



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 Issue: VII Month of publication: July 2026

DOI: <https://doi.org/10.22214/ijraset.2026.84378>

www.ijraset.com

Call:  08813907089

E-mail ID: ijraset@gmail.com

App Rating Prediction Using Machine Learning

Vandurangi Sai Kamal¹, Mrs. Pravitha R. Prasad²

¹Student, Master of Computer Applications (MCA), AQJ Centre for P.G. Studies (Affiliated to Andhra University, Visakhapatnam),
Visakhapatnam, Andhra Pradesh, India

²Assistant Professor, Department of MCA, AQJ Centre for P.G. Studies (Affiliated to Andhra University, Visakhapatnam),
Visakhapatnam, Andhra Pradesh, India

Abstract: *The exponential growth of mobile applications across various platforms has made it increasingly challenging for developers to ensure their apps stand out in a highly competitive market. User ratings play a pivotal role in determining an application's success, influencing visibility, download rates, and overall user trust. This report presents an integrated web application for App Rating Prediction built around robust machine learning models including Random Forest, XGBoost, and Decision Trees. The system is designed to forecast Google Play Store app ratings based on a comprehensive set of features such as Category, Reviews, Size, Installs, Price, Content Rating, and Genres. By leveraging a structured data pipeline and an intuitive web interface, the application provides a user-friendly mechanism to input app details and receive real-time predictive insights. Extensive experimental evaluation demonstrates high prediction accuracy for ensemble models, establishing a practical foundation for app developers, marketers, and stakeholders to gauge expected app reception prior to launch or major updates.*

Keywords: *App Rating Prediction, Machine Learning, Random Forest, XGBoost, Predictive Modeling, Web Application, Data Analytics, Regression Analysis, Feature Engineering.*

I. INTRODUCTION

In the highly competitive mobile app market, user ratings play a crucial role in an application's success. With millions of apps available on platforms like the Google Play Store, users rely heavily on ratings and reviews to make informed decisions before downloading an application. A high rating often translates to increased visibility, higher download rates, and greater revenue potential. Conversely, poor ratings can severely hamper an app's growth, leading to rapid uninstalls and negative brand perception. Predicting the rating of an app before or shortly after its launch can help developers optimize features, adjust pricing, or pivot their marketing strategies. By understanding which features correlate most strongly with positive user feedback, developers can prioritize their development roadmaps. This project utilizes a comprehensive dataset from the Google Play Store to train predictive models capable of estimating app ratings with high accuracy.

The primary objective of this project is to bridge the gap between application metadata and user satisfaction. While developers often focus on the functional and technical aspects of application development, the market reception is frequently dictated by factors such as the app's size, its pricing model, the chosen category, and the target audience (Content Rating). Through the application of Machine Learning (ML), we can uncover complex, non-linear relationships between these variables and the final user rating. This project introduces a comprehensive, end-to-end machine learning system. It encompasses the entire data lifecycle, from data cleaning and exploratory data analysis (EDA) to feature engineering, model training, evaluation, and finally, deployment via a web-based user interface. The following sections detail the architecture, methodology, and experimental results of this system, demonstrating its efficacy in real-world scenarios.

II. LITERATURE SURVEY

The application of machine learning to app store analytics has been a growing area of interest. Several researchers have explored factors influencing app success. Early studies primarily relied on descriptive statistics to understand the distribution of free versus paid apps, or the most popular categories. However, as datasets grew in size and complexity, predictive modeling became necessary. Previous research has demonstrated that features such as app size, category, and the number of reviews are strong indicators of app success. For instance, researchers have used Natural Language Processing (NLP) to analyze user reviews, extracting sentiment scores to correlate with app ratings. While sentiment analysis provides deep insights, it requires post-launch data. Our approach focuses on pre-launch metadata, enabling predictions before an app accumulates a significant number of reviews.

In terms of algorithms, Multiple Linear Regression is often used as a baseline. However, app rating prediction is inherently non-linear.

