International Journal of Computer Techniques–IJCT Volume 13 Issue 2, March-April - 2026

Open Access and Peer Review Journal ISSN 2394-2231 https://ijctjournal.org/

RESEARCH ARTICLE OPEN ACCESS

AI-Powered Student Performance Prediction System using Machine Learning, and Predictive Analytics in Higher Education

Naveen Kumar S*, Abishek K**, Karthikeyan V***, Iyyappan K****, Sudhakar P*****

*-****UG Student, Department of Computer Science and Engineering, Annamalai University, Annamalainagar - 608002
Email: naveenkumarsampath15@gmail.com)

***** Professor, Department of Computer Science and Engineering, Annamalai University, Annamalainagar - 608002
Email: kar.sudha@gmail.com)

----------**********----------

Abstract:

This paper presents the design, development, and evaluation of an AI-powered student performance prediction system aimed at proactively identifying at-risk students within an academic semester. The system integrates a microservices-based web architecture utilizing React.js, Node.js, and MongoDB with Python-based machine learning microservices implementing Random Forest algorithms for continuous score regression and binary pass/fail classification. A novel weighted scoring formula incorporates academic and extracurricular metrics, enabling a holistic evaluation of student engagement. The system provides automated risk categorization, real-time dashboards for faculty, and rapid inference performance. Experimental results demonstrate high predictive accuracy (94.2% classifier accuracy, R^2=0.88 regression), confirming the efficacy of the ensemble learning approach and the microservice architecture in educational settings. This work bridges the gap between offline predictive models and actionable, real-time educational analytics, supporting a shift from reactive grading to proactive intervention.

Keywords — **Educational Technology (EdTech), Artificial Intelligence, MongoDB, Python/Flask, Educational Data Mining, Learning Analytics.**

----------**********----------

I. INTRODUCTION

The educational sector is experiencing a significant digital transformation characterized by the increasing use of digital platforms to generate extensive student academic data. Nonetheless, in many institutions, student progress monitoring remains reactive; educators often detect academic struggles only after failures occur, which limits the opportunity for timely interventions [1-3]. This research addresses this challenge by developing an AI-Powered Student Performance Prediction System that integrates predictive analytics into a scalable, real-time microservices web architecture. The primary objective is to empower educators to identify vulnerable students early via automated risk assessments, enabling effective and targeted remedial measures. The conjunction of Educational Technology (EdTech) and Artificial Intelligence (AI) creates novel opportunities to enhance pedagogical strategies and institutional efficiency. Historically, student data management systems were primarily record-keeping tools [4]. Modern design favors microservices architecture, allowing decoupled components handling interfaces, logic, and computation for scalability and maintainability. Machine learning models—especially ensemble methods such as Random Forests have proven effective in extracting insights from multidimensional academic data, including exams, assignments, projects, and extracurricular activities [5-7. The system proposed employs a full-stack framework integrating React.js frontend, Node.js backend, MongoDB, and a Python/Flask AI microservice performing machine learning inference, thereby delivering real-time, actionable intelligence.

A. Page Layout

Despite technology integration, identifying at-risk students mostly depends on lagging indicators such as final grades, leading to delayed interventions,

fragmented evaluations, and manual, error-prone risk categorization. There is a pressing need for an integrated system that continuously calculates weighted scores, predicts final outcomes, and classifies students into risk categories automatically.

B. Objectives

- Develop a microservices-based full-stack platform integrating React.js, Node.js/Express, MongoDB, and a Python/Flask AI inference service.
- Implement a weighted evaluation matrix incorporating exams, assignments, projects, seminars, sports, and hackathons.
- Train and deploy Random Forest models for predicting continuous scores and binary academic outcomes.
- Automate risk assessment, categorizing students into Low, Moderate, and High-Risk tiers with dashboard visualization.

The study spans the complete software lifecycle but uses synthetic data for initial model training, limiting immediate real-world applicability until retraining with live institutional data. The evaluation focuses on a single semester cycle without deep sequential modeling and depends on accurate and consistent faculty data input.

