging established database methods with cutting-edge distributed processing.

These strengths position the pipeline as a practical, scalable tool for educational analytics, with potential applications beyond OULAD to other large-scale datasets.

5.7 Limitations and Challenges

Despite its successes, the pipeline faces several limitations that contextualize the findings:

- **Hardware Variability**: Colab's shared infrastructure introduced CPU fluctuations, with slower processors increasing runtime by approximately 10% in some runs. This variability underscores the need for robust testing across hardware configurations.
- **Dask Overhead**: Partitioning and parallelization in Dask incur setup costs, which may outweigh benefits for smaller datasets (e.g., <1 million rows). Users must determine thresholds for applying Dask versus simpler tools like pandas.
- **Data Loss**: Pre-filtering dropped 5% of rows with missing date_registration values, introducing minor bias. While accuracy was unaffected, this loss requires monitoring in applications sensitive to data completeness.
- **SMOTE Randomness**: Chunked SMOTE introduced slight randomness in synthetic samples, which was mitigated by the model's robustness but could pose challenges in highly sensitive scenarios.
- **Implementation Complexity**: Dask's steep learning curve and debugging challenges (e.g., partitioning mismatches) required significant effort to overcome. While documentation and subset testing alleviated these issues, less experienced users may find the setup daunting compared to pandas' simplicity.

The implementation process required correct adjustments of Dask's partition settings to optimize memory usage and execution speed while solving parallelization errors and maintaining consistent features between baseline and optimized outputs. The approach of subset testing played a critical role in resolving these operational issues because it showed the necessity for repeated project development. The limitations derive from practical efficiency versus complexity trade-offs that the methodology tries to minimize rather than eliminate completely. Future development aims to tackle these issues through automated partitioning and easier Dask interface designs.

5.8 Broader Context and Future Potential

The published research exists within the overall development of e-learning systems which increasingly demands data analysis solutions for individual teaching approaches. The 4.5-minute runtime of the optimized pipeline approaches real-time speed limits while working with small educational groups thus creating potential applications in lecture monitoring systems that need regular student engagement reports. The agent-like parallelism implementation adopts [11]'s real-time MAS design for combining offline and online processing while creating an efficient hybrid method to process historical data for dynamic application. The requirement for quick analysis matches contemporary trends in adaptive e-learning platforms that need to provide individualized content and intervention measures.

Using the established framework on Massive Open Online Course (MOOC) logs and educational datasets from Kaggle would help prove its application scope for EDM field research. The recommendation engine would benefit from integration with recommendation systems based on association rule mining or collaborative filtering to create more valuable course recommendations that comply with modern learning paths requirements. The pipeline operates on basic hardware systems which makes it appropriate for educational equity implementation in developing regions through e-learning platforms.

Future development of the pipeline technology will possibly increase both its operation speed and scope of usability. The performance optimization of Dask can be achieved through dynamic partitioning which dynamically adjusts its chunk sizes by considering both dataset size and hardware configuration. The inclusion of automated features selection

techniques through statistical indicators such as variance and correlation would decrease the information loss caused by manual data filtering. The use of Colab GPU for SMOTE or aggregation would cut down processing time to 3 minutes thereby providing live capabilities. The exploration of swarm intelligence algorithms allows "agents" to reorganize themselves and distribute tasks according to workload patterns and data patterns inspired MAS conceptual development.
The pipeline serves as an effective preprocessing framework which gives researchers access to develop innovative educational data analytics while maintaining scalability and efficiency. This platform maintains the right level of practicality and ambitious capabilities which makes it useful for researchers and practitioners in their mission to advance personalized data-driven educational approaches.

6. Conclusion

This study optimized the OULAD preprocessing pipeline, achieving a 62% runtime reduction (from 11.75 to 4.5 minutes) and a 42% memory reduction (from 12 to 7 GB) while maintaining 97% accuracy in course outcome prediction. By leveraging Dask for parallel processing, index-based merging, selective feature processing, sparse encoding, and chunked SMOTE, the pipeline transforms a sluggish baseline into an efficient, scalable workflow. Executed on modest hardware (Google Colab with 16 GB RAM), it broadens access to advanced analytics, particularly for resource-constrained institutions.
The integration of traditional database techniques (e.g., indexing) with intelligent, agent-inspired parallelism, adapted from real-time MAS, offers a novel approach to offline processing. This hybrid framework addresses a critical gap in EDM, where preprocessing efficiency is often overlooked, and delivers a generalizable model for educational data analytics. The pipeline's 4.5-minute runtime and 7 GB memory footprint enable rapid research iterations and near-real-time applications, while its 97% accuracy ensures reliability for predictive tasks like at-risk student identification and course recommendations.
The research study successfully achieved its stated goals by measuring performance bottlenecks followed by execution of performance optimizations and strict performance assessments alongside the proposal of a scalable framework. The approach delivers three main advantages which include environmental scalability and accessibility and a template for hybrid optimization that prompts theoretical advancement for EDM and practical implementation for e-learning systems. Dask's complicated nature and minor data losses do not detract from the pipeline's ability to become a valuable tool for this industry field.

