International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

# Optimizing Mobile App Recommendations Using Crowdsourced Educational Data

**Mr Manik Rao Patil[1],Sk Rafeeq[2],S Ajay Reddy[3],D Sugunakar[4]**

[1]Assistant Professor; Dept. of Information Technology, Guru Nanak Institutions Technical Campus, Hyderabad, India.

[2,3,4]B.Tech Students, Dept. of Information Technology, Guru Nanak Institutions Technical Campus, Hyderabad, India. Mail Id; Skrafee77@Gmail.Com[2]

***Abstract***

*In the modern digital landscape, personalized recommendation systems have become essential for improving students' learning experiences. The widespread use of mobile devices enables the collection of valuable app usage data, which can be utilized to deliver customized educational app suggestions. This study presents a recommendation system designed for university students at different academic levels, including Undergraduate (UG), Postgraduate (PG), and Graduate learners, by analyzing their application usage patterns. To ensure effective data representation, Natural Language Processing (NLP) techniques such as text cleaning, tokenization, and feature extraction are applied to process app descriptions and user interaction data. These techniques help in identifying meaningful patterns and relevant features. The Random Forest Classifier is adopted as the primary model due to its robustness, capability to manage high-dimensional data, and strong classification performance. The proposed system effectively identifies student preferences and recommends applications that align with their academic requirements. Experimental evaluation demonstrates that the model provides accurate, scalable, and interpretable recommendations, outperforming traditional approaches. Additionally, the system addresses challenges such as data sparsity while enhancing personalization and overall user satisfaction in educational app recommendation systems.*

***Keywords:*** *Personalized Recommendation System, Educational Apps, Student Learning, Mobile Usage Data, Natural Language Processing (NLP), Text Processing, Feature Extraction, Random Forest Classifier, Machine Learning, User Behavior Analysis, App Recommendation, Data Sparsity, Personalization, Predictive Modeling, Student Preferences*

## Introduction

In the current era of digital education, personalized recommendation systems have become increasingly important in enhancing the learning experience of students. With the rapid growth in the use of smartphones and tablets, students now depend heavily on mobile applications for academic learning, productivity, research, and skill development. These educational applications serve multiple purposes, such as subject understanding, time management, collaboration, and exam preparation. However, the vast number of available applications often makes it difficult for students to identify those that best match their academic requirements and personal learning preferences. To overcome this challenge, intelligent recommendation systems can be utilized to guide students toward suitable and effective applications.

This research focuses on developing a machine learning–based educational app recommendation system tailored for university students across different academic levels, namely Undergraduate (UG), Postgraduate (PG), and Graduate students. The system makes use of both app usage data and user interaction patterns to generate personalized recommendations. Natural Language Processing (NLP) techniques are applied to preprocess and analyze textual data such as app descriptions. Methods including text cleaning, tokenization, stop-word removal, and feature extraction help convert unstructured information into meaningful representations. These processed features enable the system to better understand both application functionalities and student needs.

To perform classification and prediction, the Random Forest Classifier is used as the core model due to its robustness and effectiveness in handling large and complex datasets. Its ensemble learning approach improves prediction accuracy while reducing overfitting, making it suitable for real-world educational data analysis. The model classifies users based on their profiles and preferences, allowing it to generate accurate and context-aware recommendations. Experimental results indicate that this approach outperforms traditional recommendation techniques in terms of both accuracy and interpretability. Overall, the system provides scalable, efficient, and user-friendly recommendations that align with students' academic goals and contribute to enhanced engagement in digital learning environments.

## Scope of the Project

The scope of this project involves the development of a personalized educational app recommendation system using machine learning and NLP techniques.

ISSN 2347–3657

Volume 14, Issue 1, 2026

The system is designed to analyze both user behavior and textual data derived from app descriptions to identify learning preferences among students at different academic levels. By applying NLP methods, relevant features are extracted from unstructured text, enabling the creation of meaningful data representations. The Random Forest Classifier is employed to classify and recommend suitable applications with high accuracy. The model is designed to be scalable and adaptable, making it applicable across various academic disciplines and educational platforms. Additionally, the study explores how personalized recommendations can improve user satisfaction, engagement, and learning outcomes. It also addresses common challenges such as data sparsity and irrelevant suggestions found in traditional recommendation systems. The system is intended to be flexible, interpretable, and easily deployable in mobile or web-based environments, contributing to improved accessibility and efficiency in digital education.

