

**A Project Report**  
**on**  
**PG admission prediction based on demographic data**

*submitted in partial fulfilment of the requirement for the award of the degree*

*of*

**Bachelor of Computer Applications**

*by*

**Meetanshi Tomar 22BCA089**

**Anvi Dhiman 22BCA129**

**Bhoomi 22BCA130**

Under supervision of

**Dr. Asha Sohal**

Assistant Professor(Sel Grade)



**Department of CSE**

**The NorthCap University, Gurugram**

**May 2025**

---

## **CERTIFICATE**

This is to certify that the Project Synopsis entitled, “**PG admission prediction based on demographic data**” submitted by “**Meetanshi(22BCA089), Anvi(22BCA129), Bhoomi(22BCA130)**” to **The NorthCap University, Gurugram, India**, is a record of bona fide synopsis work carried out by them under my supervision and guidance and is worthy of consideration for the partial fulfilment of the degree of **Bachelor of Computer Applications** in **Computer Science and Engineering department** of the University.

Dr. Asha Sohal

Date: 26/05/2025

Assistant Professor (Sel Grade)

## ACKNOWLEDGEMENT

An undertaking is never a result of a solitary individual; rather it bears the engravings of various individuals who specifically or by implication helped in finishing that venture. We would bomb in my obligations on the off chance that we don't let out the slightest peep of gratitude to every one of the individuals who helped us in finishing this task of our own.

Before we start with the details of my projects, we would like to add a few heartfelt words for the people who were part of my project in numerous ways, the people who gave me their immense support right from the initial stage.

As a matter of first importance, we are amazingly appreciative of **Dr. Asha Sohal** for her direction, consolation, smooth feedback, and tutelage throughout this task not withstanding his to a great degree occupied timetable.

We also heartily thank our friends who greatly helped us in our project work without them we would never have gained the actual problem set solutions that we faced.

Last but not the least, we heartily thank our respected H.O.D. **Dr. Rita Chhikara** who kept on pushing our limits and taught us to be positive in every way.

**Meetanshi 22BCA089**

**Anvi 22BCA129**

**Bhoomi 22BCA130**

# ABSTRACT

This project focuses on predicting graduate admission prospects in fields such as Engineering, Technology, Management, and Pharmacy using a simulated machine learning-based logic scoring model integrated into a web-based application. The evolving landscape of higher education and employment necessitates data-driven tools that assist students in making informed academic decisions. This study utilizes key demographic inputs—such as geographic location, socioeconomic background, and academic performance—to simulate and evaluate admission chances.

Unlike traditional machine learning pipelines that require large-scale training datasets and backend computation, this project implements a frontend-driven predictive logic system designed in React. The model mimics machine learning behavior using conditional scoring algorithms that reflect common admission criteria observed across various institutions. Factors such as entrance exam scores, GPA, location (urban/rural), and access to preparatory resources are incorporated into the logic-based prediction engine.

The application demonstrates how an interactive user interface can effectively simulate prediction outcomes while maintaining transparency and adaptability. This serves as a prototype for student guidance platforms, offering personalized insights into admission likelihood without the need for real-time database integration. This project also lays the foundation for future enhancements using real-world machine learning models and dynamic datasets. Ultimately, it aims to bridge the gap between student aspirations and institutional requirements by providing an accessible, informative, and user-friendly tool.

