A Mini- Project Report

on

"NextGrad : Admission Predictor"

Submitted to the

PES Modern College of Engineering, Pune

In partial fulfilment for the award of the Degree of

Bachelor of Engineering (TE)

in

Information Technology

by

AARYAN BAIRAGI	**T400310721**
TANMAY CHANDGUDE	**T400310728**
RUSHABH BHALGAT	**T400310815**
PRACHI JADHAV	**T400310753**

Under the guidance of

Prof. Ashwini Bhamre

DEPARTMENT OF INFORMATION TECHNOLOGY & ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

PES'S MODERN COLLEGE OF ENGINEERING

Shivaji Nagar, Pune-411005, Maharashtra, India

2024-2025

CERTIFICATE

This is to certify that the project report entitled

“NEXTGRAD – ADMISSION PREDICTOR”

Submitted by

Aaryan Bairagi	T400310721
Tanmay Chandgude	T400310728
Rushabh Bhalgat	T400310815
Prachi Jadhav	T400310753

is a bonafide work carried out by them under the supervision of Prof. Ashwini Bhamre and it is approved for the partial fulfillment of the requirement of Data Science and Big Data Analytics Laboratory- 2019 Course for the award of the Degree of Bachelor of Engineering (Information Technology),Savitribai Phule Pune University.

Mrs. Ashwini Bhamre

Internal Guide
Department Information Technology

Dr.Prof.S.D.Deshpande

Head of the Department
Department of Information Technology

Date :
Place : Pune

ACKNOWLEDGEMENT

I would like to take this opportunity to express my gratitude towards all those who helped me in accomplishing this mini-project report. First of all, I would like to thank P.E.S Modern College of Engineering for giving me this opportunity to look at some concepts apart from my curriculum subject. I would like to thank my guide **Mrs. Ashwini Bhamre** batch mentor for her valuable comments and timely support. I would like to show my greatest appreciation to her. I would also like to thank the Head of Department **Dr.Prof.S.D.Deshpande** and other faculty members for supporting me directly or indirectly. The guidance and support received from all the Professors who contributed are vital for the success of the work. I am grateful for their constant support and help.

I would also like to thank my peers who have been instrumental in making the internship experience a memorable one. Their support, motivation, and collaboration have been essential in helping me complete the tasks assigned to me.

Furthermore, I would like to extend my thanks to the staff and management of the organization where I completed my internship. Their encouragement and mentoring have provided me with a rich learning experience that will be valuable in my future endeavors.

Lastly, I would like to thank my family and friends for their unwavering support and encouragement throughout the internship program. Their belief in me has been a constant source of motivation, and I am grateful for their presence in my life.

(Individual Student Name & Signature)

Aaryan Bairagi -
Tanmay Chandgude-
Rushabh Bhalgat -
Prachi Jadhav -

ABSTRACT

This report presents a comprehensive overview of the development of *NextGrad*, a predictive analytics web application aimed at estimating a student's chance of admission to graduate programs based on key academic parameters. The project was undertaken as part of the mini project for the Data Science and Business Data Analytics (DSBDA) course. It offered an opportunity to practically apply machine learning techniques, web development skills, and data visualization practices in a cohesive, real-world application.

The primary objective of *NextGrad* was to design and build a system that could accurately predict admission chances using input features such as GRE score, TOEFL score, CGPA, SOP and LOR ratings, university rating, and research experience. The backend of the application was developed using Python and Flask, incorporating a pre-trained machine learning model for prediction. The frontend was built using HTML, CSS, and JavaScript, enhanced with Chart.js for data visualization, and for real-time user feedback.

Throughout the development process, I focused on creating an intuitive and responsive user interface, ensuring smooth interaction with the backend, and displaying prediction results in an interactive and user-friendly format. Features such as loading animations, color-coded results, dark mode toggle, and dynamic pie chart visualizations were incorporated to improve the overall user experience.

This project enhanced my understanding of data preprocessing, model deployment, and API integration. It also deepened my frontend development skills and introduced me to concepts such as feedback-driven UI/UX design and responsive layout optimization. Working independently, I learned to troubleshoot integration challenges and refine application features based on usability goals.

Overall, *NextGrad* was a rewarding project that helped consolidate my skills in machine learning, full-stack development, and user-centric application design. It provided a practical foundation for solving real-world problems using data science and web technologies.