## Objectives

The primary objective of this research is to design and implement a machine learning-based system that provides personalized educational app recommendations for university students. The study aims to classify students into Undergraduate, Postgraduate, and Graduate levels based on their app usage patterns and learning behavior. It also focuses on applying NLP techniques to preprocess and extract meaningful features from textual data, ensuring accurate representation of application content. Another key objective is to utilize the Random Forest Classifier due to its strong performance, interpretability, and robustness in classification tasks. The research evaluates the model using performance metrics such as accuracy, precision, and scalability in real-world datasets. Additionally, it seeks to address common issues like data sparsity and the cold-start problem that often affect recommendation quality. The system is designed to enhance user engagement by aligning recommendations with students' academic interests and goals. Furthermore, the study explores the impact of personalized recommendations on learning motivation and digital resource usage. It also aims to assist educational institutions in integrating intelligent recommendation technologies into their platforms. Finally, the research contributes to the advancement of AI-driven education systems by emphasizing transparency, efficiency, and adaptability.

## Existing System

The existing system is based on the Gated Recurrent Unit (GRU), a type of recurrent neural network designed to model sequential data by capturing temporal dependencies. GRU is particularly effective in analyzing user interaction patterns over time, making it suitable for generating personalized app recommendations. By maintaining relevant past information and adapting to changes in user behavior, GRU can provide dynamic and time-sensitive suggestions. This capability is especially useful in educational environments where user preferences evolve continuously. However, despite its advantages, the GRU-based approach has several limitations. It requires high computational resources, especially when dealing with large datasets, and involves longer training times. Additionally, the model may suffer from overfitting when trained on limited data and is often difficult to interpret due to its black-box nature. It is also less efficient when handling sparse or noisy data, which can affect recommendation quality.

## Literature Survey

Recent studies have explored various approaches to mobile app recommendation systems. The MobileRec dataset introduced by Maqbool et al. provides a large-scale collection of user–app interactions along with app metadata, demonstrating the importance of combining textual features with behavioral data to improve recommendation accuracy. Similarly, the SAppKG framework proposed by Dave et al. utilizes knowledge graphs and side information to generate secure and personalized recommendations, emphasizing privacy and improved prediction performance. Another study by Badier et al. focuses on developing a recommendation model for after-school educational applications, showing that personalized suggestions significantly enhance student engagement and satisfaction. Barnes highlights the effectiveness of the Random Forest Classifier in higher education, particularly in predicting academic outcomes, due to its ability to handle diverse data types and provide interpretability through feature importance. Furthermore, Islam proposes a multi-model machine learning framework that integrates NLP and ensemble techniques to improve recommendation accuracy and address data sparsity issues. These studies collectively demonstrate the growing importance of combining machine learning and NLP techniques in educational recommendation systems.

## Proposed System

The proposed system aims to provide personalized educational app recommendations by analyzing both user behavior and textual data. Initially, data preprocessing is performed using NLP techniques such as text cleaning, tokenization, stop-word removal, and vectorization. These steps transform unstructured textual information into structured numerical features that capture the semantic meaning of app descriptions. By combining these features with app usage patterns, the system creates a comprehensive dataset that reflects student preferences across different academic levels.

**Project Description**
**General**

The proposed project aims to develop a personalized educational application recommendation system for university students by integrating machine learning techniques with Natural Language Processing (NLP). With the rapid expansion of mobile-based learning platforms, students often encounter difficulty in selecting applications that best match their academic requirements and learning preferences. This system addresses the problem by analyzing user interaction data, app usage patterns, and textual descriptions of applications to generate accurate and context-aware recommendations. The dataset is gathered from various educational platforms and includes important attributes such as application name, category, description, ratings, and user engagement details.

To transform unstructured textual data into a usable format, NLP techniques including text cleaning, tokenization, and feature extraction are applied. These preprocessing steps enable the conversion of descriptive content into structured numerical representations suitable for machine learning models. The core algorithm used in this system is the Random Forest Classifier, selected for its reliability, scalability, and strong performance in handling high-dimensional datasets. The model classifies students into categories such as Undergraduate (UG), Postgraduate (PG), and Graduate levels based on their behavioral patterns and preferences, and accordingly recommends relevant applications.

In addition to improving recommendation accuracy, the system effectively handles challenges such as data sparsity and irrelevant suggestions commonly found in traditional approaches. Model performance is evaluated using standard metrics, including accuracy, precision, recall, and F1-score, to ensure robustness and consistency. Once trained, the model is stored and deployed for real-time predictions, allowing seamless integration into web or mobile-based platforms. Overall, this project provides an interpretable, scalable, and efficient solution that enhances student engagement with digital learning tools. By combining behavioral insights with content-based analysis, the system ensures personalized and meaningful recommendations. Furthermore, it demonstrates the potential of machine learning in improving educational outcomes and simplifying the process of discovering relevant academic resources in a data-driven manner.

**Methodologies**

The methodology adopted in this project follows a systematic pipeline that includes data collection, dataset construction, data preparation, model selection, prediction, evaluation, and deployment. Initially, data is collected from multiple educational applications used by university students across different academic levels. This data includes user interaction logs, app usage behavior, demographic information, and textual descriptions of applications. Care is taken to ensure that the data collection process adheres to ethical standards, including user consent and privacy protection.