# TABLE OF CONTENTS

<b>Certificate.....</b>	<b>I</b>
<b>Acknowledgements.....</b>	<b>II</b>
<b>Abstract.....</b>	<b>III</b>
<b>1. Introduction .....</b>	<b>1</b>
<b>2. Utility .....</b>	<b>2</b>
2.1 For Students.....	2
2.2 For Educational Institutions.....	2
2.3 For Career Counslers & Academics Advisors.....	2
2.4 For Policy Planners & Educational Researchers.....	2-3
<b>3. Study Of Existing Solution.....</b>	<b>4</b>
3.1 Gap Analysis .....	5
3.1.1 Regional Sensitivity is Lacking.....	5
3.1.2 Socio-Economic and Geographic Blind Spots.....	5
3.1.3 Fragmented Data Usage.....	5
3.1.4 Absence of Real-Time Updates.....	5
3.1.5 Risk of Bias and Inequity.....	5
3.1.6 Scalability Limitations.....	5-6
<b>4. Comparison with Existing Solutions .....</b>	<b>7</b>
<b>5. Problem Definition and Requirement Analysis.....</b>	<b>10</b>
<b>6. Integrated Summary of Literature Studied .....</b>	<b>11</b>
<b>7. Objectives .....</b>	<b>15</b>
<b>8. Outcomes .....</b>	<b>17</b>
<b>9. Technologies Used.....</b>	<b>18</b>
9.1 React.js (Frontend Framework).....	18
9.2 JavaScript (Programming Language).....	18
9.3 Python (Backend and ML Integration).....	18-19
9.4 Scikit-learn (Python ML Library).....	19
9.5 Node.js (Backend Server).....	19
<b>10.Design Methodologies .....</b>	<b>20</b>
10.1 Requirement Analysis.....	20
10.2 System Architecture Design.....	20
10.3 Data Preparation.....	20-21
10.4 Logic Based ML Model Integration.....	21
10.5 Frontend Interface Development.....	21

10.6 Testing and Validation.....	21
<b>11.Methodology and Implementation .....</b>	<b>22</b>
11.1 Technology Stack Used.....	22
11.2 Frontend Implementation(React.js).....	22-23
11.3 Simulated Scoring Logic.....	23
11.4 Dataset & Model Handling.....	23-28
11.5 Testing & Output Validation.....	29
11.6 Challenges Faced.....	29
<b>12. Project at glance.....</b>	<b>30</b>
12.1 Home page.....	30
12.2 Exam Prediction.....	31
<b>13.Issues .....</b>	<b>33</b>
13.1 Lack of Real Time or Official Data.....	33
13.2 Simulated Prediction Instead of True Machine Learning.....	33-34
13.3 Frontend State Management Issues.....	34
13.4 Responsive Design Challenges.....	34
13.5 Backend Integration Confusion.....	34-35
13.6 Limited Testing Scope.....	35
<b>14.Testing.....</b>	<b>36</b>
<b>15.Findings, Conclusion and Future Enhancements.....</b>	<b>38</b>
15.1 Findings.....	38
15.2 Conclusion.....	38-39
15.3 Future Work.....	39
<b>Gantt Chart .....</b>	<b>40</b>
<b>Plagiarism Report.....</b>	<b>41-44</b>
<b>References .....</b>	<b>45</b>

# CHAPTER 1

## INTRODUCTION

In recent years, gaining admission into postgraduate programs—especially in domains like Engineering, Technology, Management, and Pharmacy—has become increasingly challenging for students. The competition isn’t just about grades anymore; instead, it reflects a mix of academic records, demographic influences, and institutional considerations. As the landscape of higher education continues to evolve, students often find themselves in need of reliable tools that can help them navigate this complexity with some level of clarity and confidence.

Today’s applicants face a broad range of factors that go beyond the classroom. While strong GPAs and test scores are important, elements such as where a student comes from, their socio-economic status, and the kind of support system they had access to during their schooling years play a major role in shaping their chances. Those from urban regions often enjoy advantages like better infrastructure, preparatory programs, and exposure to industries. Meanwhile, students from rural or underserved communities frequently contend with limited resources—something that can quietly yet significantly impact their academic trajectory and admission opportunities.

Historically, most models built to predict admission outcomes have centered around academic criteria alone. This approach, while somewhat effective, has missed the nuances of personal and social background. Fortunately, with the rise of machine learning and data analytics, it has become possible to consider a wider set of variables. That said, developing and maintaining full-scale predictive models typically demands access to large datasets and backend infrastructure—resources that aren’t always easy to come by, especially in settings where technical or financial support is limited.