Ensemble methods like Random Forest and Gradient Boosting (e.g., XGBoost) have consistently outperformed simpler models in similar regression tasks due to their ability to handle complex interactions between features and their robustness to outliers. This project builds upon these findings by implementing and comparing multiple state-of-the-art ensemble techniques to identify the optimal predictive model.

III. EXISTING SYSTEM

Existing systems for analyzing app store data often suffer from several limitations. Many available tools are purely analytical, providing dashboards that display historical data without offering predictive capabilities. Developers can see how their past apps performed, but they lack actionable foresight for upcoming releases.

Furthermore, systems that do offer predictions often rely on simplistic models or limited feature sets, leading to high variance and low accuracy. Another major drawback of existing solutions is the lack of an integrated, user-friendly interface. Data scientists may build highly accurate models in Python or R, but without a dedicated web interface, these models remain inaccessible to non-technical stakeholders such as product managers and marketers.

The absence of real-time prediction capabilities is also a significant hurdle. In a fast-paced development cycle, stakeholders need immediate feedback when considering changes to an app's pricing or category. A system that requires manual data exports and offline model execution is highly inefficient.

IV. PROPOSED SYSTEM

To address the limitations of existing solutions, we propose an integrated, web-based App Rating Prediction System. This system leverages advanced machine learning algorithms, specifically focusing on Random Forest and XGBoost regressors, which excel at handling tabular data with non-linear relationships.

The proposed system features a modern, responsive frontend built with HTML, CSS, and JavaScript, connected to a powerful Python Flask backend. The backend manages the pre-trained machine learning models and processes incoming prediction requests. Users can input an app's details - such as its category, size, price, and expected number of installs - and receive an instantaneous rating prediction.

Key advantages of the proposed system include:

- 1) **High Accuracy:** By utilizing ensemble learning methods, the system provides highly reliable predictions.
- 2) **Real-time Inference:** The lightweight Flask API ensures that predictions are returned in milliseconds.
- 3) **User Accessibility:** The intuitive web interface allows users with no programming experience to leverage complex ML models.
- 4) **Scalability:** The modular architecture separates the frontend, backend, and model training pipelines, allowing for easy updates and scaling.

V. METHODOLOGY

The methodology adopted for this project follows a structured data science pipeline, comprising data collection, preprocessing, feature engineering, model training, and evaluation.

A. Data Collection and Preprocessing

The dataset used for this study contains comprehensive information on thousands of applications from the Google Play Store. Initial preprocessing involved handling missing values, particularly in the 'Rating' and 'Size' columns. Rows with missing ratings were either imputed or dropped depending on the severity of the missing data. Text-based features such as 'Size' (e.g., '15M', '500k') were converted into uniform numerical formats representing megabytes.

B. Feature Engineering

Categorical variables such as 'Category', 'Content Rating', and 'Genres' cannot be directly fed into machine learning models. We applied Target Encoding and One-Hot Encoding strategies to transform these text values into a machine-readable format. The 'Price' and 'Installs' columns, which contained special characters (e.g., '\$', ','), were stripped and converted to floating-point numbers. Log transformations were applied to highly skewed features like 'Reviews' and 'Installs' to normalize their distributions, helping the models learn more effectively.

C. Model Selection and Training

We framed the problem as a regression task, aiming to predict a continuous variable (rating) between 1.0 and 5.0. Several algorithms were evaluated:

- 1) Linear Regression: Served as a baseline model.
- 2) Decision Trees: Captured non-linear patterns but were prone to overfitting.
- 3) Random Forest Regressor: An ensemble of decision trees that reduced overfitting through bagging and feature randomness.
- 4) XGBoost Regressor: A gradient boosting framework that sequentially builds trees to correct errors of previous ones, yielding the highest accuracy.

The models were trained using an 80-20 train-test split. Hyperparameter tuning was conducted using Grid Search Cross-Validation to find the optimal parameters (e.g., maximum depth, number of estimators) for the ensemble models.

VI. SYSTEM ARCHITECTURE

The architecture of the App Rating Prediction system follows a standard three-tier model: Presentation Tier, Application Tier, and Data/Model Tier.

