

# University Admission Prediction Application

## Problem Definition

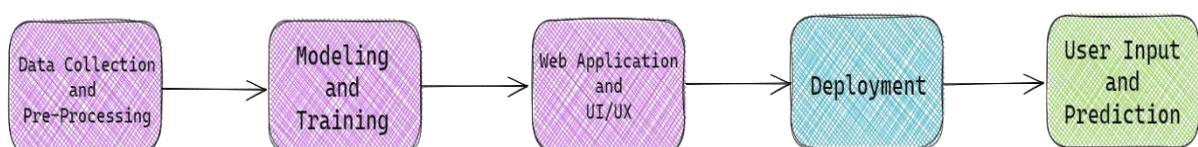
The process of applying to graduate programs at universities can prove to be quite daunting and highly competitive. As part of the application package, prospective students are required to submit a range of documents such as GRE and TOEFL scores, a Statement of Purpose, Letters of Recommendation, as well as any relevant research papers. These documents serve as vital parameters for universities in selecting suitable candidates for their upcoming programs. A lot of times, students are not aware about their chances of getting admit to a particular course. Having access to this information would make it easier for them to prioritize certain universities over others and save on time and application fee.

The machine learning application addresses this situation to simplify the admission process for students by utilizing predictive algorithms to determine the chances of being accepted into each university. The program evaluates all the relevant criteria, including GRE and TOEFL scores, Statement of Purpose, Letters of Recommendation, research papers, and university rankings, to produce a probability value as output. This probability value is an indicator of your likelihood of being accepted into your desired program/university. By utilizing this advanced tool, you can gain valuable insight into the admissions process and make informed decisions about your future academic pursuits.

The end users of this application would be prospective students looking to pursue higher education in United States of America.

## System Design

We chose to build our application using web development technologies, such as HTML and CSS, along with Flask, a backend framework to handle the server-side functionality, to keep our codebase in Python and for faster iteration. Below is a simple diagram of our application system:



The key components of the system include:

**Data collection and pre-processing:** This step involves importing the data cleaning and pre-processing the data to remove inconsistencies or missing values, and applying feature engineering techniques such as scaling, normalization, or categorical encoding to prepare the data for input into the ML model.

**Modeling:** We trained our machine learning models using the cleaned and preprocessed data and evaluated its performance using appropriate evaluation metrics. Different algorithms such as linear regression, Support Vector Regressor, XGBoost Regressor and CatBoost Regressor were used and evaluated for this project. We utilized libraries such as Pandas, NumPy, and Scikit-learn for our modeling.

**Web application:** Next step involved building the web application using web development technologies such as HTML, CSS, and a backend framework Flask to handle the server-side functionality. The trained machine learning model is integrated into the backend of the application to process input data provided by the user.

**User experience (UX):** This step involved designing a user-friendly interface for the web application, ensuring smooth user interactions, and providing meaningful feedback to users on the prediction results.

**Deployment:** The final step involves deploying the web application on a web server or cloud platform AWS to make it accessible to users worldwide. This involves configuring the web server, setting up the database, and ensuring the security and scalability of the application.

We chose Flask as the backend framework due to its simplicity and flexibility for building web applications with Python, which is the language used in building our machine learning model. The choice of machine learning algorithms has been based on their performance on the test data. The deployment AWS has been used due to its ease of use, cost-effectiveness, and scalability.

We have chosen these technologies based on their suitability for the project requirements and expertise of the team members.

## Machine Learning Component

The machine learning component of our application is powered by a regression model that predicts university admission likelihood based on various student-related features such as test scores, GPA, and extracurricular activities. The model is trained on a medium-sized dataset of past university admissions with relevant student information. We used a variety of data pre-processing techniques such as scaling and one-hot encoding to prepare the data for modeling. We iteratively developed the model by experimenting with various algorithms and hyperparameters, evaluating its performance with metrics such as mean squared error and R-squared.

