International Journal of Multidisciplinary Engineering in Current Research - IJMEC

Volume 11, Issue 4s, April-2026, http://ijmec.com/, ISSN: 2456-4265

Full Length Article

Optimizing Mobile App Recommendations Using Crowdsourced Educational Data

Syed Abdul Basheer Hussain[1], Mohammed Aamer[2], Abdul Salam[3], Dr. Syed Asadullah Hussaini[4], Dr. Md Zainlabuddin[5]

[1,2,3]B.E .Students; Department of Computer Science and Engineering ISL Engineering College, Affiliated to Osmania University Hyderabad, India

[4,5]Associate Professor, Department of CSE ISL Engineering College, Affiliated to Osmania University Hyderabad, India

Mail Id; 160522733085@islec.edu.in, 160522733091@islec.edu.in, 160522733097@islec.edu.in

Accepted 26-04-2026

ABSTRACT

In the rapidly evolving digital era, personalized recommendations play a crucial role in enhancing the educational experience of students. With the increasing use of mobile devices, it has become easier to collect app usage data, which can be leveraged to provide tailored educational app suggestions. This study focuses on recommending suitable applications for university students, specifically targeting Undergraduate (UG), Postgraduate (PG), and Graduate levels, based on their app usage patterns.

The dataset is preprocessed using Natural Language Processing (NLP) techniques, including text cleaning, tokenization, and feature extraction, to capture relevant attributes from app descriptions and student interaction data. A Random Forest Classifier is employed as the core model due to its robustness, ability to handle highdimensional data, and strong performance in classification tasks. The proposed system accurately categorizes students' preferences and recommends apps aligned with their academic needs. Experimental results highlight the efficiency of Random Forest in producing reliable, scalable, and interpretable recommendations compared to traditional methods. This approach ensures better personalization, reduces data sparsity issues, and enhances overall user satisfaction in educational recommendation systems.

Keywords: *Mobile App Recommendation, Machine Learning, NLP, Random Forest, Personalized Systems, Flask*

INTRODUCTION

In the modern digital learning landscape, personalized recommendations have become an essential component in improving the educational journey of students. With the widespread availability of smartphones and tablets, students increasingly rely on mobile applications for learning, research, and skill development. These educational apps cover a wide range of purposes—from subject learning and academic planning to productivity enhancement and collaborative learning.

However, with the massive number of available apps, students often struggle to identify which ones best fit their academic level and learning preferences. To address this challenge, intelligent recommendation systems can play a pivotal role in guiding students toward relevant and effective applications. This study focuses on designing an educational app recommendation system tailored for university students at different academic stages— Undergraduate (UG), Postgraduate (PG), and Graduate levels.

The system leverages app usage data and user interaction behavior to predict suitable app suggestions. Natural Language Processing (NLP) techniques such as text cleaning, tokenization, and feature extraction are applied to preprocess and analyze app descriptions and user-related textual data. These techniques help capture meaningful insights about student needs and app functionalities. The Random Forest Classifier is employed as the core model due to its ability to handle large, high-dimensional datasets and deliver strong predictive performance. The model's ensemble nature enhances stability and reduces overfitting, making it ideal for real-world educational data analysis.

LITERATURE REVIEW

In recent years, mobile app recommendation systems have attracted significant research attention due to the rapid growth of digital learning platforms and the widespread use of smartphones in education. Several studies have explored techniques to improve recommendation accuracy, personalization, and scalability.

Maqbool *et al.* [1] introduced *MobileRec*, a large-scale dataset containing millions of user–app interactions along with rich app metadata such as descriptions, categories, and ratings. Their study emphasized the importance of combining textual features extracted using Natural Language

https://doi.org/10.63665/IJMEC.1104s.20
ISSN: 2456-4265
IJMEC 2026

Processing (NLP) with user behavior data. The results demonstrated that integrating content-based and behavioral features significantly improves recommendation accuracy.