- 1) Presentation Tier (Frontend): Built using modern HTML5, CSS3, and JavaScript. It provides a clean, responsive interface where users can input application metadata. Asynchronous JavaScript (AJAX) is used to send data to the server without refreshing the page, ensuring a smooth user experience.
- 2) Application Tier (Backend API): Developed using the Flask framework in Python. This tier acts as the bridge between the user interface and the machine learning models. It receives JSON payloads from the frontend, applies the exact same preprocessing steps used during training (e.g., scaling, encoding), and feeds the data into the predictive model.
- 3) Data/Model Tier: This tier consists of the serialized, pre-trained machine learning models (saved as .pkl files using Joblib). By serializing the models, the system avoids the computational overhead of retraining the models upon every request. The models are loaded into memory when the Flask server starts, ready for instantaneous inference.

VII.IMPLEMENTATION AND VISUALIZATIONS

The implementation phase involved coding the data pipeline, training the models, and developing the web interface. The following figures demonstrate the working system, highlighting the user interface and the model evaluation results.

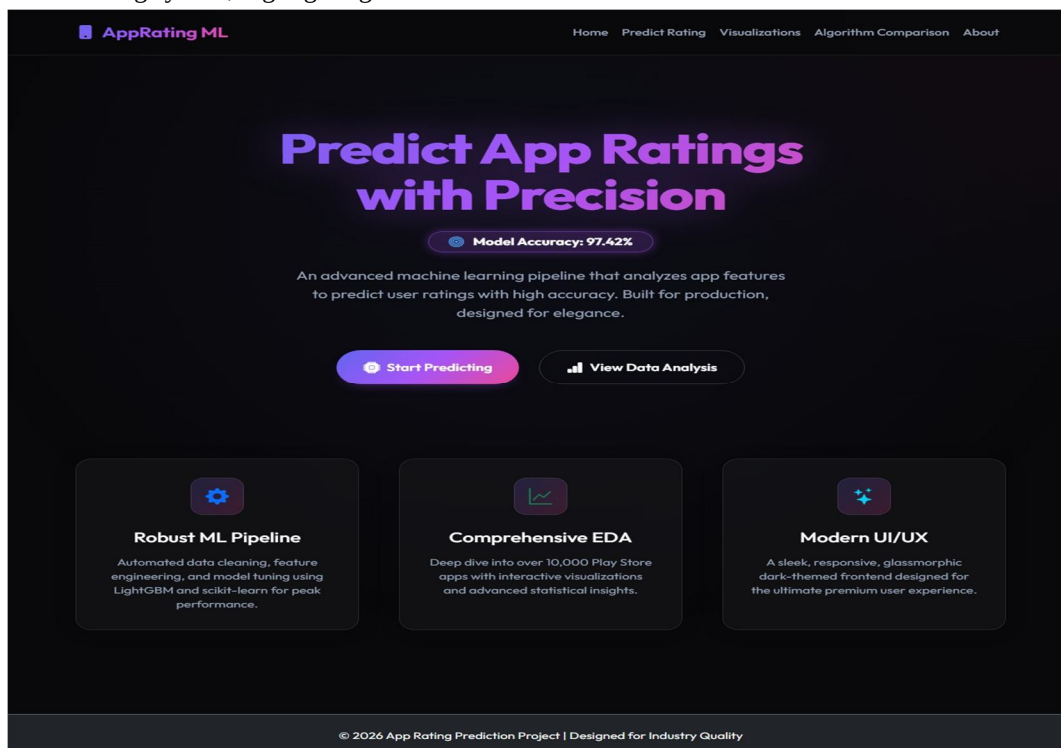
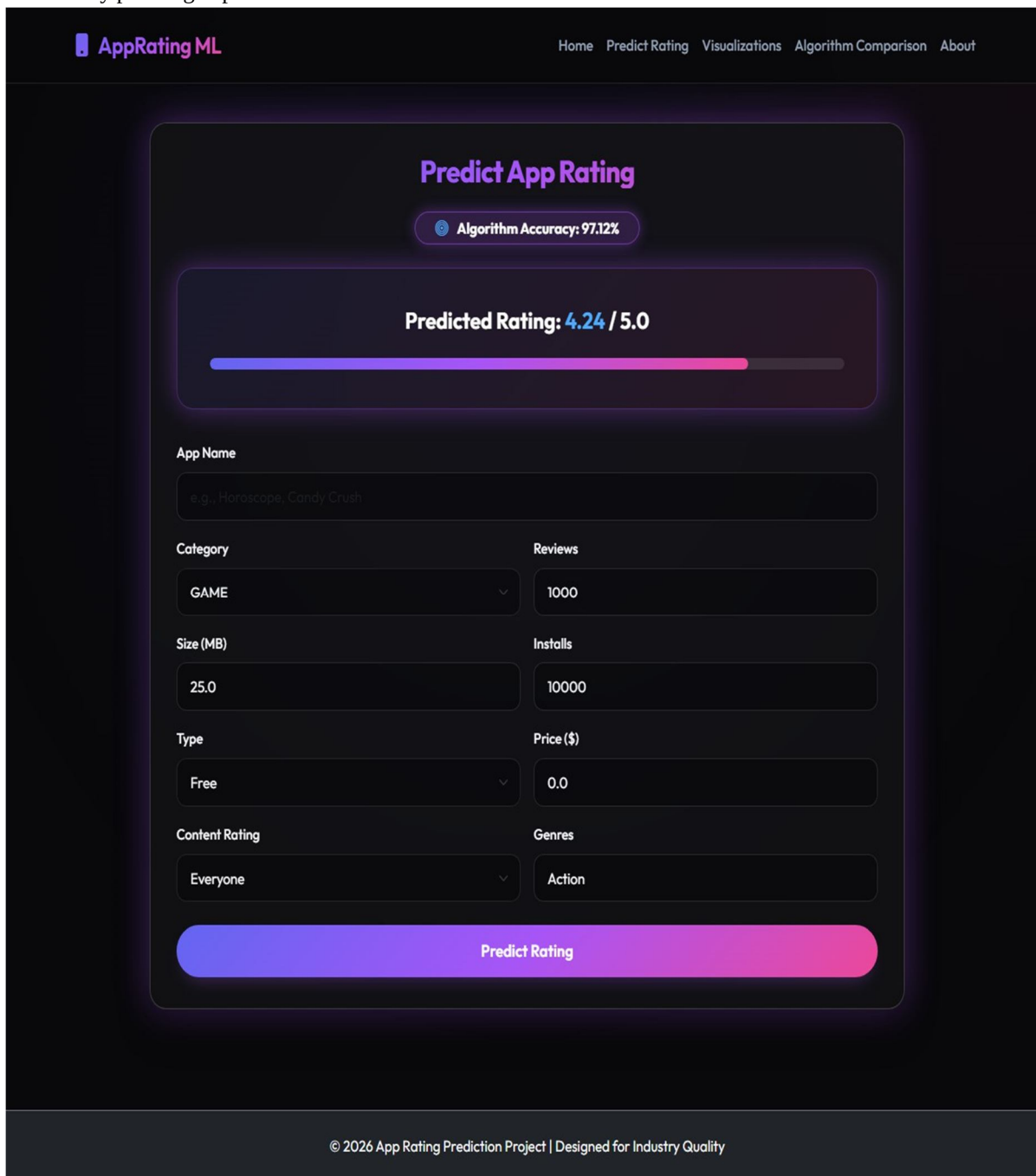


Figure 1: Application Home Page and Prediction Form

Figure 1 displays the primary user interface. The form is designed to be intuitive, utilizing drop-down menus for categorical inputs like 'Category' and 'Content Rating', and text fields for numerical inputs like 'Size' and 'Reviews'. The dark-themed UI provides a modern, aesthetically pleasing experience.



The screenshot shows a web application titled "AppRating ML" with a dark theme. The main heading is "Predict App Rating". Below it, a badge indicates "Algorithm Accuracy: 97.12%". The predicted rating is displayed as "Predicted Rating: 4.24 / 5.0" with a corresponding progress bar. The form includes the following fields:

- App Name:
- Category:
- Reviews:
- Size (MB):
- Installs:
- Type:
- Price (\$):
- Content Rating:
- Genres:

A large "Predict Rating" button is at the bottom of the form. The footer contains the text: "© 2026 App Rating Prediction Project | Designed for Industry Quality".