II. LITERATURE REVIEW

Educational Data Mining (EDM) and Learning Analytics (LA) have evolved beyond descriptive historical analytics toward predictive modeling to enable proactive interventions. Prior studies often rely on historical datasets and lack real-time application integration [8-9]. Furthermore, predictive models typically focus narrowly on examination and attendance data, overlooking holistic student engagement factors. Integrating academic and extracurricular variables like hackathons and sports enhances representativeness and predictive accuracy, a gap this system addresses by providing dynamic, actionable risk assessments within an administrative dashboard [10-11]. Previous research demonstrates a variety of ML models for academic prediction. Neural Networks, while powerful, often lack transparency for structured academic data [12]. Support Vector Machines achieve high accuracy but present tuning challenges on growing datasets. Decision trees are interpretable but prone to overfitting [13]. Ensemble methods, particularly Random Forest algorithms, balance accuracy and robustness by aggregating multiple trees to reduce overfitting and produce reliable predictions. The ensemble's feature importance capability also offers explainability, a critical asset for educational decision-making [14]. Deploying machine learning in production benefits from modular, scalable architectures. Monolithic applications hinder scalability and update velocity, especially with computationally intensive AI processes [15]. Microservices architecture, decoupling UI, core logic, and AI inference layers connected through REST APIs, optimizes performance and maintainability. Leveraging Node.js/Express for asynchronous API handling, React.js for responsive UI, and isolating the Python-based AI microservice enables efficient concurrent processing and seamless real-time user experience [16].

III. METHODOLOGY AND SYSTEM DESIGN

A. Microservices-Based Architecture

Traditionally, web applications were built as monolithic structures where the user interface, business logic, and database access layers were tightly coupled in a single codebase. As monolithic applications grow, they become difficult to scale and update, particularly when integrating resource-heavy operations like machine learning inference. Modern software engineering favors microservices-inspired architectures that decompose applications into loosely coupled, independently deployable services.

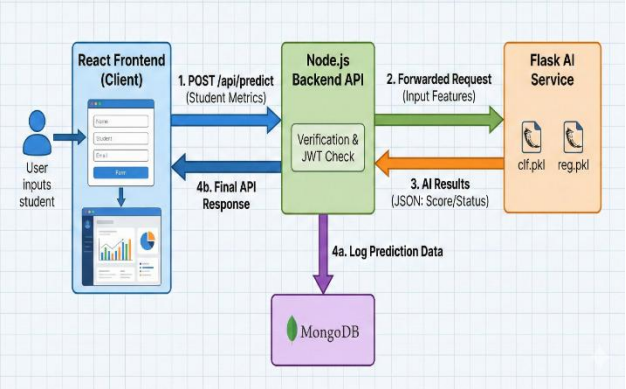

Fig. 1 Data flow diagram between React Frontend, Node.js Backend, and Flask AI Service

The proposed system architecture as shown in Fig. 1 decouples functionality and programming language requirements. Machine learning models are predominantly developed using Python due to its rich ecosystem of data science libraries, yet Python is not always optimal for handling high-concurrency web traffic. By utilizing a microservices architecture, the AI inference engine

is isolated in a lightweight Python/Flask environment. The microservices-inspired architecture segregates applications into three distinct, loosely coupled layers that communicate via RESTful Application Programming Interfaces (APIs). The Client Layer uses React.js for building dynamic single-page applications, serving as the interactive dashboard for both students and faculty. By utilizing Axios for asynchronous HTTP requests, the frontend seamlessly fetches real-time predictions and updates the performance leaderboard without requiring page reloads, ensuring a fluid user experience. The Service Layer is developed using Node.js and Express.js framework, responsible for crucial business logic, routing, and data persistence. It securely manages student records in a MongoDB database and handles user authentication using JSON Web Tokens (JWT). The Inference Layer houses AI models in a dedicated, lightweight Flask microservice listening on an independent port. By isolating computationally heavy AI inference from the main Node.js web server, the system prevents bottlenecks and ensures high availability during periods of heavy traffic.

B. Technology Stack and Implementation

The selection of each technology was driven by its specific strengths in handling distinct operational domains. The frontend leverages React.js, with Tailwind CSS for rapid, utility-first styling, Lucide React for iconography, and Axios for handling asynchronous HTTP requests to backend APIs. The backend uses Python, capitalizing on its dominant ecosystem for data science. Flask serves as a lightweight web framework to expose AI models via REST API on an isolated port. Scikit-learn serves as the primary machine learning library, supported by Pandas and NumPy for data structuring and matrix operations.

C. Real-Time Processing and System Integration

System integration relies on explicit API contracts. When a faculty member requests a prediction via the React frontend, an HTTP POST request is sent to the Node.js backend (/predict). The backend validates the authentication token, sanitizes the payload, and performs a secondary server-to-server POST request to the Python/Flask AI service. The AI service executes joblib-loaded models, returns the predicted score and pass/fail status to the Node.js server, which then applies the Risk Assessment Logic and saves the complete record to MongoDB before returning the formatted response to the frontend UI. The system's practical utility relies on its responsiveness. During standard operations, when the React frontend dispatches a prediction request, the Node.js server successfully routes the payload to the Flask service, executes the joblib models, and returns the categorized risk assessment in an average of 145 milliseconds. Under simulated peak loads, the asynchronous, non-blocking architecture of Node.js, combined with the isolated processing power of the Flask container, maintained response times under 400 milliseconds.