The collected data is then organized into a structured dataset containing both numerical and textual features. These features include app-related information such as name, category, description, ratings, and user engagement metrics. Data cleaning is performed to remove inconsistencies, missing values, and irrelevant entries, ensuring high-quality input for model training. Following this, data preparation is carried out using NLP techniques, where textual content is processed through steps such as tokenization, stop-word removal, and feature extraction. Categorical variables are encoded, and the dataset is normalized to maintain uniformity across features. The prepared dataset is then divided into training and testing subsets to evaluate model performance.

For model selection, the Random Forest Classifier is chosen due to its ability to handle mixed data types, reduce overfitting, and provide interpretable results through feature importance analysis. Hyperparameters such as the number of decision trees and maximum depth are optimized to achieve the best performance. Once trained, the model is used to analyze user data and predict suitable applications based on academic level and usage patterns. The prediction process integrates both textual and behavioral features, enabling context-aware recommendations.

The performance of the model is validated using evaluation metrics such as accuracy, precision, recall, and F1-score. Comparative analysis with baseline models confirms the effectiveness of the chosen approach. Finally, the trained model is saved using appropriate serialization techniques, allowing it to be deployed for real-time use. This ensures that the system can provide instant recommendations for new users without requiring retraining, making it suitable for integration into modern digital learning platforms.

**Techniques and Algorithms Used**

The existing technique considered in this study is the Gated Recurrent Unit (GRU), a type of recurrent neural network designed to model sequential data efficiently. GRU uses gating mechanisms to regulate the flow of information, enabling it to capture long-term dependencies while addressing issues such as vanishing gradients. It is commonly applied in time-series analysis, natural language processing, and recommendation systems due to its ability to learn from sequential user interactions. However, despite

ISSN 2347–3657

Volume 14, Issue 1, 2026

its effectiveness, GRU models often require significant computational resources, longer training time, and may lack interpretability.

In contrast, the proposed system utilizes the Random Forest Classifier, an ensemble learning algorithm that builds multiple decision trees and combines their outputs to produce accurate and stable predictions. In this approach, NLP techniques are first applied to preprocess textual data such as app descriptions and user interaction content. The processed text is converted into numerical feature vectors, which are then used as input for the Random Forest model. Each decision tree independently predicts the suitability of an application for a specific academic level, and the final recommendation is determined through majority voting.

This method offers several advantages, including reduced overfitting, efficient handling of high-dimensional data, and improved interpretability through feature importance analysis. As a result, the proposed system delivers reliable and personalized recommendations that align with students' academic needs, making it a practical and effective solution for educational app recommendation systems.

## Requirements Engineering

### General Overview

In evaluating the performance of the proposed system across multiple databases, the results demonstrate consistently low error rates. This is primarily attributed to the discriminative power of the selected features combined with the regression capabilities of the employed classifiers. When comparing the highest accuracies, which correspond to the lowest error rates, with those reported in prior studies, the system shows competitive performance, highlighting its robustness and efficiency in processing data.

### Hardware Requirements

The hardware requirements serve as a critical foundation for the system's implementation contract, providing a complete and consistent specification of the overall system. These requirements guide software engineers in designing the system architecture, focusing on *what* the system should accomplish rather than *how* it should be implemented. The minimum hardware specifications include a dual-core processor, 4 GB of DDR RAM, and a 250 GB hard disk, ensuring that the system can operate efficiently while supporting all intended functionalities.

### Software Requirements

The software requirements document defines and specifies the system's expected functionalities. It serves as a blueprint for developing the software requirements specification, which in turn aids in cost estimation, team planning, task allocation, and progress monitoring throughout the development lifecycle. The system is designed to operate on Windows 7, 8, or 10, using Spyder3 as the development platform and Python as the primary programming language. Both the development environment and front-end interface are implemented through Spyder3 to streamline coding and testing processes.