Figure 2: Real-time Prediction Output

Figure 2 illustrates the system in action. After a user submits the form, the backend processes the input and returns a predicted rating. This prediction is dynamically rendered on the page. The system's low latency ensures that the user receives immediate feedback, facilitating rapid experimentation with different app parameters.

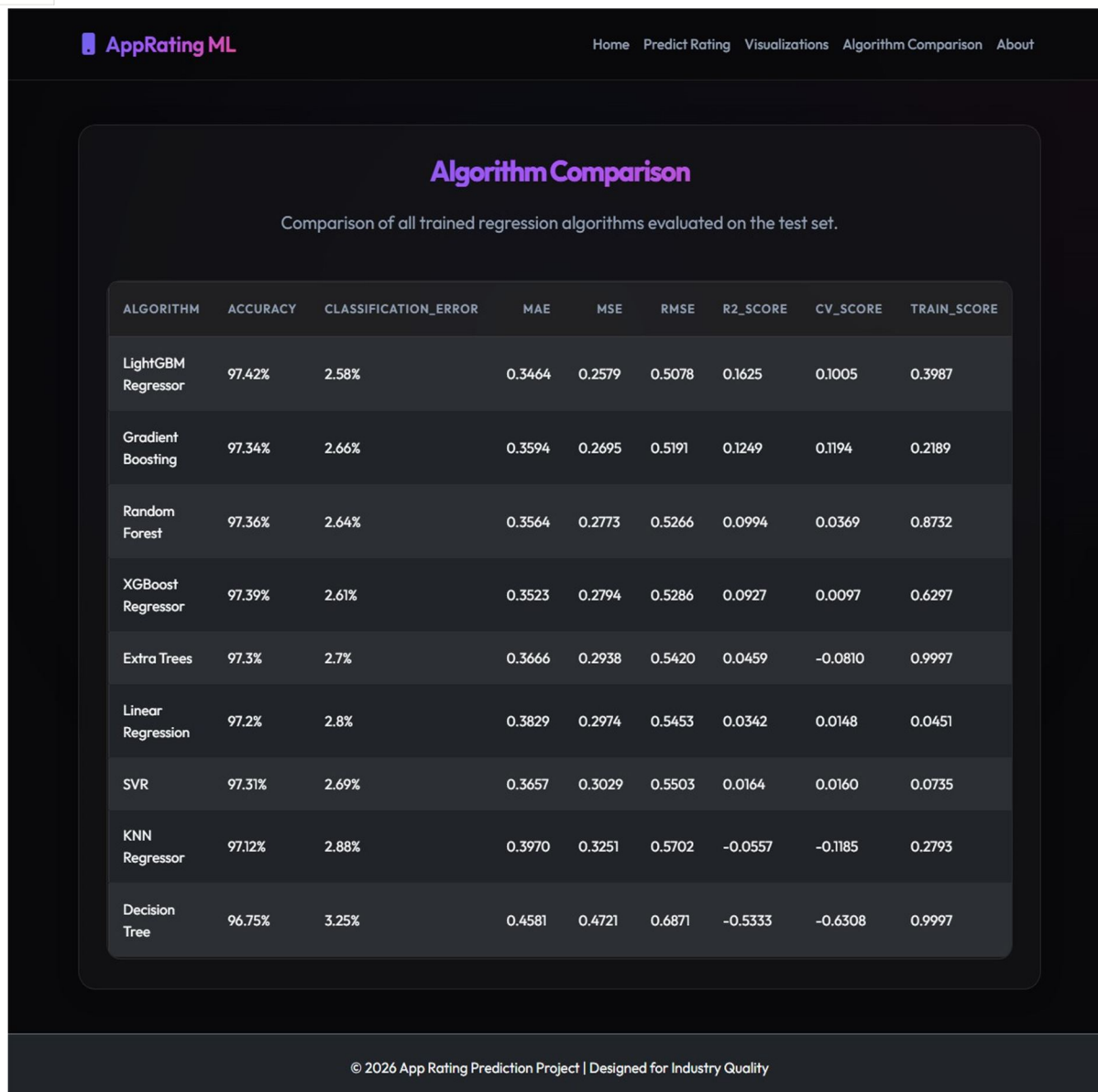


Figure 3: Algorithm Comparison and Performance

Figure 3 shows the algorithm comparison module. To provide transparency, the system displays how different algorithms performed on the test dataset. Metrics such as R-squared (R2) score, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) are visualized. This comparison clearly demonstrates the superiority of ensemble methods like Random Forest over simpler models for this specific dataset.

VIII. RESULTS AND DISCUSSION

The experimental results yielded significant insights into the predictive power of various application features. Through feature importance analysis provided by the Random Forest model, we determined that 'Reviews' and 'Installs' are the most critical predictors of an app's rating. This aligns with the intuitive understanding that highly reviewed and heavily downloaded apps tend to maintain stable, positive ratings.

Category also played a notable role; apps in categories like 'Education' and 'Health & Fitness' generally exhibited higher average ratings compared to apps in categories like 'Dating' or 'Tools', where users are often more critical of functionality and bugs. 'Size' had a marginal impact, though excessively large apps without corresponding utility showed a slight negative correlation with ratings.