Our final model achieved the lowest RMSE on the test set, indicating a strong fit to the data.

**The ML model:** The university admission prediction application is powered by a regression model (CatBoost Regressor) that predicts the probability of a student getting admitted to a particular university based on various features such as their academic performance, standardized test scores, extracurricular activities, etc. The model is based on a gradient boosting algorithm that has been implemented using the Scikit-learn library.

**The data:** The model is trained on a dataset that contains information about students who have applied to various universities in the past. The dataset includes features such as GPA, GRE/TOEFL scores, essay scores, recommendation letters, and other relevant information. The dataset has been obtained from Kaggle which in turn was collected from various sources such as university websites, government databases, and student forums.

**The development of the model:** The development of the model has been an iterative process that involved several stages of data pre-processing, feature engineering, and model tuning.

- Initially, the data was cleaned and pre-processed to remove any missing values and outliers.
- Then, various features were engineered based on domain knowledge and intuition.
- After that, the model was trained on the initial dataset and evaluated using various performance metrics such as mean absolute error (MAE) and root mean squared error (RMSE).
- Based on the evaluation results, the model was fine-tuned by tweaking various hyperparameters such as learning rate, max depth, and number of estimators.

This iterative process was repeated several times until the model achieved satisfactory performance on both the training and validation datasets. The final model was then deployed as part of the university admission prediction application.

## System Evaluation

We have put in efforts to validate and evaluate the performance of our system during various stages of its development. During the data collection and preprocessing phase, we ensured that reliable sources of data were used and that inconsistencies or missing values were removed to ensure data quality. We also applied feature engineering techniques, such as scaling, normalization, and categorical encoding, to prepare the data for input into the machine learning model.

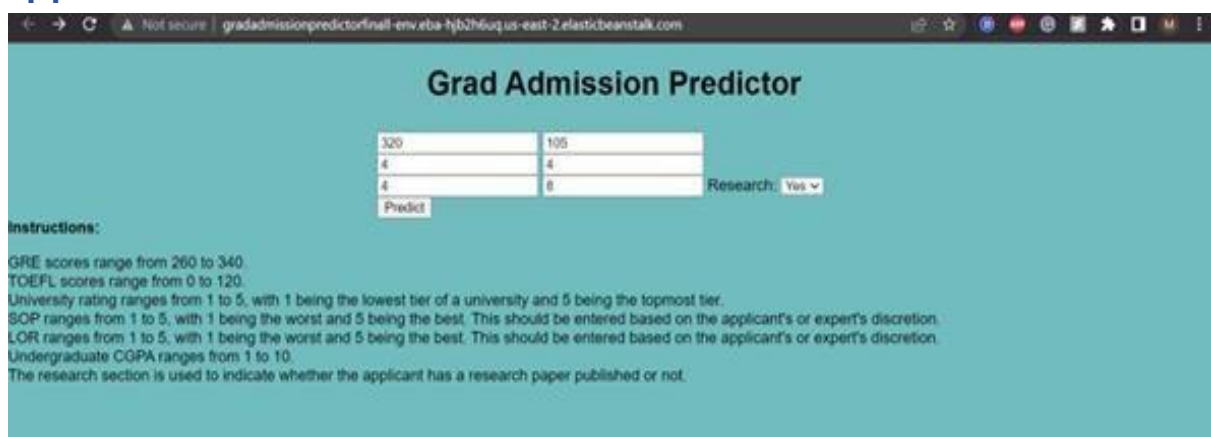
For model training, we tried using different algorithms, such as linear regression, Support Vector Regressor, XGBoost Regressor and CatBoost Regressor and then evaluated their performance using appropriate evaluation metrics, such as RMSE. We have also compared our model predictions to a random sample from the data. This helped us assess the effectiveness of a particular model in making predictions and fine-tune it for better performance.

Once the model was trained and validated, it has been integrated into a web application using web development technologies and backend frameworks. We tried to ensure that the application was functional and user-friendly, with proper handling of input data provided by the user.