### Functional Requirements

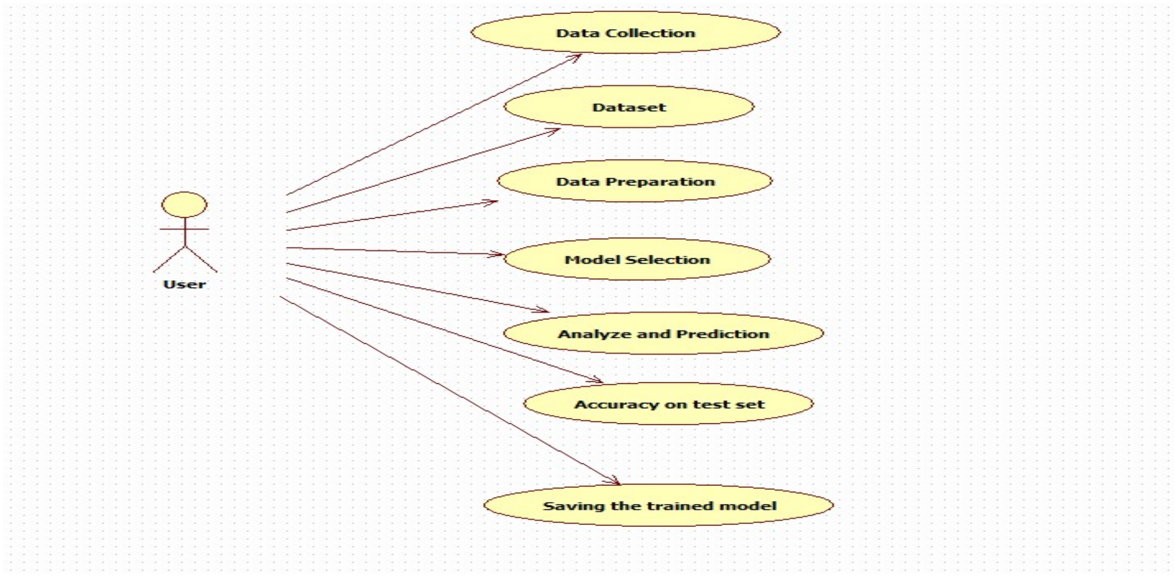
Functional requirements describe the specific behaviors and functions of the system or its components. The proposed system introduces a novel approach to achieving semantic security for data confidentiality in attribute-based deduplication systems by leveraging a hybrid cloud architecture. This design ensures that sensitive data remains secure while optimizing storage efficiency, marking a significant advancement over conventional methods.

### Non-Functional Requirements

The system also incorporates several critical non-functional requirements that enhance its usability, reliability, performance, supportability, and overall implementation. The automated nature of the system minimizes the need for user intervention, improving usability. Leveraging Python as the development platform contributes to higher reliability due to its robust and mature coding environment. Performance is optimized through the use of high-level programming languages and advanced back-end technologies, allowing rapid response times for end users. Supportability is ensured through cross-platform compatibility, enabling the system to operate across a broad range of hardware and software configurations. The system is deployed in a web environment via Jupyter Notebook, with Windows 10 Professional serving as the underlying platform, and a dedicated server managing intelligent operations. The user interface, accessible through Jupyter Notebook, provides seamless interaction with the server environment, ensuring both accessibility and operational efficiency.