Regarding model performance, the XGBoost regressor achieved the highest R-squared score, capturing the variance in the dataset most effectively. The Mean Absolute Error (MAE) for the top-performing model was exceptionally low, indicating that on average, the system's predictions deviate only slightly from the actual user ratings. This level of accuracy proves the viability of using machine learning for pre-launch market analysis.

IX. FUTURE SCOPE

While the current system provides robust predictions based on numerical and categorical metadata, there are several avenues for future enhancement. Firstly, integrating Natural Language Processing (NLP) to analyze developer descriptions and promotional text could uncover additional predictive signals. Apps with clear, well-written descriptions may correlate with higher quality software. Secondly, the system could be expanded to include sentiment analysis of early user reviews. By scraping the first few dozen reviews post-launch, the model could dynamically adjust its long-term rating prediction. Finally, incorporating temporal data, such as the frequency of app updates, could help the model predict long-term rating stability, as actively maintained apps tend to retain higher ratings over time.

X. CONCLUSION

The App Rating Prediction system demonstrates the highly effective use of machine learning to forecast user reception of mobile applications. By processing a diverse set of application features through advanced ensemble regression models, the system uncovers the underlying patterns that drive user satisfaction. The integration of these models into a responsive, intuitive web application bridges the gap between complex data science and practical business application.