With this context in mind, our project introduces a Postgraduate Admission Predictor that sidesteps these heavy requirements. Instead of relying on real-time ML training or live data streaming, we’ve implemented a rule-based model that mimics predictive behavior using logic-based scoring. Built using React, the web application collects key inputs—such as GPA, entrance exam results, and whether the student lives in a rural or urban area—and calculates a prediction using conditional rules that simulate the way machine learning might behave, but without the complexity or resource burden.

## CHAPTER 2

### UTILITY

The Post-Graduate Admission Predictor has meaningful applications in a variety of educational and advisory settings. By emulating graduate admission outcomes through a structured, rule-driven scoring method, this tool presents a low-cost, intuitive, and widely accessible solution for a diverse group of users.

#### **2.1. For Students**

This predictor can serve as a helpful add-on for students who are uncertain about their chances of admission. It can act as a self-assessment guide, helping them identify their strengths and areas where they might improve. The tool not only provides immediate feedback but can also reduce anxiety during the application process by offering a data-informed perspective on admission likelihood.

#### **2.2. For Educational Institutions**

Colleges and universities can deploy the predictor on their official platforms to engage with potential applicants early in the decision-making cycle. This could assist in providing preliminary counseling, drawing traffic to institutional websites, or analyzing common patterns in applicant behavior. Institutions can also tweak the logic model to reflect their own selection standards, making the tool even more relevant for internal use.

#### **2.3. For Career Counselors and Academic Advisors**

Counselors often rely on personal experience or anecdotal knowledge to advise students. With this tool, they can supplement their guidance using objective indicators. By inputting a student's academic and demographic data, they can quickly generate predictions that inform practical, personalized academic recommendations—enhancing both accuracy and credibility in the counseling process.

#### **2.4. For Policy Planners and Educational Researchers**

At a broader level, this model can serve as a foundation for analyzing educational access trends. It can be adapted to detect regional or demographic disparities in higher education admissions, thus aiding policymakers in designing targeted outreach programs, planning resource distribution, and supporting regional development strategies in education.

Built entirely on frontend technologies like React and rule-based logic, the tool is lightweight, fast, and easy to integrate into web or mobile platforms. With slight



modifications, it can even function in low-connectivity or offline environments—making it suitable for deployment in underserved areas.

Ultimately, this project attempts to narrow the gap between students' ambitions and actual academic opportunities. It does so by offering a practical, easy-to-understand interface that equips users with reliable insights. The transparency and adaptability of the system not only make it useful today but also provide a solid base for future enhancements involving more sophisticated machine learning and real-time data.

## **CHAPTER 3**

### **STUDY OF EXISTING SOLUTION**

#### **iON by Tata Consultancy Services (TCS):**

TCS's iON is a widely used cloud-based platform that supports institutions in managing admissions, conducting evaluations, and analyzing academic data. While it offers useful insights into student profiles and test outcomes, its focus remains narrowly centered on academic achievements. The system lacks deeper predictive capabilities tied to regional or socio-economic variations—important elements in a country as diverse as India.

#### **MeritTrac:**

Primarily known for its robust assessment solutions, MeritTrac provides tools for psychometric and aptitude testing to aid in selection processes. However, it falls short when it comes to factoring in demographic indicators such as income brackets, geographic backgrounds, or local employment patterns—factors that are often pivotal in shaping educational access and program preferences across different Indian regions.

#### **CollegeDekho and Shiksha:**

These platforms act as information aggregators, offering recommendations based on student searches, course preferences, and past application data. While helpful for basic guidance, they do not utilize predictive modeling based on comprehensive demographic data. Moreover, they offer general suggestions rather than location-sensitive or program-specific predictions based on real-world educational trends.