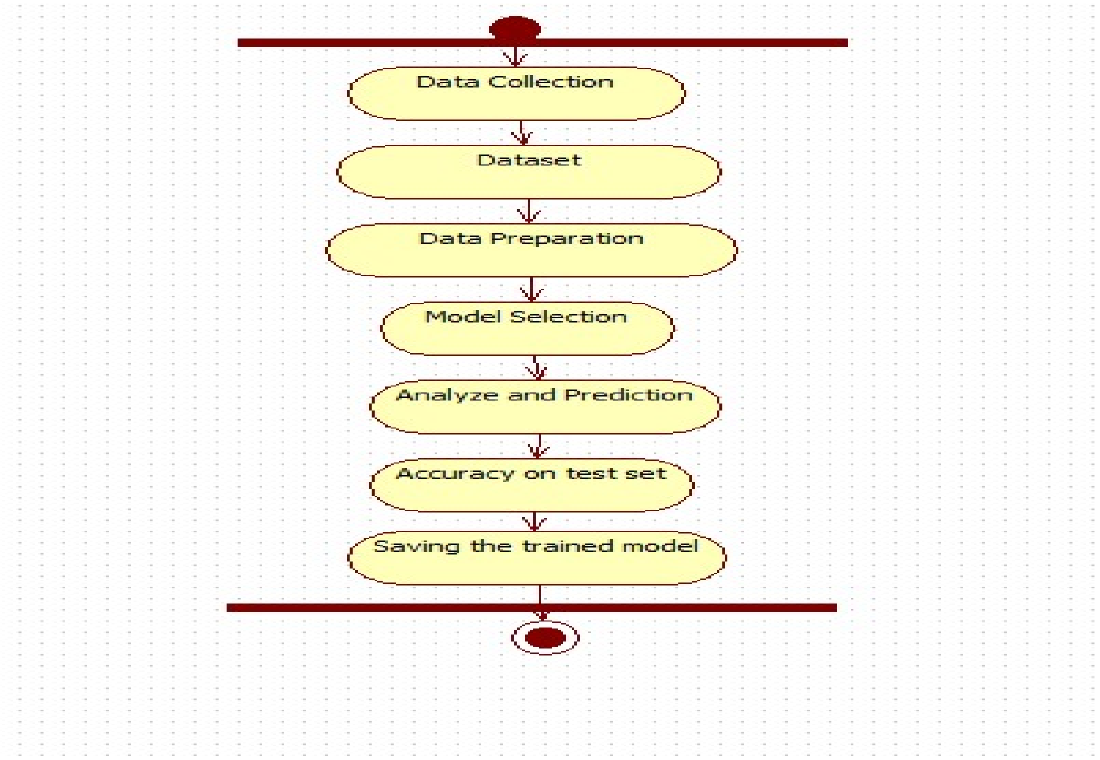
## UML DIAGRAMS

### USE CASE DIAGRAM



**EXPLANATION:**
The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

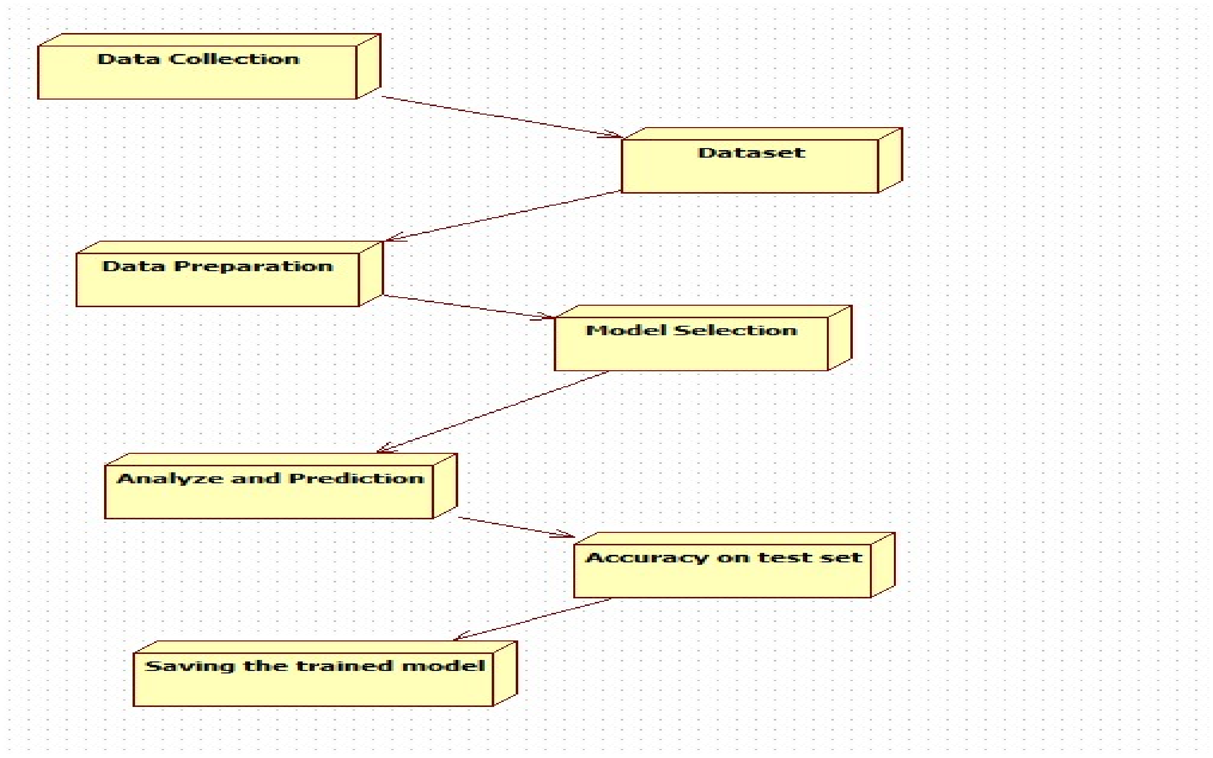
## ACTIVITY DIAGRAM



**EXPLANATION:**
Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

ISSN 2347–3657
Volume 14, Issue 1, 2026

DEPLOYMENT DIAGRAM



**EXPLANATION:**
Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it.

**Development Tools**
**PythonProgrammingLanguage**
Python is a high-level, interpreted, interactive, and object-oriented scripting language renowned for its readability and simplicity. Unlike many programming languages that rely heavily on punctuation, Python frequently uses English keywords, which reduces syntactical complexity and makes code more intuitive. The language was originally developed by Guido van Rossum in the late 1980s and early 1990s at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python draws inspiration from a variety of programming languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and Unix shell scripting, and has evolved into a versatile, widely adopted language. Its source code is distributed under the GNU General Public License (GPL), and it is maintained by a core development team, with Guido van Rossum continuing to provide strategic guidance on its development.

Python is valued for several key characteristics that enhance its usability and versatility. As an interpreted language, Python executes code at runtime, eliminating the need for compilation and facilitating rapid development and testing. Its interactive mode allows programmers to engage directly with the interpreter, supporting experimentation and iterative code development. Python also supports object-oriented programming (OOP), enabling code encapsulation and reuse, and is particularly suitable for beginners due to its simple syntax and clear structure. Beyond basic programming, Python is capable of supporting a wide spectrum of applications, from text processing and web development to games and data analysis.
The language offers a comprehensive set of features that make it both powerful and accessible. Python is easy to learn and maintain, with a concise set of keywords, a clear syntax, and a readable code structure. Its broad standard library is cross-platform compatible, working seamlessly across UNIX, Windows, and macOS systems. Python's portability and extendability allow developers to integrate low-level modules or interface with other languages such as C, C++, and Java, as well as technologies like COM, ActiveX, and CORBA. It supports GUI development across multiple platforms and provides robust database connectivity. Furthermore, Python's dynamic typing, automatic memory management, functional and structured programming paradigms,

and scalability make it suitable for both small-scale scripts and large, complex software systems.

