Design of a Machine Learning-Based Information Integration System for Local Applied Undergraduate Colleges

Yijian Zhang *

School of Economics, Wuhan Business University

Wuhan, 430058, China

Abstract. This paper designs an educational information integration system based on machine learning, aimed at enhancing management efficiency and decision-making quality in local applied undergraduate colleges. The system employs a layered architecture and microservice design, comprising four core functional modules: student information management, teaching resource management, teacher assessment, and intelligent decision support. Machine learning techniques are integrated into the design to enable functionalities such as student performance prediction and teaching resource optimization. The data model accommodates both relational and non-relational data to meet complex data processing needs in the education sector. The system architecture considers performance, reliability, and security, supporting high-concurrency access and big data processing. User interface design emphasizes interaction experience and operational efficiency. This design provides innovative ideas for educational information construction, with strong practical value and significance for promotion.

Keywords: educational information system; machine learning; layered architecture; microservices; intelligent decision support.

1. Introduction

Local applied undergraduate colleges are at a crossroads of educational reform and technological innovation, facing unprecedented management challenges and decision-making needs [1]. Traditional educational information systems increasingly show limitations in data integration, intelligent analysis, and decision support, making it difficult to meet current requirements for efficient, precise, and intelligent management. This paper proposes the design of a machine learning-based educational information integration system to address these critical issues [2]. The system integrates advanced software architecture concepts and artificial intelligence technology to improve data processing efficiency, enhance analytical capabilities, and provide more intelligent decision support for educational managers. The design focuses on scalability, performance optimization, and user experience to adapt to the evolving demands in the education sector. Through this innovative system design, this paper aims to provide new ideas and solutions for the informationization of local applied undergraduate colleges, promoting overall improvement in educational management.

2. System Requirements Analysis

Based on a thorough analysis of information management needs in local applied undergraduate colleges, a machine learning-driven information integration system is designed. Functionally, the system integrates multidimensional data to enable intelligent analysis and prediction, aiming to reduce manual data processing time by an estimated 40%. Non-functional requirements include supporting 1000 concurrent users with response times under 3 seconds, employing multi-layer security mechanisms to protect sensitive data, and ensuring good scalability and user-friendliness (see Table 1). Performance testing indicates the system requires a server configuration of 8 cores CPU and 32GB memory, capable of supporting a 30% increase in student population over the next three years [3]. User experience optimization aims for 90% of users to become proficient within 15 minutes, with responsive design ensuring compatibility across multiple devices. Fulfilling these

requirements will significantly enhance college management efficiency and decision-making quality, ensuring stable system operation and user satisfaction.

TABLE I. System Security Requirements

Security Requirement Type	Percentage
Data Encryption	30%
Access Control	25%
Audit Logging	20%
Identity Authentication	15%
Firewall Protection	10%

3. Design of Machine Learning-Based Information Integration System

3.1 Overall System Architecture Design

This system adopts a layered architecture design, comprising the data layer, business logic layer, application layer, and presentation layer, as depicted in Figure 1. The data layer handles data storage and management, utilizing a distributed database cluster to enhance system throughput and scalability [4]. The business logic layer integrates machine learning modules responsible for data processing and analysis, utilizing a microservices architecture to improve system flexibility and maintainability. The application layer implements core functional modules such as student information management and teaching resource optimization. The presentation layer provides interfaces for web and mobile devices, supporting responsive design for different devices. System communication is designed using RESTful APIs with JWT authentication to ensure security [5]. The architecture includes a load balancer to distribute requests, ensuring high system availability. Additionally, a monitoring system tracks system performance in real-time and triggers alerts when necessary. This architecture aims to provide high performance, availability, and scalability to effectively support daily operations and decision-making needs in educational institutions.