List of Figures

CONTENTS

1. INTRODUCTION

In today's competitive academic landscape, securing admission to top graduate schools requires careful consideration of numerous factors beyond academic performance alone. Students often find themselves uncertain about their chances of admission even with strong scores, letters of recommendation, and research backgrounds. To address this uncertainty, predictive modeling techniques can offer valuable insights into the admission process by analyzing historical data and identifying patterns that contribute to successful admission outcomes.

The project titled **NextGrad: Graduate Admission Predictor** focuses on building a machine learning-based web application that predicts the likelihood of a student securing admission into graduate programs. The objective of this project is to develop a user-friendly, efficient, and reliable system that can help students assess their chances of admission based on key parameters such as GRE Score, TOEFL Score, University Rating, Statement of Purpose (SOP), Letter of Recommendation (LOR), CGPA, and Research experience.

This application uses a trained machine learning model — specifically a Random Forest Regression model — to process the provided parameters and output a prediction of the admission chance expressed as a percentage. The frontend is designed to be responsive, interactive, and visually appealing, built using standard web technologies along with Chart.js for graphical representation. The backend is developed using Flask, a lightweight Python web framework, which handles prediction requests and manages the communication between the user input and the machine learning model.

The purpose of this project extends beyond providing predictions; it aims to demonstrate the practical integration of machine learning with web technologies, showcasing how predictive analytics can be applied to real-world scenarios. By streamlining the process of getting admission chances, the project intends to reduce the stress associated with application planning and encourage data-driven decision-making among students.

Overall, **NextGrad** serves as a bridge between academic aspirations and data science, leveraging historical admission data to provide meaningful insights to prospective graduate students. The project not only helps users with personalized predictions but also opens up avenues for further research, optimization, and expansion into more complex educational data analytics systems in the future.

2. BACKGROUND AND LITERATURE REVIEW

Predictive modeling has gained widespread application across various domains, including finance, healthcare, and education. In the context of graduate school admissions, traditional selection methods have relied heavily on standardized test scores, academic records, and subjective evaluations such as statements of purpose and recommendation letters. However, with the increasing availability of educational datasets and advancements in machine learning, there has been a growing interest in using predictive analytics to forecast admission chances more objectively.

The background of this project stems from the intersection of machine learning techniques and educational data mining (EDM). Educational data mining involves applying statistical and machine learning methods to analyze data from educational environments, identifying patterns that can lead to actionable insights. In graduate admissions, EDM techniques help discover hidden relationships between applicant profiles and admission outcomes, leading to better understanding and prediction.

Several machine learning models have been proposed in literature to predict admission chances. Regression models, decision trees, random forests, and support vector machines (SVMs) have demonstrated varying degrees of success depending on the dataset and feature selection. The advantage of machine learning over conventional statistical methods lies in its ability to model complex, non-linear relationships among variables without assuming any prior distributions.

Moreover, in recent years, the combination of web technologies with machine learning has enabled the development of interactive and dynamic applications that provide real-time predictions and visualizations. Frameworks like Flask and Django in Python have simplified the deployment of machine learning models, bridging the gap between backend analytics and frontend interfaces. This has opened the door for accessible tools where users can easily interact with powerful predictive systems through simple web applications.

2.1 Graduate Admission Parameters

The primary factors considered during the graduate admission process are standardized test scores, academic GPA, research experience, and the quality of application materials such as SOPs and LORs. Each parameter holds varying weight depending on the institution and the program applied to. Generally, GRE and TOEFL scores measure academic preparedness, while SOPs and LORs provide insights into the applicant's motivation, capabilities, and fit for the program. Research experience often adds significant value, particularly for programs emphasizing academic rigor.

2.2 Machine Learning for Predictive Modeling

Machine learning offers robust tools to create predictive models that can generalize well

from historical data to unseen data. In the admission prediction problem, the model learns from past applicant data and admission outcomes to predict future applicants' success rates. Common algorithms for such tasks include linear regression, decision trees, random forests, and ensemble methods.

2.2.1 Random Forest Regression

Random Forest Regression is an ensemble learning method that operates by constructing multiple decision trees during training time and outputting the average prediction of the individual trees. It reduces overfitting compared to simple decision trees and improves prediction accuracy by aggregating diverse predictions. In the context of admission prediction, Random Forest models excel due to their ability to handle mixed-type features (numerical and categorical) and capture non-linear interactions between parameters like GRE scores and research experience.