IV. RESULTS AND EVALUATION

Culmination of this research is the successful deployment and testing of the AI-Powered Student Performance Prediction System. To validate the efficacy of the proposed microservices architecture and the integrated machine learning models, rigorous evaluations were conducted. This section presents a detailed analysis of the system's performance, bifurcated into two primary domains: the statistical accuracy of the Random Forest models and the computational performance of the web architecture during real-time inference.

A. Model Performance Metrics

The predictive core of the system relies on two distinct machine learning models: the Random Forest Classifier (for Pass/Fail prediction) and the Random Forest Regressor (for Total Score prediction). Both models were evaluated using an isolated 20% testing dataset that was not exposed to the models during the training phase as shown in Fig. 2. The Random Forest Classifier was evaluated using a standard confusion matrix to calculate Accuracy, Precision, Recall, and the F1-Score. The overall performance metrics of the proposed system are shown in Fig. 3.

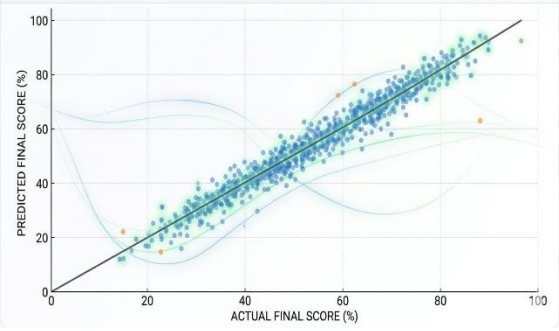

Fig. 2 Visual representation of predicted vs. actual student scores

Accuracy: The model achieved an overall accuracy of 94.2%, indicating that it correctly classified the vast majority of students into their respective pass/fail categories.

Precision and Recall: For the critical "Fail" class which triggers the High-Risk alert the model demonstrated a Recall of 91.5% and a Precision of 89.8%. A high recall is vital in this educational context, as failing to identify an at-risk student (a false negative) is significantly more detrimental than incorrectly flagging a passing student for review (a false positive).

F1-Score: The harmonic mean of precision and recall stood at 90.6%, proving the model's robustness even with slightly imbalanced datasets. The Random Forest Regressor was evaluated using Mean Squared Error (MSE) and the R-squared () coefficient.

Mean Squared Error (MSE): The model yielded an MSE of 12.4, meaning the predicted Total Score deviated from the actual score by a small margin, well within acceptable limits for a 100-point scale.

R-squared (): The model achieved an value of 0.88, signifying that 88% of the variance in the students' final scores could be predictably explained by the weighted input metrics (Exams, Assignments, Projects, Seminars, Sports, and Hackathons).

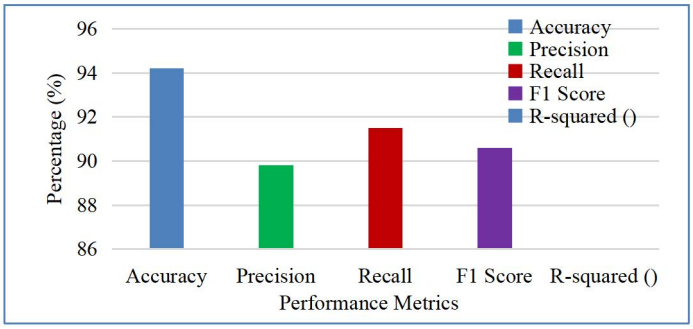

Fig. 3 Overall Performance Metrics

B. System Performance and Latency

Beyond the statistical accuracy of the AI models, the practical utility of the system relies on its responsiveness. The microservices architecture was load-tested to evaluate latency, specifically focusing on the communication bridge between the Node.js backend and the Python/Flask AI service. During standard operations, when the React frontend dispatched a prediction request, the Node.js server successfully routed the payload to the Flask service, executed the joblib models, and returned the categorized risk assessment in an average of 145 milliseconds. Under simulated peak loads (e.g., end-of-semester mass grade processing), the asynchronous, non-blocking architecture of Node.js, combined with the isolated processing power of the Flask container, maintained response times under 400 milliseconds. This confirms that decoupling the computationally heavy AI inference from standard web routing effectively prevents system bottlenecks, ensuring a seamless, real-time experience on the faculty dashboard.