In terms of limitations, we must acknowledge that the accuracy of our predictions depends on the quality and completeness of the data collected. If the data we have used for training and validation is not any longer representative of the current admission trends or lacks certain important features, the predictions from our model may not be accurate. Additionally, our choice of machine learning algorithms used could impact the system's performance, and there may be limitations in terms of scalability and security during the deployment process on a web server or cloud platform.

To address these limitations, we could have explored using a few more advanced machine learning techniques or incorporating additional data sources to improve the accuracy of predictions. We could have also conducted extensive testing and validation of the application after deployment to identify and address any potential issues. To keep it up-to-date with changing admission trends and technologies, regular updates and maintenance of the system would also be necessary.

## Application Demonstration



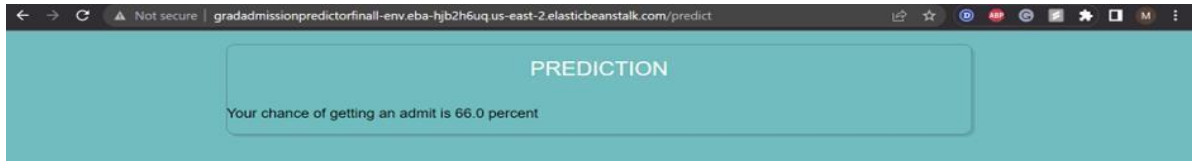
The screenshot shows a web browser window with the URL `gradadmissionpredictorfinal-env.eba-hjb2h6uq.us-east-2.elasticbeanstalk.com`. The page title is "Grad Admission Predictor". The form contains the following fields and values:

GRE	TOEFL	University Rating	SOP	LOR	Undergraduate CGPA	Research
320	105	4	4	4	8	Yes

Below the form is a "Predict" button. The "Instructions" section provides the following details:

- GRE scores range from 260 to 340.
- TOEFL scores range from 0 to 120.
- University rating ranges from 1 to 5, with 1 being the lowest tier of a university and 5 being the topmost tier.
- SOP ranges from 1 to 5, with 1 being the worst and 5 being the best. This should be entered based on the applicant's or expert's discretion.
- LOR ranges from 1 to 5, with 1 being the worst and 5 being the best. This should be entered based on the applicant's or expert's discretion.
- Undergraduate CGPA ranges from 1 to 10.
- The research section is used to indicate whether the applicant has a research paper published or not.

### User Interface Page



### Output Page

The tool is a standalone web application built using Flask API, CSS, and HTML which is deployed on AWS Beanstalk, providing a reliable and scalable solution for users. We chose Flask API to process data and perform more complex calculations than a web interface. We can scale this to handle many requests by distributing requests across multiple instances, and we can integrate this with other systems or applications easily. We can also build automated workflows or scripts that interact with the solution by using Flask API.

Prospective graduate students will be the primary users of this application. They will interact via a standalone web application where they enter the required parameters. The application has 7 user input features to predict the probability of graduate admission into a particular school. Users can input standardized scores like GRE, TOEFL, SOP standard scores, and LOR standard scores along with research experience based on the instructions given below.

Below are the basic Instructions on how to use the application:

- GRE scores range from 260 to 340.
- TOEFL scores range from 0 to 120.
- University rating ranges from 1 to 5, with 1 being the lowest tier of a university and 5 being the topmost tier.
- SOP ranges from 1 to 5, with 1 being the worst and 5 being the best. This, should be entered based on the applicant's or expert's discretion.
- LOR ranges from 1 to 5, with 1 being the worst and 5 being the best. This should be entered based on the applicant's or expert's discretion.
- Undergraduate CGPA ranges from 1 to 10.
- The research section is used to indicate whether the applicant has a research paper published or not.