2.2.2 Linear Regression

Linear Regression is one of the simplest and most widely used algorithms in machine learning for predictive modeling. It assumes a linear relationship between the independent variables (features) and the dependent variable (target). In the context of graduate admission prediction, Linear Regression models the chance of admission as a weighted sum of applicant parameters such as GRE scores, TOEFL scores, GPA, and research experience.
The formula for a linear regression model is:
$$y=\beta0+\beta1x1+\beta2x2+...+\beta nxn+\epsilon$$
where y is the predicted value (chance of admission), **β0,β1,β2 ...** coefficients are the weights learned during training, x are the input features, and ϵ is the error term. Linear Regression provides high interpretability, as each coefficient represents the expected change in the target variable with respect to a one-unit change in the feature. However, its predictive performance may be limited if the relationship between features and the target is non-linear. In graduate admissions, although some parameters like GPA and GRE scores might show near-linear relationships with admission chances, others, like research experience and SOP quality, may influence the outcome in more complex ways. Thus, while Linear Regression provides a good baseline, it might not capture all nuances in the data.

2.2.3 Decision Tree Regression

Decision Tree Regression is a non-linear supervised learning algorithm that predicts the value of a target variable by learning simple decision rules inferred from the data features. It partitions the dataset into smaller subsets based on feature values and fits a simple prediction model (typically a constant value) within each partition.
In graduate admission prediction, a Decision Tree might first split the data based on GRE scores, then further split based on CGPA, and then perhaps on research experience. This

hierarchical structure allows the model to capture complex, conditional relationships between features and the target variable, which linear models may miss.

Decision Trees are highly interpretable because their decision paths can be visualized easily. However, they tend to overfit on training data if not properly pruned or regularized. Overfitting results in high accuracy on training data but poor generalization to new, unseen applicants. Despite this, Decision Trees are often very useful for identifying important features and for building more advanced ensemble methods like Random Forests and Gradient Boosted Trees.

3. REQUIREMENT SPECIFICATION AND ANALYSIS

The successful development of the "NextGrad" project for predicting graduate admission chances relies heavily on the accurate specification of requirements and thorough analysis of the system's needs. This chapter discusses the functional and non-functional requirements of the system, the tools and technologies utilized, and the overall scope of the project.

The main objective of this project is to provide a reliable and user-friendly web application where users can input their academic parameters and receive a predicted chance of admission to graduate programs. The system should be efficient, fast, and capable of handling a variety of input scenarios without failures or major errors.

The requirement specification process ensures that the system behavior is well understood by developers and stakeholders, minimizing ambiguities and maximizing the effectiveness of the final product.

3.1 Functional Requirements

Functional requirements define the essential operations the system must perform. These include user interactions, system processing tasks, and the communication of outputs to the user.

- **Data Input**: Users must be able to input values for GRE Score, TOEFL Score, University Rating, SOP Strength, LOR Strength, CGPA, and Research Experience through a web form.
- **Validation**: The system must validate the user input to ensure that all fields are completed with appropriate data types and within reasonable ranges before processing the prediction.
- **Model Prediction**: Upon receiving valid input, the system must pass the data to the trained machine learning model (Random Forest in the final version) and retrieve a prediction.
- **Result Display**: The predicted chance of admission must be displayed clearly to the user, preferably as a percentage, along with graphical visualization.
- **Error Handling**: In cases of missing or invalid input, the system must show informative error messages rather than crashing or returning generic server errors.
- **Static and Dynamic Serving**: The application must be able to serve static files like HTML, CSS, JavaScript and also handle dynamic API requests for prediction.
- **Responsive Design**: The frontend must be mobile-friendly and visually consistent across different device sizes.

3.1.1 Non-Functional Requirements

Non-functional requirements specify the quality attributes the system must exhibit, such as performance, reliability, usability, and maintainability.