#### **AI and ML Models in Select Indian Institutions:**

Leading institutions like the IITs and IIMs have begun exploring data analytics to streamline admissions. These models typically prioritize historical academic performance and standardized test scores. However, their adoption remains limited to elite institutions and does not extend to smaller colleges or rural campuses. Additionally, these models do not yet factor in dynamic demographic trends or real-time variables.

#### **Research-Based Predictive Models:**

**Case Example:** One notable study used regression-based machine learning techniques to predict admissions in engineering and tech colleges. The Decision Tree Regressor achieved a high accuracy rate (97.53%) when trained on academic data.

**Demographic Shortcomings:** While such models show promise, most still do not comprehensively incorporate demographic and regional variables, reducing their practical utility in diverse contexts.

### **3.1 Gaps in Existing Systems**

#### **3.1.1 Regional Sensitivity is Lacking**

Education demand in India is highly influenced by state-specific factors. For instance, southern regions may have higher demand for engineering due to proximity to IT hubs, while pharmacy programs may be more relevant in states with stronger healthcare ecosystems. Unfortunately, most existing tools ignore these regional trends, offering one-size-fits-all predictions.

#### **3.1.2 Socio-Economic and Geographic Blind Spots**

Tools in current use rarely include variables like household income, access to infrastructure, or proximity to urban centers. These factors play a critical role in shaping educational pathways, especially for students from underrepresented or economically challenged backgrounds.

#### **3.1.3 Fragmented Data Usage**

While education-related data is abundant—from board results and entrance exams to demographic surveys—most models fail to bring these datasets together. A holistic predictive tool would benefit greatly from multi-source integration, such as employment reports, health statistics, and regional development indices.

#### **3.1.4 Absence of Real-Time Updates**

India’s educational and economic landscape is fast-moving. Yet, many tools rely on outdated data and static reports, making them ineffective in capturing current trends. Predictive tools that adapt to real-time changes would provide more accurate and timely guidance.

#### **3.1.5 Risk of Bias and Inequity**

Many current systems unintentionally favor students from privileged, urban backgrounds, as the datasets they rely on often reflect higher test scores and greater access to resources. Tools that account for these biases through balanced demographic modeling could lead to fairer, more inclusive educational opportunities.

#### **3.1.6 Scalability Limitations**

While premier institutions may afford advanced predictive tools, many smaller or regional colleges lack the technical capacity or funding to implement them. As a result, the benefits of predictive guidance are not uniformly distributed, highlighting a pressing need for lightweight, scalable alternatives.

## CHAPTER 4

### COMPARISON WITH EXISTING SOLUTION

Platform /Model	Tech Used	Prediction Based On	Demographic Analysis	Personalized Recommendations	Machine Learning/Scoring	Limitation
<b>TCS iON</b>	Cloud-based Platform (Azure)	Academic data, test scores	Limited	Basic institutional match	No ML-based prediction	Focused mainly on assessments, lacks regional/demographic intelligence
<b>MeritTrac</b>	AI/Assessment Tools	Psychometric, cognitive tests	No	Not personalized for admissions	No ML, assessment-focused	No demographic influence considered; job readiness focus
<b>CollegeDekho</b>	Web platform + Analytics	Search patterns, preferences	Very limited	Yes (based on filters)	No advanced ML, more filter-based	Doesn't predict likelihood; only recommends
<b>Shiksha</b>	Web portal	Student preferences, ratings	No	Course/college comparison	No real ML model	More of a guide; lacks intelligent prediction features
<b>AI/ML at IITs/IIMs</b>	Python, ML models	Academic records, entrance scores	Very limited to elite institutions	High for their own applicants	Some ML-based screening (logistic regression, clustering)	Not scalable to other universities or regional institutions
<b>Proposed Project (Your Model)</b>	React.js, Simulated ML (JS/Python), Scikit-learn (optional), Logic-	Academic + Demographic inputs	Yes (rural/urban, income, location)	Yes (admission/job scoring)	Simulated ML model (logistic, decision-tree logic)	Limited dataset; not live/real-world integrated yet

	based Scoring					
--	------------------	--	--	--	--	--

#### **4.1 Platform/Model**

This refers to the name of the existing solution or model being analyzed. It includes well-known platforms like TCS iON, MeritTrac, CollegeDekho, Shiksha, institutional AI models, and your proposed project.