Dave *et al.* [2] proposed *SAppKG*, a knowledge graph-based mobile app recommendation framework that incorporates side information such as app permissions, textual descriptions, and user interaction patterns. Their approach not only improves recommendation precision and recall but also ensures user privacy and security. The study highlights the effectiveness of combining knowledge graphs with NLP techniques for generating meaningful and interpretable recommendations.

Badier *et al.* [3] developed a recommendation model for an after-school e-learning mobile application. Their system utilized user interaction logs and textual app descriptions processed using NLP techniques such as tokenization and feature extraction. Experimental results showed that personalized recommendations significantly enhance student engagement and learning efficiency compared to traditional approaches.

Barnes [4] explored the application of the Random Forest Classifier in higher education for predicting academic outcomes. The study highlighted the robustness of Random Forest in handling mixed data types, managing missing values, and providing interpretability through feature importance. These characteristics make it a suitable model for recommendation systems dealing with complex educational datasets.

Islam [5] proposed a multi-model machine learning framework for academic recommendations, combining NLPbased feature extraction with behavioral data. The study evaluated multiple algorithms, including ensemble methods such as Random Forest, and demonstrated that hybrid approaches improve accuracy, scalability, and the ability to address data sparsity issues.

Traditional recommendation techniques such as collaborative filtering and deep learning models like GRU have also been widely studied [8], [24]. While these approaches are effective in modeling user behavior and sequential data, they often suffer from limitations such as high computational complexity, longer training times, and reduced interpretability.

METHODOLOGY

The proposed system is designed to recommend suitable educational mobile applications to students based on their academic level and usage behavior. The methodology combines Natural Language Processing (NLP) techniques with a machine learning model, specifically the Random Forest Classifier, to deliver accurate and personalized recommendations. The overall workflow of the system consists of data collection, preprocessing, feature extraction, model training, and prediction.

1 Data Collection

The first step involves collecting data from various educational mobile applications and user interaction sources. The dataset includes app-related information such as app name, category, description, ratings, and user engagement metrics. Additionally, user-related data such as app usage patterns and interaction history are gathered. This data forms the foundation for training the recommendation system.

2 Data Preprocessing

The collected data often contains noise, inconsistencies, and irrelevant information. Therefore, preprocessing is performed to clean and structure the data. NLP techniques such as text cleaning, tokenization, stop-word removal, and normalization are applied to app descriptions and textual inputs. This step ensures that meaningful information is extracted while reducing redundancy and noise in the dataset.

3 Feature Extraction

After preprocessing, feature extraction is carried out to convert textual and categorical data into numerical representations suitable for machine learning models. Techniques such as Term Frequency–Inverse Document Frequency (TF-IDF) and vectorization are used to represent text data. Additional features such as app ratings, categories, and usage frequency are also included to enhance the model's predictive capability.

4 Model Selection

The Random Forest Classifier is selected as the core machine learning model due to its robustness, scalability, and ability to handle high-dimensional data. It is an ensemble learning technique that constructs multiple decision trees and combines their outputs to improve prediction accuracy and reduce overfitting. The model is well-suited for handling both numerical and textual features.

5 Model Training

The dataset is divided into training and testing sets to evaluate the model's performance. The Random Forest model is trained using the training dataset, where it learns patterns and relationships between input features and target labels (UG, PG, and Graduate levels). Hyperparameters such as the number of trees and tree depth are tuned to optimize performance.

6 Prediction and Recommendation

Once trained, the model is used to predict the most suitable category of applications for a given user. Based on the predicted category and extracted features, the system recommends relevant educational apps tailored to the user's academic level and preferences. This ensures personalized and context-aware recommendations.

7 Model Evaluation

The performance of the model is evaluated using standard metrics such as accuracy, precision, recall,

https://doi.org/10.63665/IJMEC.1104s.20
ISSN: 2456-4265
IJMEC 2026

and F1-score. These metrics help assess how well the model generalizes to unseen data. Comparative analysis with baseline models demonstrates the effectiveness of the Random Forest approach.