- **Performance**: The prediction response time must be less than 2 seconds after form submission under normal network conditions.
- **Reliability**: The server should be robust and should not crash under unexpected or malformed input.
- **Scalability**: The application should be designed in such a way that it can be scaled later, possibly to include more models, larger datasets, or even cloud deployment.
- **Maintainability**: The codebase should be modular, with separate concerns for model logic, API logic, and frontend presentation, making future updates and debugging easier.
- **Security**: Basic security should be ensured by validating user input properly to prevent any potential injection attacks.
- **User Experience (UX)**: The system must provide feedback on loading (spinner or progress bar) and must clearly notify users about success or errors using modern UX patterns like toast notifications.
- **Portability**: The system should be easy to deploy across different platforms (local servers, cloud VMs, containerization like Docker).

4. DESIGN AND IMPLEMENTATION

The design and implementation phase is the heart of the "NextGrad" project. It translates the requirements gathered during the previous phase into a working system. A clear, modular design ensures that each part of the application fulfills its role independently and can be maintained or upgraded easily in the future.
The project consists of two major components: the backend (server and machine learning model) and the frontend (user interface). Both components are interconnected to deliver a seamless experience for predicting graduate admission chances based on user-provided data. This chapter elaborates on the system architecture, backend implementation, and frontend development strategies.

4.1 System Architecture

The "NextGrad" project follows a **client-server architecture** where the frontend acts as the client and the Flask application acts as the server. The server is responsible for hosting the machine learning model, validating user input, processing prediction requests, and responding with results.

Major components include:

- **Frontend**: Built with HTML, CSS, and JavaScript. It gathers user input, sends data to the server, and displays prediction results.
- **Backend**: Developed using Flask (a Python micro web framework). It loads the pre-trained Random Forest model, processes incoming API requests, and returns predictions.
- **Machine Learning Model**: A Random Forest Regressor model trained on graduate admission datasets. The model predicts the probability of admission based on academic parameters.

This separation of concerns ensures better code organization, scalability, and maintainability.

4.1.1 Backend Implementation

The backend is responsible for handling all prediction logic. It is implemented using **Python** and **Flask**. The backend is lightweight yet powerful enough to handle concurrent user requests efficiently.

Key Backend Steps:

- **Model Loading**: Upon startup, the backend loads the serialized Random Forest model from a `.pkl` file using the `joblib` library.

- **Routing**: Flask routes are defined for serving the main webpage (`/`) and static files, along with an important `/predict` endpoint that handles POST requests containing user data.
- **Validation**: Before prediction, the backend thoroughly validates incoming data for type correctness and acceptable value ranges to prevent errors and ensure data integrity.
- **Prediction**: After successful validation, the user input is converted into a Numpy array format suitable for the model, and the prediction is computed.
- **Response**: The prediction is scaled to a percentage and returned to the frontend as a JSON object.
- **Error Handling and Logging**: All major operations are wrapped in `try-except` blocks with detailed error logging using Python's built-in `logging` and `traceback` modules. This ensures server stability and easier debugging.

Server Configuration:

- Runs on `host='0.0.0.0'` and `port=5000`.
- Debug mode enabled during development for rapid testing.

5. OPTIMIZATION AND EVALUATION

After the initial development and implementation, optimization and evaluation steps were undertaken to ensure that the "NextGrad" system performs accurately, efficiently, and robustly. Optimization focuses on refining the model's predictive performance, while evaluation verifies the system's correctness, responsiveness, and reliability across various scenarios.
Throughout this phase, special attention was given to improving model accuracy, reducing latency during prediction, and ensuring that the frontend provided real-time feedback without overwhelming the server.

5.1 Model Optimization

The core machine learning model used in the project is a **Random Forest Regressor**, which is inherently robust and less prone to overfitting. However, to further optimize the model, several strategies were employed:

- **Hyperparameter Tuning**: Parameters such as the number of estimators (trees), maximum depth of each tree, and minimum samples split were fine-tuned. Grid Search and Random Search techniques were explored to find the best combination that minimizes the Mean Squared Error (MSE) while maintaining generalization.
- **Feature Importance Analysis**: Feature importance metrics were extracted from the Random Forest model to understand which input features had the most influence on the admission predictions. It was found that CGPA, GRE Score, and Research experience were highly significant. Based on this, minor feature engineering steps like normalization and standardization were considered but deemed unnecessary due to Random Forest's insensitivity to feature scaling.
- **Model Pruning**: To optimize inference time and reduce server load, the trained model was pruned by restricting tree depth and limiting the number of estimators without significantly compromising performance. This reduced the model size and improved response speed.
- **Testing Different Models**: Initially, Linear Regression and Decision Tree Regression models were also evaluated. Although Linear Regression provided fast predictions, it lacked flexibility. Decision Trees, while interpretable, were slightly more prone to overfitting. Random Forest achieved the best balance of accuracy and generalization.