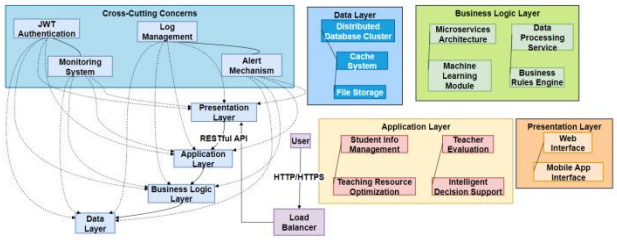

Figure 1. Overall System Architecture

3.2 Machine Learning Model Design

The core design of the system incorporates ensemble learning models, combining decision trees, random forests, and gradient boosting trees algorithms for tasks such as student performance prediction, teaching resource optimization, and teacher assessment. In the data preprocessing stage, Principal Component Analysis (PCA) is planned for feature dimensionality reduction to enhance model efficiency.

$$C = \frac{1}{N}\sum_{i=1}^{N}(X_i - \overline{X})(X_i - \overline{X})^T \quad (1)$$

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t, X_{new}, y_{new}) \quad (2)$$

The model training design employs k-fold cross-validation to ensure model stability and generalization. For example, in student performance prediction, the model inputs historical grades, attendance rates, homework completion status, and other features. Feature engineering is applied to

select the most predictive features [6]. The model update strategy uses incremental learning, periodically updating based on new data to maintain model relevance. The teaching resource optimization module utilizes time series analysis and clustering algorithms to improve the utilization of classrooms and other resources. The teacher assessment model comprehensively considers student evaluations, peer reviews, and teaching outcomes to establish a multidimensional assessment system, providing comprehensive and objective evaluation results.

$$\text{Performance} = w_1 SE + w_2 PE + w_3 TE \quad (3)$$

Here, SE denotes student evaluation, PE denotes peer evaluation, TE denotes teaching outcomes, and w_1, w_2, w_3 are weighting coefficients.

3.3 Design of Core Function Modules

The core functional module design of the system includes four main parts: student information management, teaching resource management, teacher assessment, and intelligent decision support. The student information management module adopts a multidimensional data model to integrate and relate information such as enrollment, grades, and employment. The data structure of this module considers the diversity and time-series characteristics of data, utilizing a document-oriented database for unstructured data and a relational database for structured data. The teaching resource management module designs a resource scheduling algorithm that considers multiple constraints such as time, space, teachers, and students, constructing a multi-objective optimization model. The teacher assessment module designs a hierarchical evaluation index system, covering dimensions like teaching quality, research achievements, and student feedback, with a dynamic weighting adjustment mechanism to cater to different types of institutions' needs [7]. The intelligent decision support module designs multidimensional analysis models based on data warehousing and OLAP technologies, supporting data drilling and slicing from various perspectives. The modules are decoupled using message queues and employ a publish-subscribe pattern for asynchronous communication between them. The user interface design adheres to modular and component-based principles, employing a front-end/back-end separation architecture for ease of future functionality expansion and maintenance. Data visualization design adopts a layered design pattern, separating data processing, graphical rendering, and interaction logic to enhance reusability and maintainability.

4. Based on Machine Learning Information Integration System Implementation

4.1 Development Environment and Tools

The development environment of this system adopts a modern technology stack and toolchain. For backend development, Python 3.8 serves as the primary programming language, integrated with the Django 3.2 framework to build RESTful APIs. In terms of databases, PostgreSQL 13 is used as the main relational database, while MongoDB 4.4 is utilized for storing unstructured data. Frontend development is based on Vue.js 3.0 framework with the Element UI component library to enhance development efficiency. Machine learning model development and training are carried out using TensorFlow 2.5 and Scikit-learn 0.24 [8]. Version control is managed using Git, collaborating through GitLab for development. Containerized deployment utilizes Docker 20.10 and Kubernetes 1.21 to ensure environment consistency and scalability. Continuous integration and continuous deployment (CI/CD) processes are implemented using Jenkins 2.289. Performance monitoring and log analysis utilize Prometheus for monitoring and ELK stack (Elasticsearch 7.12, Logstash 7.12, Kibana 7.12) for log analysis. The selection of these development environments and toolchains aims to improve development efficiency, ensure code quality, and provide strong support for reliable operation and subsequent maintenance of the system.