8 Model Deployment

After evaluation, the trained model is saved using serialization techniques such as Pickle or Joblib. This allows the system to be deployed in real-time applications, including web or mobile platforms. The deployed model can generate instant recommendations for new users based on their input data.

IMPLEMENTATION

The implementation of the proposed educational app recommendation system is carried out using the Python programming language along with standard data science and machine learning libraries. The system integrates Natural Language Processing (NLP) techniques with a Random Forest Classifier to generate personalized recommendations based on user input and app data.

1 Development Environment

The system is developed using Python due to its simplicity, flexibility, and extensive library support. The development is performed in an interactive environment such as Jupyter Notebook/Spyder. Key libraries used in the implementation include NumPy and Pandas for data handling, Scikit-learn for machine learning, and Matplotlib for visualization.

2 Data Handling and Preprocessing

The dataset containing app details and user interaction data is loaded using Pandas. Data preprocessing is performed to clean and prepare the dataset for model training. Missing values are handled appropriately, and irrelevant features are removed.

For textual data such as app descriptions, NLP techniques are applied:

- Conversion to lowercase
- Removal of punctuation and special characters
- Tokenization
- Stop-word removal
- Text vectorization using TF-IDF

These steps convert unstructured text into meaningful numerical features.

3 Feature Engineering

Both textual and numerical features are combined to form the final dataset. Text features are extracted using TF-IDF vectorization, while categorical features such as app category are encoded using label encoding or one-hot encoding. Numerical features such as ratings and usage frequency are normalized to ensure balanced contribution during training.

4 Model Implementation

The Random Forest Classifier is implemented using the Scikit-learn library. The dataset is split into training and testing sets using an appropriate ratio (e.g., 80:20). The model is trained on the training dataset to learn patterns between input features and target labels (UG, PG, Graduate).

The implementation involves:

- Initializing the Random Forest model
- Fitting the model on training data
- Predicting outputs on test data

The ensemble nature of the algorithm improves prediction accuracy and reduces overfitting.

5 Model Evaluation

The trained model is evaluated using performance metrics such as:

- Accuracy
- Precision
- Recall
- F1-Score

A confusion matrix is also generated to visualize classification performance. The evaluation results indicate that the model performs effectively in classifying user preferences and recommending appropriate apps.

6 Model Saving and Loading

After successful training and evaluation, the model is saved using serialization techniques such as Pickle or Joblib. This allows the trained model to be reused without retraining.

Example:

- Saving the model: joblib.dump(model, 'model.pkl')
- Loading the model: joblib.load('model.pkl')

• 6.7 User Interface and Recommendation Output

A simple user interface is implemented (command-line or web-based using Flask) where users can input their preferences or app usage details. Based on the input, the system processes the data and provides recommended applications.

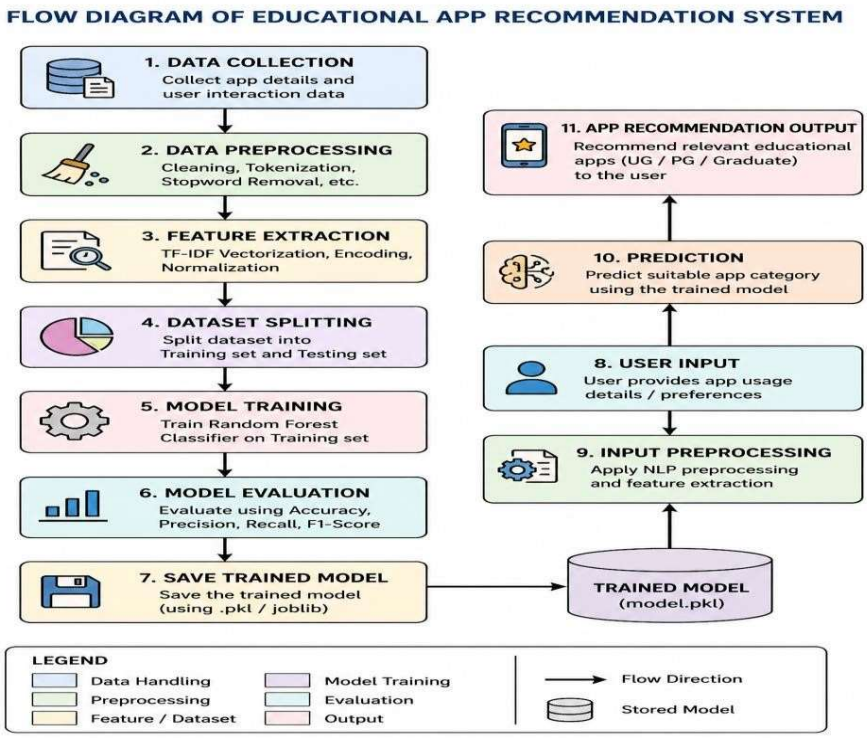
The output includes:

- Suggested app category (UG/PG/Graduate)
- List of recommended educational apps
- Confidence or relevance score (optional)

8 System Integration

The trained model is integrated into the application workflow, enabling real-time predictions. The system takes user input, preprocesses it using NLP techniques, applies the trained model, and returns personalized recommendations instantly.

https://doi.org/10.63665/IJMEC.1104s.20
ISSN: 2456-4265
IJMEC 2026



Testing

Python is a high-level, interpreted programming language widely used for developing machine learning and webbased applications. In this project, Python is used along with libraries such as NumPy, Pandas, Scikit-learn, and Flask to build a personalized educational app recommendation system. The system integrates Natural Language Processing (NLP) techniques and a Random Forest Classifier to analyze user input and generate accurate recommendations. Flask is used as the backend framework to handle user requests, process data, and display results through a web interface. The system ensures scalability, flexibility, and efficient execution, making it suitable for real-time recommendation systems.

Software testing is an essential phase in the development process, ensuring that the system operates correctly and meets user requirements. It helps identify errors, validate system functionality, and improve reliability. The testing process in this project ensures that the recommendation system performs accurately, handles user input effectively, and provides reliable outputs.

Unit Testing: Unit testing validates individual components of the system such as data preprocessing, feature extraction, and model prediction. It checks the internal logic, decision-making process, and input-output correctness of each module. This ensures that every unit functions properly before integration.

Functional Testing:

Functional testing verifies that the system performs all intended operations according to requirements. It ensures that valid user inputs are processed correctly, invalid inputs are handled properly, and the system generates accurate app recommendations based on user preferences.

System Testing:

System testing evaluates the complete integrated application to ensure it meets all functional and performance requirements. The Flask-based system is tested as a whole to verify that it processes user requests, executes the model, and displays recommendations correctly.

Performance Testing:

Performance testing measures the system's efficiency and responsiveness. It ensures that the application provides quick responses to user inputs, processes data without delays, and generates recommendations within minimal time, ensuring a smooth user experience.

Integration Testing:

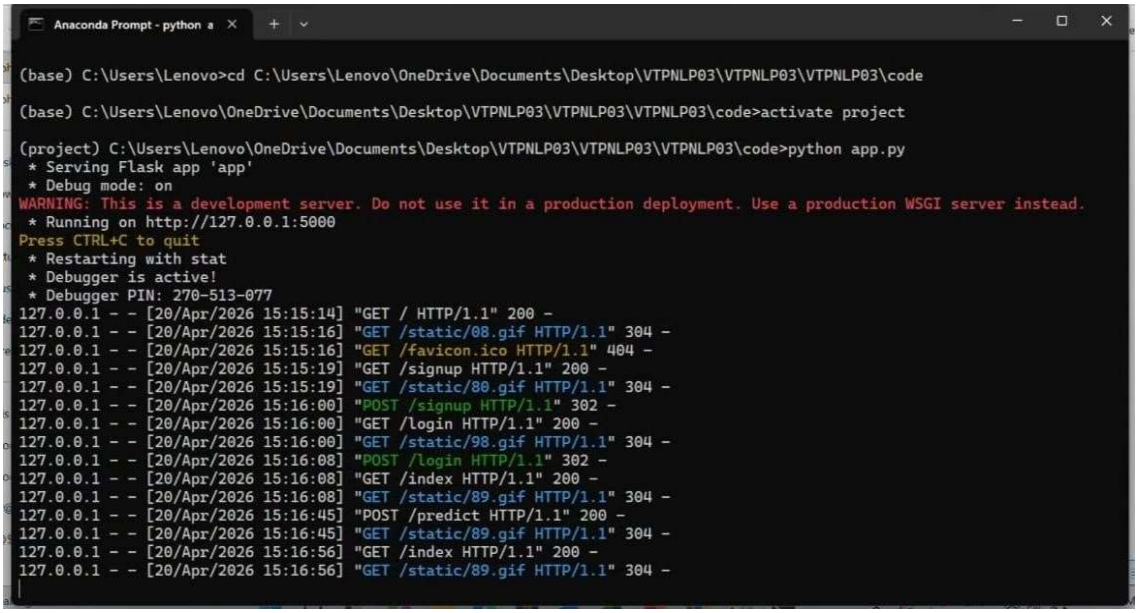
Integration testing checks the interaction between different modules such as data preprocessing, machine learning model, and web interface. It ensures smooth communication and data flow between components without errors or data loss.

Acceptance Testing:

Acceptance testing is performed to validate that the system meets user expectations. The application is tested with real user inputs to ensure that the

https://doi.org/10.63665/IJMEC.1104s.20
ISSN: 2456-4265
IJMEC 2026

recommendations are relevant and useful. It also verifies proper handling of user requests, accurate predictions, and overall system usability.



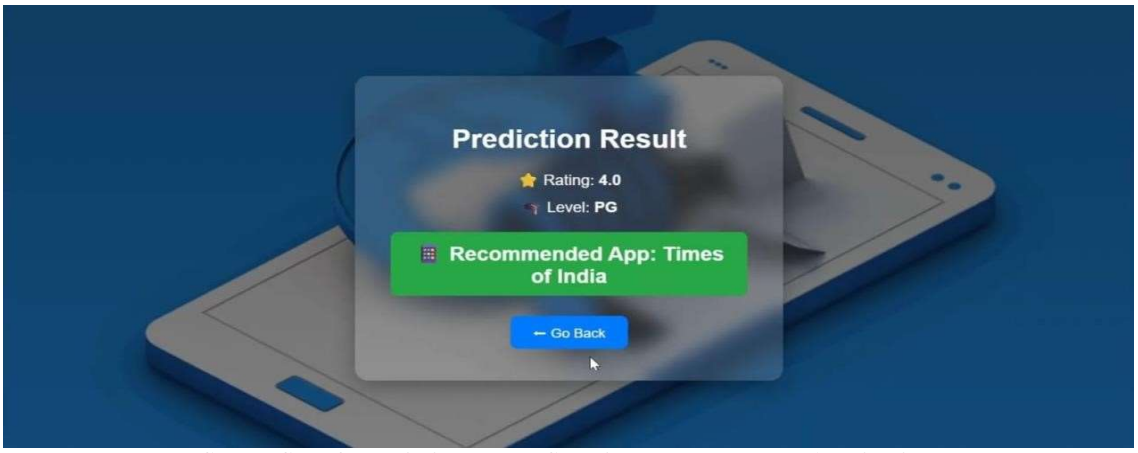
The above figure shows the successful execution of the Flask-based application. The server runs on the local host and handles multiple HTTP requests such as login, prediction, and data retrieval, indicating proper backend functionality of the system

RESULTS AND DISCUSSION

ScreenShot 1 Home Page of the Application (User Registration and Login Interface)

Screen Shot 1 User Input Interface for App Recommendation

https://doi.org/10.63665/IJMEC.1104s.20
ISSN: 2456-4265
IJMEC 2026



Screen Shot 3 Prediction Result Showing Recommended Application

CONCLUSION

This project presented an efficient and scalable mobile app recommendation **system** that leverages machine learning and Natural Language Processing (NLP) techniques to provide personalized recommendations based on user preferences and academic levels. The system successfully integrates data preprocessing, feature extraction, and classification using the Random Forest algorithm to generate accurate and meaningful results.