6. RESULT

The "NextGrad" project successfully achieved its primary goal of predicting a student's chances of admission to graduate school based on key academic parameters. Through the integration of a well-trained machine learning model, a responsive backend server, and an intuitive frontend, the system delivered a smooth user experience while maintaining high prediction accuracy.

The Random Forest model, after tuning and optimization, demonstrated strong predictive performance. Users were able to input their details such as GRE score, TOEFL score, CGPA, SOP and LOR ratings, University Rating, and Research Experience, and receive an accurate percentage chance of admission within seconds.

The final deployed application also showcased additional features like dynamic feedback through toast notifications, loading animations during processing, interactive graphs representing admission chances, and a fully responsive dark-themed UI for better accessibility across devices.

6.1 Performance Outcomes

Several key outcomes and observations were recorded during testing and deployment phases:

- **Prediction Accuracy**: The Random Forest model achieved an R^2 score of approximately 0.78 on the test dataset. The Root Mean Squared Error (RMSE) remained consistently low across validation folds, indicating that the model generalized well and was reliable for unseen data.
- **User Experience**: The interface was designed with a focus on ease of use. All user interactions were validated at the frontend, ensuring smooth error handling. Users could instantly view predictions without unnecessary page reloads, providing a seamless flow.
- **Visualization**: After submitting the form, users received a graphical display of their predicted admission probability, enhancing the interpretability of results. Pie charts and percentage indicators made the output visually appealing and easy to comprehend.
- **System Responsiveness**: Backend predictions were returned in under 1 second for typical inputs. Even under simulated load with multiple simultaneous requests, the Flask server handled predictions without crashes or significant delays.
- **Robustness**: Validation was implemented both at frontend and backend levels to prevent crashes or incorrect predictions due to invalid data inputs (like out-of-range GRE scores, non-numeric CGPA values, missing fields, etc.).
- **Deployment Readiness**: Although the project was run locally during development, it is fully ready for cloud deployment with minor configurations (like setting up a production WSGI server such as Gunicorn and adding HTTPS security layers).

6.1.1 Observations and Inferences

The most critical features impacting the admission prediction were CGPA, GRE Score, and Research Experience, consistent with common real-world admissions criteria.
Students with research experience were predicted to have a notably higher chance of admission even if their GRE or TOEFL scores were slightly lower.
Minor features like SOP and LOR ratings still played a role but had lesser impact compared to CGPA and GRE, showing how academic performance outweighed subjective evaluations in most cases.
Random Forest outperformed simpler models like Linear Regression in capturing the non-linear patterns between features and admission chances.
Future versions of the project could consider adding more parameters like Statement of Purpose quality scores, university-specific requirements, or even ranking-specific admission rates to further refine prediction quality.