### Reflection

In terms of what worked for the project, we found that using a well-prepared and relevant dataset was crucial for achieving accurate results. Throughout the development of our model, we followed an iterative process that involved conducting rigorous experiments to evaluate the performance of each model accurately. By evaluating and comparing different models' performance, we were able to identify which model was most suitable for our specific problem. We also benefited from strong team dynamics and used effective collaboration and communication between our team members to ensure that we were making progress and addressing any challenges that arose. This included regular team meetings, clear division of tasks, and open communication channels. We also demonstrated strong problem-solving skills, and our team had a high level of

technical proficiency and critical thinking to identify and address challenges that arose throughout the project.

On the other hand, one significant challenge that we faced was the availability of insufficient and irrelevant data. To overcome this challenge, we cleaned the data and transformed it into a format that was suitable for our machine learning algorithms. If we had more time and resources, we could collect more data to improve the accuracy of the model. With more resources, we could've experimented with more complex models and algorithms that may provide better accuracy and insights. We could also have spent more time optimizing our algorithms and hyperparameters to achieve better performance.

If given unlimited time and resources, we will enhance the user interface by incorporating additional elements that will make it more intuitive and user-friendly. For example, we will add more visual cues such as icons, tooltips, or hover effects to help guide the user's interactions with the interface. By doing so, we can increase the overall usability and effectiveness of the interface, ultimately resulting in a better user experience. We will also set a minimum and maximum value that the user is allowed to enter for a certain input field or variable. This will ensure that the input entered by the user is within a certain acceptable range, which can help prevent errors or invalid input. We would also invest in real-time monitoring features to improve the system's reliability and accuracy. Additionally, we would experiment with more complex models and algorithms to achieve better performance.

If we decide to move forward with this application, we will refine the model to improve its accuracy and reliability. This will involve collecting more data, experimenting with different models and algorithms, and optimizing the model's hyperparameters. We would also like to collect user feedback by conducting user tests, surveys, and interviews to identify areas for improvement and ensure that the application meets the users' needs and expectations. We will monitor the system's performance, update the model with new data, and address any issues or bugs that arise.

We would also like to experiment with NLP models, which can be used to analyse the language used to rate the letters of recommendation (LOR) and statement of purpose (SOP) submitted by the candidates. These models can identify the sentiment, tone, and context of the language used in the letters and statements, allowing for a more accurate assessment of the candidate's qualities and strengths. By using an NLP model to analyse the LOR and SOP, the subjective element of the evaluation process can be minimized. Instead of relying on the personal biases and opinions of the evaluators, the model can provide a more objective rating of the candidate's qualifications based on the analysis of their written communication.

## Broader Impacts

**Positive impacts:** This application can help students understand their chances of getting into a particular university based on their academic background, test scores, and other relevant factors. This can help them make informed decisions about which universities to apply to and which ones may be more difficult to get into.

The application can help students identify areas where they may need to improve to increase their chances of acceptance.

**Negative impacts:** Like all predictive models, our model relies on statistical models and algorithms to make predictions, which may not capture the unique qualities or experiences of individual applicants. This could reduce the emphasis on individualized evaluation and undermine the holistic approach that many universities value.

Our application tends to focus on specific factors such as test scores and GPA, which can lead students to focus solely on these factors at the expense of other important aspects of their application, such as extracurricular activities and personal statements.

**To mitigate potential harms:** We will have to regularly review and update the algorithm to ensure that it remains accurate and reflects changes in the application process or the applicant pool. This can help to prevent unintended consequences that may arise from an outdated or inaccurate algorithm.

This application can provide guidance to users on how to interpret their admission predictions and how to use the information to make informed decisions. This can include emphasizing the importance of considering other factors in the application process and reminding users that admission predictions are not guaranteed.

## References

Dataset: <https://www.kaggle.com/competitions/acm-summer19-inclass-1/data>

M. S. Acharya, A. Armaan, and A. S. Antony, "A comparison of regression models for prediction of graduate admissions," ICCIDS 2019 - 2nd Int. Conf. Comput. Intell. Data Sci. Proc., pp. 1–5, 2019.