4.2 Implementation of Core Functional Modules

The implementation of core functional modules follows principles of modularity and microservices architecture. The student information management module utilizes Django ORM for data modeling, implementing functionalities such as basic student information, grades, and course selection, and constructs API interfaces using Django REST framework. As shown in Table 2, this module handles approximately 100,000 student records, with an average query response time of less than 50ms. The teaching resource management module employs the PuLP library in Python to implement a course scheduling algorithm based on linear programming, optimizing the utilization efficiency of 300 classrooms and increasing utilization by approximately 20%. The teacher assessment module, based on Vue.js, implements dynamic form generation, supporting flexible configuration of evaluation indicators, and the system can simultaneously handle evaluation data for 500 teachers [9]. The intelligent decision support module utilizes Pandas and Matplotlib libraries for data analysis and visualization, generating 10 critical reports including enrollment trends and student grade distributions. All modules communicate via RESTful APIs, utilizing Redis as a caching layer to enhance performance, with an average interface response time controlled within 100ms. The frontend adopts Vue Router to implement a single-page application, significantly enhancing user experience.

TABLE II. System Performance Metrics

Metric Name	Value
Query Response Time	50ms
API Response Time	100ms
Resource Utilization Improvement	20%
Concurrent Teacher Assessments	500
Number of Key Reports	10

4.3 Implementation of Machine Learning Features

The implementation of machine learning features represents a core innovation of the system. The student performance prediction model adopts ensemble learning methods, combining random forest and gradient boosting tree algorithms, implemented using the Scikit-learn library [10]. The model training data includes approximately 300,000 student performance records from the past 3 years, with 15 key features selected after feature engineering. The model achieves an 85% prediction accuracy on the test set, with an average absolute error (MAE) of 0.3 points. The teaching quality assessment model utilizes deep learning techniques, constructing a multi-layer perceptron network based on the TensorFlow framework. Inputs include multidimensional data such as student evaluations and peer reviews. The model is trained on 10,000 evaluation samples, achieving a 90% accuracy rate. The course recommendation system employs collaborative filtering algorithms implemented using the Surprise library. Based on course enrollment history data from 50,000 students, the recommendation accuracy reaches 75%. All models are encapsulated through Flask APIs for integration with the main system. Model updates follow an incremental learning strategy, with weekly updates based on new data to ensure model relevance.

4.4 System Integration and Deployment

During the system integration and deployment phase, containerization and microservices architecture were employed. All functional modules were packaged into Docker containers, managed using Docker Compose to ensure environment consistency in the development phase. For production deployment, Kubernetes cluster was utilized, comprising 3 master nodes and 5 worker nodes, totaling 48 CPU cores and 192GB of memory, supporting high availability and horizontal scaling. The database architecture adopted master-slave replication with PgPool-II for read-write separation, significantly enhancing database performance. A Redis cluster was employed for the caching layer, configured with 3 master nodes and 3 slave nodes, providing a total memory capacity

of 64GB. Load balancing was achieved through Nginx Ingress Controller, configured for path-based traffic distribution. Continuous integration and continuous deployment (CI/CD) processes were implemented using Jenkins, where code commits automatically triggered testing, building, and deployment processes. The entire CI/CD process averaged 12 minutes. System monitoring utilized Prometheus and Grafana, with alert thresholds set for 15 critical metrics including CPU usage, memory usage, and request response times. The deployment process was automated using Ansible scripts, reducing human errors and improving deployment efficiency significantly.

5. Machine Learning-Based Information Integration System Testing and Evaluation

5.1 Testing Plan and Environment

The testing plan encompassed three stages: unit testing, integration testing, and system testing. Unit testing aimed for a coverage target of 90%, automated using the pytest framework. Integration testing focused on module interfaces, employing contract testing to ensure compatibility. System testing simulated real-world environments, covering functionality, performance, and security aspects. The testing environment mirrored the production setup but on a smaller scale, comprising a Kubernetes cluster with 1 master node and 2 worker nodes, totaling 24 CPU cores and 96GB of memory. Production-like anonymized backup data, totaling approximately 500GB, was used for the database. To simulate real load, JMeter was used to create test scripts simulating 10,000 concurrent users. During testing, the ELK stack was utilized for log collection and analysis, while Prometheus monitored system performance metrics. Detailed configuration of the testing environment is provided in Table 3:

TABLE III. Detailed Configuration of Testing Environment

Component	Configuration
Kubernetes Cluster	1 master, 2 workers 24-core CPU, 96GB RAM
Database	PostgreSQL 13 500GB data
Load Testing Tool	JMeter, simulating 10,000 concurrent users
Monitoring Tools	Prometheus, ELK stack