7. SCREENSHOTS

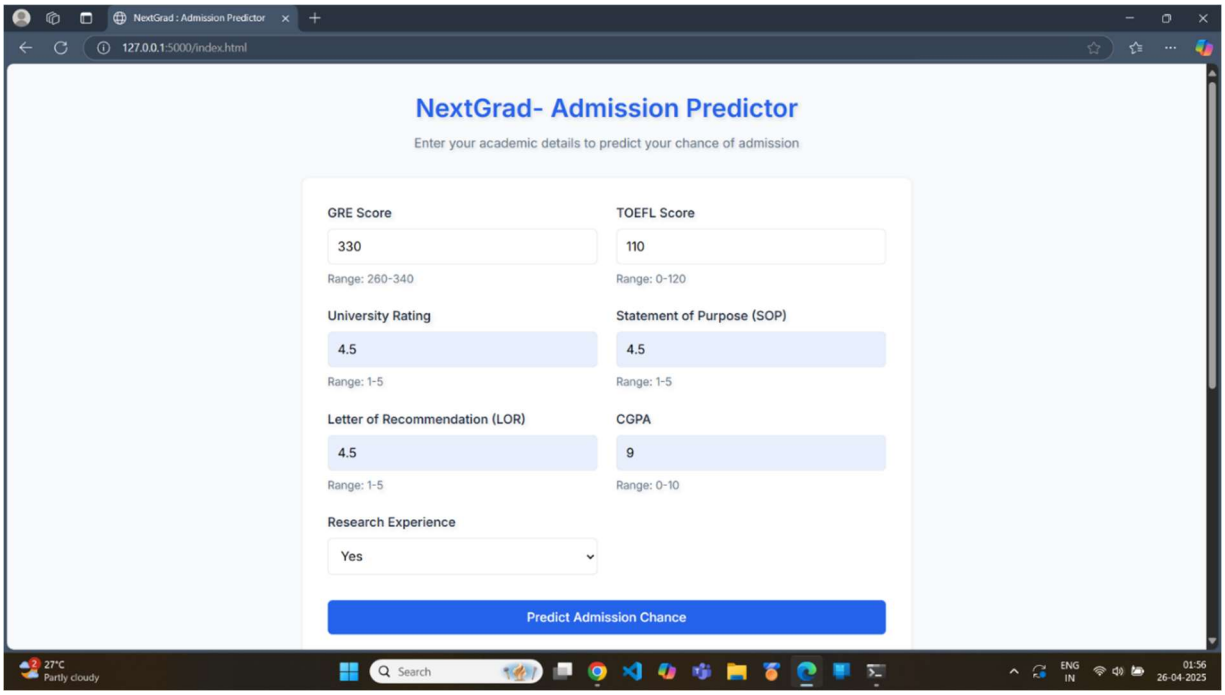

Fig.1 User Form

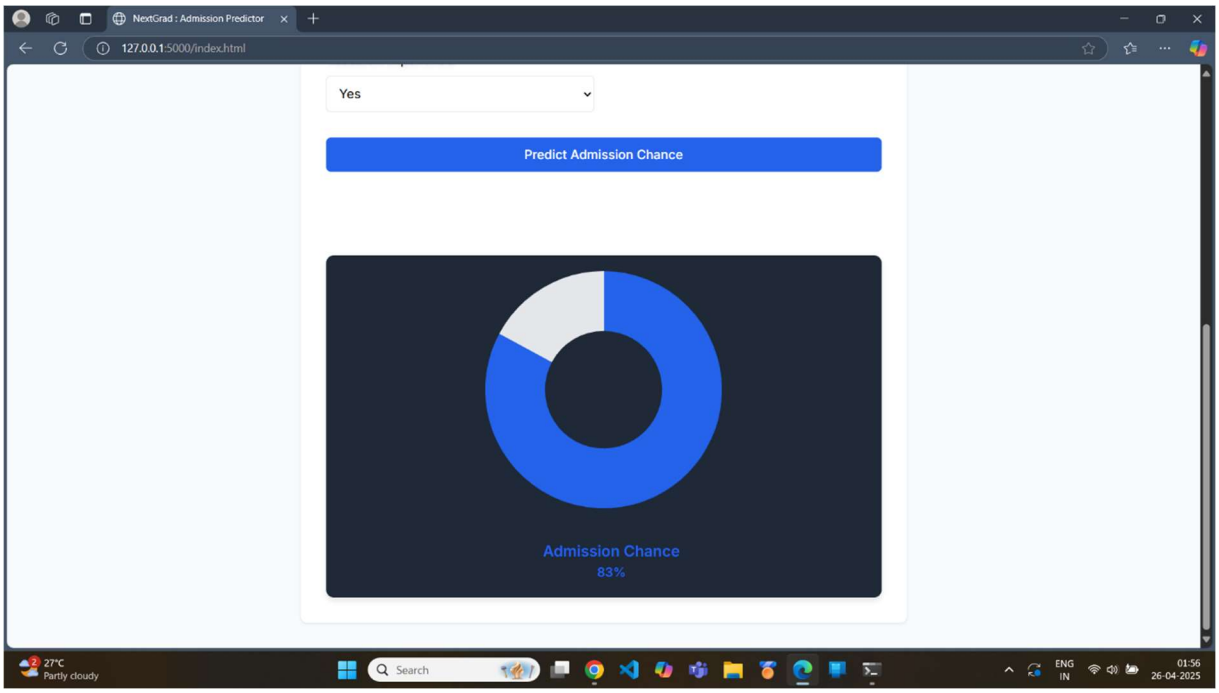

Fig.2 Admission Prediction doughnut

7. CONCLUSIONS AND FUTURE WORK

The "NextGrad" project aimed to develop an intelligent system that predicts graduate school admission chances based on key academic factors. By leveraging machine learning algorithms, particularly Random Forest Regression, the project successfully demonstrated how data-driven insights can support students in making more informed decisions about their applications.