In addition, Python includes a rich ecosystem of libraries that enhance its capabilities for scientific computing, data analysis, visualization, and machine learning. Prominent libraries include **NumPy**, which provides N-dimensional array objects; **pandas**, which offers data structures such as dataframes for efficient data analysis; **matplotlib**, which enables the creation of high-quality 2D plots; and **scikit-learn**, which provides a suite of machine learning algorithms for tasks such as classification, regression, and clustering. Together, these features and libraries make Python an ideal development tool for modern computational and data-driven applications.

Figure : NumPy, Pandas, Matplotlib, Scikit-learn

**Software Testing**

Software testing is a critical phase in the software development lifecycle, aimed at identifying defects, errors, or weaknesses in a system to ensure it meets functional requirements and user expectations. Testing involves systematically exercising the software to verify that it behaves as intended and does not fail under specified conditions. To achieve this, a structured methodology is employed, beginning with a comprehensive test plan that outlines the verification of general functionality and special features across multiple platform configurations. Strict quality control measures are applied to confirm that the application conforms to the specifications defined in the system requirements document and is free from critical defects.

Several types of testing are implemented to address different aspects of system quality. **Unit testing** focuses on individual components of the software, verifying that internal logic, decision branches, and data flows produce valid outputs. This structural testing is performed after completing each unit, ensuring that all code paths function as specified. **Functional testing** evaluates whether the system's features operate according to business and technical requirements, validating valid and invalid inputs, functional outputs, and interactions with external procedures. **System testing** examines the integrated software as a whole, confirming that all components work together to produce predictable and reliable results. **Performance testing** assesses the system's responsiveness, measuring processing times for tasks such as compilation, data retrieval, and user interactions. **Integration testing** evaluates the interaction between multiple software components to identify interface defects and ensure seamless interoperability. Finally, **user acceptance testing (UAT)** involves end users in validating that the system satisfies functional requirements and operates correctly in real-world scenarios. For example, in a data synchronization context, acknowledgments must be received by sender nodes, route operations occur only upon request, and cache updates reflect the current node status automatically. Collectively, these testing methodologies ensure a robust, reliable, and high-performing system.

A test plan is built by dividing the project into manageable units and designing specific testing strategies for each component. Unit tests help detect potential bugs early, allowing developers to correct errors before system integration. This structured approach improves overall software quality, reduces maintenance costs, and ensures that the final product meets both functional and non-functional expectations.

**Future Enhancements**

Future improvements to the system can significantly enhance its intelligence, adaptability, and user engagement. Advanced deep learning models, such as BERT, RoBERTa, or LSTM, could be incorporated to capture richer contextual information from educational app descriptions, enhancing the accuracy of recommendations. Real-time monitoring of student interaction data from mobile applications would allow dynamic updates to personalized suggestions, while additional behavioral metrics, such as time spent on apps, click patterns, and learning outcomes, could further refine recommendations. Multi-lingual support would broaden accessibility for students using apps in regional languages, and hybrid recommendation techniques combining collaborative filtering and content-based methods could address challenges related to cold-start and data sparsity. Explainable AI techniques could be integrated to provide transparency in recommendation decisions, enhancing trust for both students and educators. Furthermore, seamless integration with university learning management systems (LMS) would align app suggestions with academic curricula, and mobile or web-based interfaces could enable instant access to personalized recommendations. Gamification elements could increase engagement, and long-term studies could assess the system's impact on learning outcomes, providing insights for iterative improvements.

## Conclusion

This project presents a personalized educational app recommendation system designed to enhance learning outcomes for university students at undergraduate, postgraduate, and graduate levels. By leveraging app usage data and textual descriptions, the system employs Natural Language Processing (NLP) techniques to extract meaningful features that reflect students' preferences and academic requirements. The Random Forest Classifier was selected as the core predictive model due to its robustness, interpretability, and capability to handle high-dimensional data. Experimental results demonstrate high accuracy, precision, and recall, indicating reliable and relevant recommendations. The system effectively addresses data sparsity challenges while providing scalable, context-aware suggestions tailored to individual learning styles. By integrating both behavioral and content-based features, the system ensures that recommended apps align with academic goals and foster greater engagement. The trained model can produce real-time predictions, supporting deployment on web and mobile platforms. This research highlights the potential of data-driven personalization in digital education, promoting more efficient use of resources and enhanced student learning experiences. Future enhancements, including deep learning integration, multilingual support, hybrid recommendation techniques, and gamification, can further improve system performance and user satisfaction. Overall, this project illustrates the capability of combining machine learning and NLP to create intelligent, adaptive, and interactive educational tools that support personalized learning in digital environments.