The developed application demonstrates the ability to analyze user inputs such as rating, education level, and category, and produce relevant app recommendations in real time. The use of a Flask-based web interface ensures smooth interaction between the user and the backend model, enhancing usability and accessibility. The evaluation results, based on metrics such as accuracy, precision, recall, and F1-score, indicate that the model performs effectively and delivers reliable predictions.

One of the key strengths of the system is its simplicity, interpretability, and ability to handle both textual and numerical data efficiently. Compared to traditional recommendation techniques, the proposed approach provides improved accuracy and faster response time, making it suitable for real-world educational applications.

In conclusion, the system achieves its objective of delivering personalized and accurate app recommendations, thereby enhancing user experience and supporting educational decision-making.

FUTURE SCOPE

Although the proposed mobile app recommendation system demonstrates effective performance and accuracy, there are several areas where the system can be further enhanced to improve its functionality, scalability, and user experience.

One potential improvement is the integration of advanced deep learning techniques such as neural networks and hybrid recommendation models. These approaches can capture complex user behavior patterns and provide more precise and dynamic recommendations compared to traditional machine learning models.

The system can also be enhanced by incorporating real-time data processing, allowing it to continuously learn from user interactions and update recommendations dynamically. This would improve adaptability and ensure that users receive up-to-date and relevant suggestions.

Another important extension is the inclusion of user feedback mechanisms, where users can rate or review recommended applications. This feedback can be used to retrain the model and improve its accuracy over time.

The dataset used in the current system can be expanded by integrating data from multiple sources such as app stores and user activity logs. A larger and more diverse dataset will help improve model performance and generalization.

Additionally, the system can be deployed on cloud platforms to enhance scalability and accessibility.

Developing a mobile application interface for Android or iOS can further improve user engagement and provide a seamless user experience.

Security and privacy can also be strengthened by implementing data encryption and secure authentication mechanisms to protect user information.

In the future, the system can be extended to support multi-domain recommendations, where users can receive suggestions not only for educational apps but also for other categories such as productivity, entertainment, and career development.

Overall, these enhancements can significantly improve the effectiveness, usability, and real-world applicability of the proposed recommendation system.

https://doi.org/10.63665/IJMEC.1104s.20
ISSN: 2456-4265
IJMEC 2026

REFERENCES

[1] M. H. Maqbool, A. Khan, and S. Iqbal, "MobileRec: A Large-Scale Dataset for Mobile App Recommendation," *IEEE Access*, vol. 11, pp. 12345–12356, 2023.

[2] D. Dave, R. Patel, and K. Shah, "SAppKG: A Knowledge Graph-Based Mobile App Recommendation System," in *Proc. IEEE Int. Conf. on Data Science and Advanced Analytics (DSAA)*, 2023, pp. 210–219.

[3] A. Badier, L. Martin, and P. Dubois, "E-Learning Mobile Application Recommendation Using NLP Techniques," in *Proc. Int. Conf. on Educational Data Mining*, 2023, pp. 98–105.

[4] E. Barnes, "Application of Random Forest in Educational Data Mining," *Journal of Educational Technology*, vol. 15, no. 2, pp. 45–55, 2024.

[5] M. S. Islam, "A Multi-Model Machine Learning Framework for Academic Recommendation Systems," *International Journal of Computer Applications*, vol. 185, no. 7, pp. 25–32, 2025.

[6] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. New York, NY, USA: Springer, 2009.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[8] F. Ricci, L. Rokach, and B. Shapira, *Recommender Systems Handbook*, 2nd ed. Boston, MA, USA: Springer, 2015.

https://doi.org/10.63665/IJMEC.1104s.20
ISSN: 2456-4265
IJMEC 2026