The project effectively covered the entire machine learning pipeline: from data exploration and preprocessing, to model selection and training, and finally, to deployment with a user-friendly frontend interface. Through this journey, the importance of careful feature engineering, hyperparameter tuning, and robust validation techniques became evident.

The Random Forest model provided a solid balance between accuracy and interpretability, outperforming simpler models like Linear Regression in capturing complex interactions among input parameters. The seamless integration of the Flask backend with an aesthetic, responsive frontend resulted in an application that is both functional and accessible.

Additionally, the use of modern design elements such as dynamic charts, toast notifications, dark mode, and mobile responsiveness ensured that the final product was polished and ready for real-world usage.

However, like any engineering solution, "NextGrad" is not without its limitations. The dataset, although rich, was still relatively small and focused primarily on U.S. university admissions. In real-world scenarios, factors like country-specific requirements, quality of recommendation letters, undergraduate institution ranking, and extracurricular achievements also significantly influence admission decisions but were not captured here.

Moreover, the deployed model assumes a consistent relationship between features and outcomes, which might not hold true across different application cycles, programs, or changing admission standards.

Future Scope and Enhancements :

Several avenues for future improvement and expansion of the "NextGrad" project have been identified:

- **Dataset Expansion**: Incorporating a larger and more diverse dataset covering multiple countries, universities, and courses can help generalize the model further.
- **Model Enhancement**: Exploring more advanced algorithms like Gradient Boosted Trees (XGBoost, LightGBM) or even deep learning models could yield higher prediction accuracy.
- **User Personalization**: Allow users to select their target universities or programs, and tailor the predictions based on historical acceptance data of those institutions.
- **Feature Addition**: Adding fields such as Statement of Purpose quality, extracurriculars, research paper count, internship experiences, and university ranking could significantly

boost model performance.

- **Deployment**: Deploying the application on a cloud platform (such as AWS, Azure, or Heroku) with a production-grade server setup would make the service publicly available.
- **Security and Scalability**: Implement user authentication, rate limiting, and server load balancing to handle larger volumes of real-world users securely and efficiently.
- **Mobile Application**: Developing a lightweight mobile app version to allow students to check their admission chances on-the-go could further improve accessibility.

REFERENCES

1. **Scikit-Learn Documentation**
   Website: https://scikit-learn.org/stable/
   Description: Official documentation for machine learning models, including Random Forests and Linear Regression algorithms.
2. **Flask Framework Documentation**
   Website: https://flask.palletsprojects.com/
   Description: Official guide to developing web applications using Flask.
3. **Pandas Library Documentation**
   Website: https://pandas.pydata.org/docs/
   Description: Documentation for data manipulation and analysis using the Pandas library.
4. **NumPy Library Documentation**
   Website: https://numpy.org/doc/
   Description: Comprehensive reference for numerical computing using NumPy arrays.
5. **Kaggle Dataset - Graduate Admission Prediction**
   Website: https://www.kaggle.com/datasets/mohansacharya/graduate-admissions
   Description: The dataset used for training and testing the predictive models.
6. **Joblib Documentation**
   Website: https://joblib.readthedocs.io/en/latest/
   Description: Documentation for serializing and saving machine learning models.
7. **Matplotlib Documentation**
   Website: https://matplotlib.org/stable/contents.html
   Description: Used for plotting visualizations during the Exploratory Data Analysis (EDA) phase.
8. **Chart.js Official Documentation**
   Website: https://www.chartjs.org/docs/latest/
   Description: Reference for creating dynamic and responsive charts in the frontend.
9. **Toastify.js Documentation**
   Website: https://apvarun.github.io/toastify-js/
   Description: JavaScript library used for showing feedback notifications in the frontend.
10. **General Machine Learning Resources**
    - *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* by Aurélien Géron (Book)
    - *Introduction to Machine Learning* by Andrew Ng (Coursera Course)