## References

[1] M. Zhai, X. Wang, and X. Zhao, "The importance of online customer review characteristics on remanufactured product sales: Evidence from the mobile phone market on Amazon.com," *J. Retailing Consum. Serv.*, vol. 77, Mar. 2024, Art. no. 103677.

[2] C. C. Aggarwal, *An Introduction to Recommender Systems*, in *Recommender Systems: The Textbook*, Cham, Switzerland: Springer, 2015, pp. 1–28.

[3] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan. 2003.

[4] N. Tintarev and J. Masthoff, "Beyond explaining single item recommendations," in *Recommender Systems Handbook*, Cham, Switzerland: Springer, 2022, pp. 711–756.

[5] C. Musto, M. de Gemmis, P. Lops, F. Narducci, and G. Semeraro, "Semantics and content-based recommendations," in *Recommender Systems Handbook*, Cham, Switzerland: Springer, 2022, pp. 251–298.

[6] P. Winoto and T. Y. Tang, "The role of user mood in movie recommendations," *Expert Syst. Appl.*, vol. 37, no. 8, pp. 6086–6092, Aug. 2010.

[7] P. Li and A. Tuzhilin, "A dynamical system framework for exploring consumer trajectories in recommender systems," SSRN, May 2024, pp. 1–55.

[8] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. Artif. Intell.*, vol. 2009, no. 1, Oct. 2009, Art. no. 421425.

[9] M. Asad, S. Shaukat, E. Javanmardi, J. Nakazato, and M. Tsukada, "A comprehensive survey on privacy-preserving techniques in federated recommendation systems," *Appl. Sci.*, vol. 13, no. 10, p. 6201, May 2023.

[10] M. Singh, "Scalability and sparsity issues in recommender datasets: A survey," *Knowl. Inf. Syst.*, vol. 62, no. 1, pp. 1–43, Jan. 2020.

[11] M. Moser, *College Tennis University Recommendation System (CTURS)*, M.S. thesis, Dept. Psychol., Univ. Salzburg, Austria, 2022.

[12] I. Hossain, M. A. H. Palash, A. T. Sejuty, N. A. Tanjim, M. A. A. L. Nasim, S. Saif, A. B. Suraj, M. M. A. Haque, and N. Karim, "A survey of recommender system techniques in the e-commerce domain," 2022, arXiv:2208.07399.

[13] G. B. Martins, J. P. Papa, and H. Adeli, "Deep learning techniques for recommender systems based on collaborative filtering," *Expert Syst.*, vol. 37, no. 6, p. 12647, Dec. 2020.

[14] F. Camilli and M. Mézard, "The decimation scheme for symmetric matrix factorization," *J.*

*Phys. A, Math. Theor.*, vol. 57, no. 8, Feb. 2024, Art. no. 085002.

[15] H. Nabli, R. B. Djemaa, and I. A. B. Amor, "Improved clustering-based hybrid recommendation system for personalized cloud services," *Cluster Comput.*, vol. 27, no. 3, pp. 2845–2874, Jun. 2024.

[16] A. A. Ka'bi, "Proposed AI algorithm and deep learning techniques for higher education development," *Int. J. Intell. Netw.*, vol. 4, pp. 68–73, Jun. 2023.

[17] W. Wang, W. Chen, Y. Luo, Y. Long, Z. Lin, L. Zhang, B. Lin, D. Cai, and X. He, "Model compression and efficient inference for large language models: A survey," 2024, arXiv:2402.09748.

[18] R. Nahta, G. S. Chauhan, Y. K. Meena, and D. Gopalani, "Deep learning with generative models for recommender systems: A survey," *Comput. Sci. Rev.*, vol. 53, Aug. 2024, Art. no. 100646.

[19] J. H. Yoon and B. Jang, "Evolution of deep learning-based sequential recommender systems: Current trends and new perspectives," *IEEE Access*, vol. 11, pp. 54265–54279, 2023.

[20] A. K. Yengikand, M. Meghdadi, S. Ahmadian, S. M. J. Jalali, A. Khosravi, and S. Nahavandi, "Deep representation learning using multilayer perceptron and stacked autoencoder for recommendation systems," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2021, pp. 2485–2491.

[21] J. Choi, J.-K. Rhee, and H. Chae, "Cell subtype classification via representation learning based on a denoising autoencoder for single-cell RNA sequencing," *IEEE Access*, vol. 9, pp. 14540–14548, 2021.

[22] Y. Geng, X. Xiao, X. Sun, and Y. Zhu, "Representation learning: Recommendation with knowledge graph via triple-autoencoder," *Front. Genet.*, vol. 13, Jun. 2022, Art. no. 891265.