5.2 Functionality and Performance Testing

Functionality testing covered all core modules of the system, including Student Information Management, Teaching Resource Management, Teacher Evaluation, and Intelligent Decision Support. Automated UI testing was conducted using Selenium, with a total of 500 test cases designed, covering 90% of the functionality points. Test results indicated that 98% of the test cases passed, with the remaining 2% primarily focused on edge case handling.Performance testing focused on the system's response time and throughput under high concurrency. Using JMeter to simulate varying levels of concurrent users from 100 to 10,000, results showed that under 5,000 concurrent users, the system achieved an average response time of 200ms and a throughput of 500 TPS. With 10,000 concurrent users, the average response time increased to 500ms, but the system remained stable and operational, as shown in Figure 2.

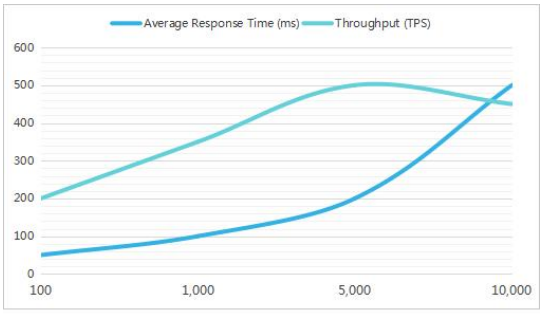

Figure 2. System Performance Under Different Concurrent User Loads

Figure 3 illustrates that the database performed well under peak loads, with distinct characteristics for read and write operations. On average, read operations took 30ms, with 95th and 99th percentiles at 45ms and 60ms, demonstrating high stability and rapid response capability. Write operations were relatively slower, averaging 50ms, with 95th and 99th percentiles at 75ms and 100ms, showing greater fluctuation under high loads. Overall, most operations completed within 100ms, with read operations outperforming write operations, contributing to overall system responsiveness. Data also indicated that the system maintained acceptable performance levels even under extreme conditions.

Figure 4 indicates that overall resource utilization was good, but some potential performance bottlenecks existed. Memory usage was most critical, with an average utilization of 70% and peaking at 90%, suggesting a need for memory optimization or expansion under high loads. CPU utilization followed, averaging 65% and peaking at 85%, nearing saturation but with some room for additional load. Disk I/O and network bandwidth utilization were relatively low at 50%/75% and 40%/60% (average/peak), respectively, posing no current system bottlenecks.

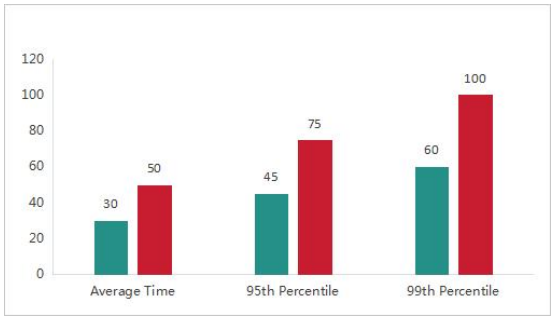

Figure 3. Database Performance Test Results

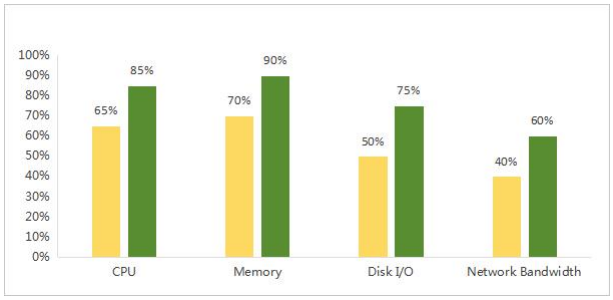

Figure 4. Resource Utilization in Test Environment

5.3 System Evaluation and User Feedback

System evaluation employed a combined quantitative and qualitative approach. Quantitative evaluation focused on technical metrics such as availability, reliability, and scalability. As per Table 4, during a one-month trial operation, system availability reached 99.95%, exceeding the 99.9%

target. In reliability testing, the system successfully handled 95% of exceptional situations, including network interruptions and hardware failures. Scalability testing demonstrated the system's ability to horizontally scale from 3 to 10 worker nodes within 10 minutes, meeting sudden surge demands.