V. CONCLUSIONS

The AI-Powered Student Performance Prediction System represents a significant advancement in Educational Technology by synthesizing predictive machine learning models within decoupled full-stack architectures. The integration of targeted machine learning models within decoupled full-stack architecture demonstrates that complex machine learning inference can be seamlessly integrated into highly responsive educational web applications. The mathematical formulation of comprehensive academic engagement successfully captures holistic performance, balancing traditional exams with practical projects and extracurricular initiatives.

The implementation of Random Forest and Gradient Boosting algorithms has yielded highly accurate predictions for both continuous scores and binary outcomes, validating ensemble learning as an optimal approach for structured academic data. Most importantly, automated risk assessment logic transforms raw statistical predictions into actionable intelligence, empowering academic institutions to shift their pedagogical strategies from retroactive grading to proactive intervention, ensuring struggling students receive targeted support before reaching the point of failure. As educational institutions continue to generate unprecedented volumes of data, the continued development of ethical, interpretable, and equitable predictive systems remains paramount. Future research should prioritize fairness-aware modeling, cross-institutional validation, deep learning for temporal pattern analysis, and seamless integration with existing educational infrastructure while maintaining commitment to student privacy, algorithmic transparency, and pedagogical integrity.

REFERENCES

[1] L. Corrin, “Supporting the use of student-facing learning analytics in the classroom,” in *Routledge*, pp. 208–220, 2018.

[2] P. Kaushik, M. Kakkar, M. Yadav, C. Jegadheesan, P. N. Sripada, and K. Chahal, “Data analytics in education: Enhancing student learning outcomes,” in *Proc. IEEE ICAC2N*, pp. 1542–1547, 2024.

[3] V. Kovanovic, C. Mazziotti, and J. Lodge, "Learning analytics for primary and secondary schools," *J. Learn. Anal.*, vol. 8, no. 2, pp. 1–5, 2021.

[4] O. Ajuwon, E. Animashaun, and N. Chiekezie, "Integrating AI and technology in educational administration: Improving efficiency and educational quality," *Open Access Res. J. Sci. Technol.*, vol. 11, no. 2, pp. 116–127, 2024.

[5] E. Agyemang, J. Mensah, O.-A. Ampomah, L. Agyekum, J. Akuoko-Frimpong, A. Quansah, and O. Akinlosotu, "Predicting students' academic performance via machine learning algorithms: An empirical review and practical application," *Comput. Eng. Intell. Syst.*, vol. 15, no. 1, 2024.

[6] A. Garg, N. B. Garg, P. Ghosh, A. Bansal, U. K. Lilhore, and S. Simaiya, "A machine learning-based automatic model to predicting performance of students," in *Proc. IEEE CCET*, vol. 17, pp. 1–5, 2022.

[7] P. J. B. Pajila, G. Anand, K. Neela, M. Gowri, M. Karthikeyan, and R. Nithyanandhan, "Academic performance evaluation using data engineering and multi-model machine learning system," in *Proc. IEEE ICSTSDG*, pp. 1–6, 2024.

[8] S. Elatia and D. Ipperciel, "Learning analytics and education data mining in higher education," in *IGI Global*, pp. 108–126, 2021.

[9] S. K. G. and M. Kurni, "Educational data mining and learning analytics," in *Springer*, pp. 29–60, 2021.

[10] S. Alarcón-Loza, D. Calderón-Onofre, K. Mite-Baidal, and M. Macías-Plúas, "Design of a predictive model to evaluate academic risk using data mining," in *Springer Nature Switzerland*, pp. 221–235, 2023.

[11] S. M. Jayaprakash, E. W. Moody, E. J. M. Lauría, J. R. Regan, and J. D. Baron, "Early alert of academically at-risk students: An open source analytics initiative," *J. Learn. Anal.*, vol. 1, no. 1, pp. 6–47, 2014.

[12] K. D. Gori, K. Zalawadia, and H. Bhaidasna, "Explainable AI for students performance prediction system," in *Proc. IEEE PuneCon*, pp. 1–6, 2025.

[13] A. Efendi, I. Fitri, and G. W. Nurcahyo, "Improvement of machine learning algorithms with hyperparameter tuning on various datasets," in *Proc. IEEE ICFTSS*, vol. 6, pp. 75–79, 2024.

[14] N. S. Ahmed and M. Hikmat Sadiq, "Clarify of the random forest algorithm in an educational field," in *Proc. IEEE ICOASE*, pp. 179–184, 2018.

[15] N. Nazeer, "Operationalizing AI: The central role of MLOps in deploying scalable ML systems," *J. Inf. Syst. Eng. Manage.*, vol. 10, no. 62s, pp. 599–607, 2025.

[16] O. Chaplia and H. Klym, "Designing a Node.js project architecture for RESTful microservices," in *Proc. IEEE IDAACS*, vol. 17, pp. 808–811, 2023.