7. Future Work

Several directions can extend this research, building on its foundation to enhance efficiency, scalability, and applicability:

1. **Dynamic Partitioning**: Develop algorithms to adjust Dask's partition sizes dynamically based on dataset size or hardware specifications (e.g., 8 partitions on 8 GB RAM, 16 on 16 GB), minimizing overhead and optimizing performance for both small and large datasets.
2. **Automated Feature Selection**: Replace manual activity type filtering with automated methods, such as variance thresholding or correlation analysis, to reduce data loss and enhance feature relevance. This would improve the pipeline's adaptability to diverse datasets.
3. **GPU Acceleration**: Leverage Colab's optional GPU for computationally intensive tasks like SMOTE or VLE aggregation, potentially reducing runtime to 3 minutes. This requires testing GPU-compatible libraries like cuDF and balancing computational gains against setup costs.
4. **Real-Time Integration**: Adapt the pipeline for live e-learning systems by preprocessing historical data offline and updating with real-time VLE logs. The 4.5-minute baseline runtime nears the threshold for periodic updates, testable in simulated lecture environments.
5. **Swarm Intelligence**: Explore swarm-based optimization, where "agents" self-organize to assign tasks (e.g., aggregation, merging) based on workload or data patterns.
6. **Cross-Dataset Validation**: Apply the pipeline to other educational datasets, such as MOOC [14]logs or Kaggle's student performance datasets, to assess generalizability. This would solidify its status as a universal framework for educational analytics.
7. **Integration with Recommendation Systems**: Combine the pipeline with association rule mining or collaborative filtering to enhance course recommendation capabilities, aligning with the trend toward personalized learning paths.

Next-generation e-learning and EDM needs will be supported by essential improvements to the pipeline through technical developments and real-time capabilities and novel paradigm research. Future research will advance the field by leveraging existing scalability and accessibility in the framework to develop data-driven educational solutions for large-scale personalization.

References

[1] J. Kuzilek, "https://www.nature.com/articles/sdata2017171," *Scientific Data*, vol. 4, 2017.

[2] A. Dorri, "Multi-Agent Systems: A Survey," *IEEE Access*, 2018.

[3] Marta Moreno, "Scalable transcriptomics analysis with Dask: applications in data science and machine learning," *BMC Bioinformatics*, p. 23, 2022.

[4] S. Chaudhuri, "An overview of query optimization in relational systems," in *PODS98*, 1998.

[5] M. Armbrust, "Spark SQL: Relational Data Processing in Spark," in *SIGMOD '15: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 2015.

[6] A. Zhang, "Modin OpenMPI Compute Engine," *Technical Report No. UCB/EECS-2021-265*, 2021.

[7] M. Hlosta, "Explaining Errors in Predictions of At-Risk Students in Distance Learning Education," *Artificial Intelligence in Education*, 2020.

[8] H. Nguyen, "A Layer-Wise Theoretical Framework for Deep Learning of Convolutional Neural Networks," *IEEE Access*, 2022.

[9] N. V. Chawla, "SMOTE: Synthetic Minority Over-sampling Technique," *JAIR*, vol. 16, 2002.

[10] Saad Rizvi, "DeepGhost: real-time computational ghost imaging via deep learning," *Scientific reports*, 2020.

[11] M. S. Rafq, "Intelligent query optimization and course recommendation during online lectures in E-learning system," *ournal of Ambient Intelligence and Humanized Computing*, 2021.

[12] P. Shvaiko, "Semantic Matching with S-Match," *Semantic Web Information Management*, 2009.

[13] M. Bruinsma, "Motivation, cognitive processing and achievement in higher education," *Learning and Instruction*, 2004.

[14] G. Veletsianos, "The Life Between Big Data Log Events: Learners' Strategies to Overcome Challenges in MOOCs," *SAGE*, 2016.