[23] S. K. Pandey and R. R. Janghel, "Recent deep learning techniques, challenges and applications for medical healthcare system: A review," *Neural Process. Lett.*, vol. 50, no. 2, pp. 1907–1935, 2019.

[24] S. Nosouhian, F. Nosouhian, and A. K. Khoshouei, "A review of recurrent neural network architectures for sequence learning: Comparison between LSTM and GRU," Preprints, Jul. 2021. [Online]. Available: https://www.preprints.org/manuscript/202107.0252/v1

[25] S. Zargar, "Introduction to sequence learning models: RNN, LSTM, GRU," Dept. Mechanical and Aerospace Eng., North Carolina State Univ., Raleigh, NC, USA, Tech. Rep., 2021.

[26] I. Rabiu, N. Salim, A. Da'u, and A. Osman, "Recommender system based on temporal models: A systematic review," *Appl. Sci.*, vol. 10, no. 7, p. 2204, Mar. 2020.

[27] K. Sun, T. Qian, T. Chen, Y. Liang, Q. V. H. Nguyen, and H. Yin, "Where to go next: Modeling long- and short-term user preferences for point-of-interest recommendation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 1, 2020, pp. 214–221.

[28] A. Abirami and S. D. S. Azariya, "Fractional-order backpropagation through time for training recurrent neural networks," *Nanotechnol. Perceptions*, vol. 20, no. S2, pp. 756–767, 2024.

[29] L. V. Nguyen, Q.-T. Vo, and T.-H. Nguyen, "Adaptive KNN-based extended collaborative filtering recommendation services," *Big Data Cogn. Comput.*, vol. 7, no. 2, p. 106, May 2023.

[30] F. Roy and M. Hasan, "An item–item collaborative filtering recommender system based on item reviews: Deep learning approach," *Vietnam J. Comput. Sci.*, vol. 10, no. 4, pp. 517–536, Aug. 2023.

[31] J. Lee, S. Kim, G. Lebanon, and Y. Singer, "Local low-rank matrix approximation," in *Proc. 30th Int. Conf. Mach. Learn.*, Jun. 2013, pp. 82–90.

[32] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen, "Scalable collaborative filtering using cluster-based smoothing," in *Proc. 28th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2005, pp. 114–121.

[33] P. Dwivedi and B. Islam, "An item-based collaborative filtering approach for movie recommendation system," in *Proc. 10th Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, Mar. 2023, pp. 153–158.

[34] E. Jawabreh, "Time-aware QoS web service selection using collaborative filtering: A literature review," in *Proc. 10th Eur. Conf. Service-Oriented Cloud Comput. (ESOCC)*, Larnaca, Cyprus, 2023, p. 55.

[35] E. Genov, C. D. Cauwer, G. V. Kriekinge, T. Coosemans, and M. Messagie, "Forecasting flexibility of electric vehicle charging: Tree and cluster-based methods," *Appl. Energy*, vol. 353, Jan. 2024, Art. no. 121969.

[36] A. Torkashvand, S. M. Jameii, and A. Reza, "Deep learning-based collaborative filtering recommender systems: A comprehensive review," *Neural Comput. Appl.*, vol. 35, no. 35, pp. 24783–24827, Dec. 2023.

[37] J. Wang, H. Mei, K. Li, X. Zhang, and X. Chen, "Collaborative filtering model of graph neural network based on random walk," *Appl. Sci.*, vol. 13, no. 3, p. 1786, Jan. 2023.

[38] P. Yadav, S. Tyagi, and H. Kaur, "Evolutionary extreme learning machine based collaborative filtering," *Int. J. Adv. Technol. Eng. Explor.*, vol. 10, no. 104, p. 858, 2023.

[39] M. Ravakhah, M. Jalali, Y. Forghani, and R. Sheibani, "Balanced hierarchical max margin matrix factorization for recommendation systems," *Expert Syst.*, vol. 39, no. 4, May 2022, Art. no. e12911.

[40] F. O. Isinkaye, “Matrix factorization in recommender systems: Algorithms, applications, and challenges,” *IETE J. Res.*, vol. 69, no. 9, pp. 6087–6100, Sep. 2023.

[41] B. B. Sinha and R. Dhanalakshmi, “Mining user interest using Bayesian PMF and Markov chain Monte Carlo for personalized recommendation systems,” in *Proc. Int. Conf. Innov. Data Anal.*, Cham, Switzerland: Springer, Jan. 2022, pp. 115–129.

[42] A. Pujahari and D. S. Sisodia, “Pair-wise preference relation based probabilistic matrix factorization for collaborative filtering in recommender systems,” *Knowl.-Based Syst.*, vol. 196, May 2020, Art. no. 105798.

[43] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[44] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder–decoder for statistical machine translation,” 2014, arXiv:1406.1078.