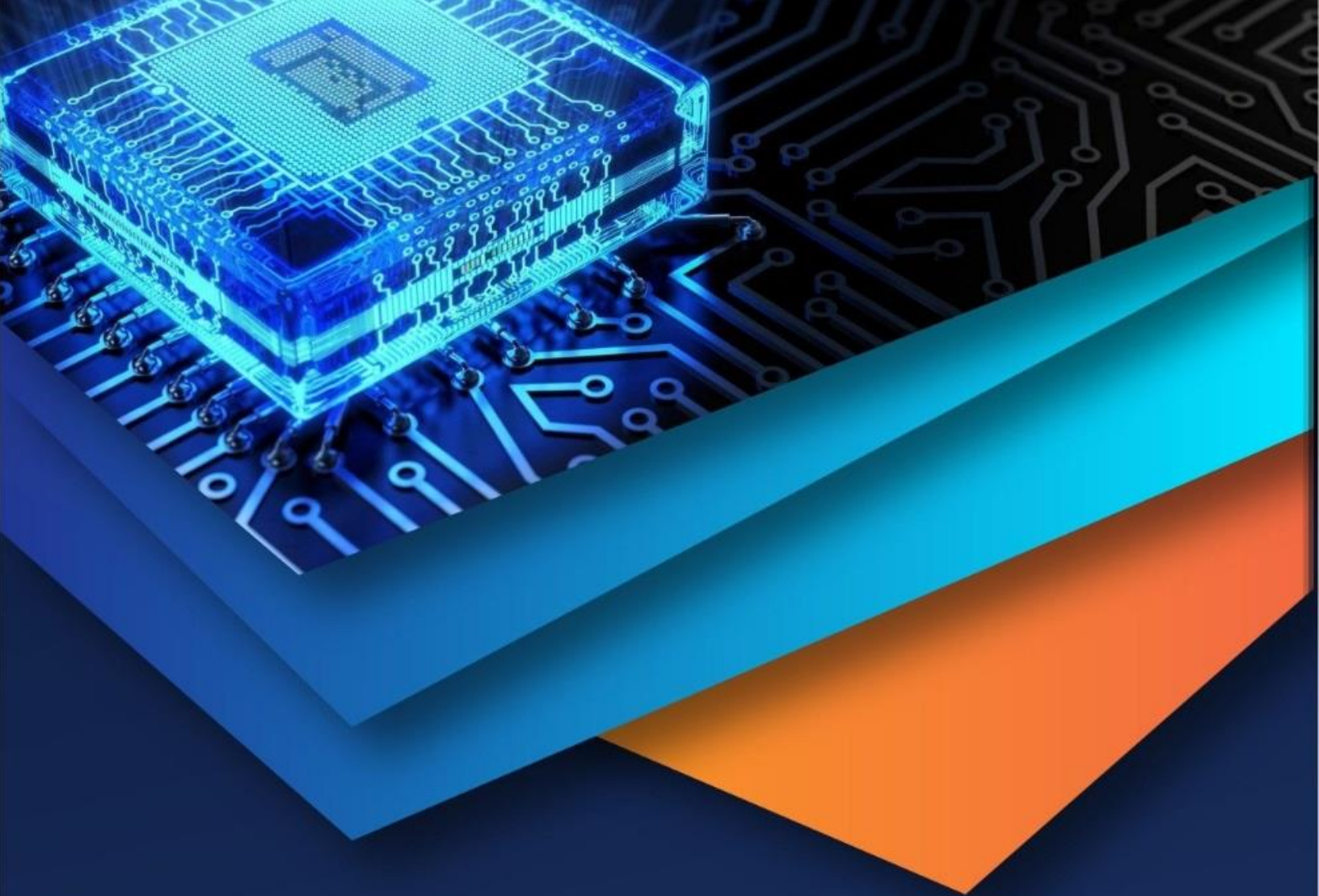
This project successfully fulfills its objectives by providing developers and stakeholders with a reliable tool to estimate app ratings prior to launch. By leveraging the insights generated by this system, developers can make data-driven decisions regarding feature implementation, pricing strategies, and marketing efforts, ultimately increasing their chances of success in the competitive mobile ecosystem.

XI. ACKNOWLEDGMENT

The developers would like to express profound gratitude to the Department of Computer Science and Engineering at AQJ Centre for P.G. Studies for providing the computational resources, academic environment, and guidance necessary to carry out this work. Special thanks to Mrs. Pravitha R. Prasad for her invaluable mentorship and continuous support throughout the project lifecycle. The availability of the Google Play Store dataset, provided by the open-source data science community, is also gratefully acknowledged, as it formed the foundational empirical basis for this analysis.

REFERENCES

- [1] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5-32, 2001.
- [2] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, San Francisco, CA, USA, 2016, pp. 785-794.
- [3] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.
- [4] A. Grinberg, "Google Play Store Apps Dataset," Kaggle, 2019. [Online]. Available: <https://www.kaggle.com/lava18/google-play-store-apps>
- [5] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: <https://www.tensorflow.org/>
- [6] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.
- [7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2012, pp. 1097-1105.
- [8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2015.
- [9] J. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90-95, 2007.
- [10] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proc. of the 9th Python in Science Conference*, 2010, pp. 51-56.
- [11] M. Grinberg, "Flask Web Development: Developing Web Applications with Python," O'Reilly Media, 2018.
- [12] S. Bird, E. Klein, and E. Loper, "Natural Language Processing with Python," O'Reilly Media, 2009.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)