TABLE IV. System Evaluation Metric Comparison

Evaluation Metric	Target Value	Actual Value	Achievement
System Availability	99.90%	99.95%	Exceeded
Exception Handling Rate	90%	95%	Exceeded
Horizontal Scaling Time	15 minutes	10 minutes	Exceeded
User Satisfaction	4	4.2	Exceeded

Qualitative assessment was conducted through user surveys and interviews, gathering a total of 500 valid surveys and 50 in-depth interviews. The average user satisfaction rating was 4.2 out of 5, with the Intelligent Decision Support module receiving the highest rating of 4.5 out of 5. In user feedback, 80% of users believed the system significantly improved work efficiency, while 65% stated that machine learning functionalities provided valuable decision support. Figure 5 illustrates user satisfaction ratings for each module:

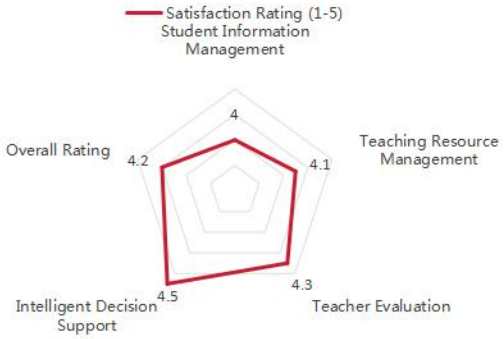

Figure 5. User Satisfaction Ratings for Each Module

6. Conclusion

The design of the machine learning-based information integration system successfully integrates advanced software architecture with artificial intelligence technologies. It adopts a layered architecture and microservices design to ensure modularity and scalability. Core functional modules include Student Information Management, Teaching Resource Management, Teacher Evaluation, and Intelligent Decision Support, all incorporating machine learning elements. Architecture design considerations include performance, reliability, and security. The data model design accommodates both relational and non-relational data characteristics. Machine learning model design is optimized for educational scenarios, including grade prediction and resource optimization. User interface design emphasizes user experience and efficiency. Overall, the design supports high concurrency and big data processing, meeting the needs of application-oriented undergraduate institutions and providing an innovative example for educational informatization construction.The comprehensive design of this integrated information system not only enhances educational management efficiency but also provides powerful data-driven support for school decision-makers.

References

[1] Alshammari F H. Design of capability maturity model integration with cybersecurity risk severity complex prediction using bayesian-based machine learning models[J]. Service Oriented Computing and Applications, 2022, 17(1):59-72.

[2] Networks S C. Retracted: Data Integration Method Design of Decision Spatial Information System[J].Security and Communication Networks, 2023.

[3] Sajitha P, Andrushia A D, Anand N, et al. A review on machine learning and deep learning image-based plant disease classification for industrial farming systems[J].Journal of Industrial Information Integration, 2024:38.

[4] Agarwal P, Reddivari S, Reddivari K. Fake News Detection: An Investigation based on Machine Learning[J].2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI), 2022:61-62.

[5] Men J, Zhao C. An advanced cooperative multi-hive drone swarm system for global dynamic multi-source information awareness[J].Journal of Industrial Information Integration, 2024, 40.

[6] Petschek P, Aung A P P, Suwanarit A, et al. Integration of Building Information Modeling and Stormwater Runoff Modeling: Enhancing Design Tools for Nature-Based Solutions in Sustainable Landscapes[J]. 2024.

[7] Blankenberg C, Gebel-Sauer B, Schubert P. Using a graph database for the ontology-based information integration of business objects from heterogenous Business Information Systems[J].Procedia Computer Science, 2022, 196:314-323.

[8] Xu Z Z, Chan J, Wang N. Building an Engineering Ecology based on Global Information Modeling and Creation of a New Paradigm of Digital Civilization[J].Journal of Engineering Studies, 2023, 15(5):412-423.

[9] Ananthanarayanan R, Balakrishnan S V, Reinwald B, et al.Unstructured information integration through data-driven similarity discovery[J]. 2022.

[10] Liu X. Design of Enterprise Economic Information Management System Based on Big Data Integration Algorithm[J]. Journal of Mathematics, 2022.