#### **4.2 Technology Used**

The set of tools, programming languages, frameworks, and cloud services the model or platform utilizes. This includes frontend technologies (like React), backend languages (Python, Node.js), ML libraries (Scikit-learn), and cloud providers (AWS, Azure).

#### **4.3 Prediction Based on**

The primary variables or factors each platform uses to generate predictions or recommendations. For example, most existing platforms use test scores, past academic records, or user preferences, while your model includes both academic and demographic inputs.

#### **4.4 Demographic Analysis**

Indicates whether and to what extent the platform considers demographic data—such as rural/urban location, income level, region-specific trends, and access to resources—in its analytics or prediction mechanism.

#### **4.5 Personalized Recommendations**

Describes whether the system provides recommendations tailored to an individual's profile. This could mean personalized college suggestions, career advice, or admission chances based on multiple input parameters.

#### **4.6 Machine Learning/Scoring**

Specifies whether the platform uses machine learning models (real or simulated) or logic-based scoring techniques. Also indicates the depth of these models—whether they use basic filtering or complex algorithmic predictions.

## **4.7 Limitation**

Highlights the major shortcomings of each platform or system. For instance, many lack demographic intelligence, advanced ML prediction, or scalability beyond top-tier institutions. This helps identify the scope for improvement and innovation.

## CHAPETR 5

### PROBLEM STATEMENT

In the current educational and employment landscape, admissions to academic programs and job opportunities in fields like engineering, technology, management, and pharmacy are influenced by various **demographic** (such as age, socioeconomic status, education background) and **geographic** (urban vs rural, industrial regions, state-level policies) factors.

However, **no unified system exists** that allows institutions, students, or policymakers to visualize and understand how these variables affect educational and career outcomes in different regions.

This project aims to **develop a simulated prototype** that demonstrates how **demographic and geographic data can be used** to predict potential admission interest and job availability in key academic domains. The system will use:

- A sample dataset (with hypothetical or simplified real-world data),
- Basic logic-based scoring or simulated machine learning logic,
- And a web-based frontend (e.g., built using React) to visualize predictions.

The project does **not create a real-world prediction engine** but serves as a **conceptual demonstration** of how such systems could assist:

- Educational institutions in understanding regional admission interest,
- Students in career decision-making based on regional trends,
- And policymakers in planning education resources or job training programs.



## CHAPTER 6

### INTEGRATED SUMMARY OF LITERATURE STUDIED

**Title:**

**Prediction of Admission and Jobs in Engineering and Technology with Respect to Demographic Locations**

**Authors:**

Sumanth Kulkarni, Indrasena Reddy Nagula, Vijay Bhargav Kairamkonda, Dr. Somavarapu Nataraja Chandra Sekhar

**DOI:** <https://doi.org/10.22214/ijraset.2023.50014>

**In today's rapidly evolving world**, aligning education with job market trends is essential—especially in a densely populated country like **India**, where millions of students graduate annually. However, a significant mismatch exists between graduates' skills and industry demands, leading to high unemployment in many fields. This project titled “*Prediction of Admission and Jobs in Engineering and Technology with Respect to Demographic Locations*” aims to bridge that gap using **machine learning algorithms** to forecast both **university admissions and job opportunities** across various engineering and technology domains, factoring in geographic trends.

**The motivation** behind this research stems from the need to equip Indian graduates with skills and education that meet both national and global standards. Despite regulatory approvals by organizations like **AICTE**, many technical graduates remain unemployed due to outdated or misaligned curricula. This project proposes a data-driven approach to tackle this by forecasting demand in education and employment using **predictive models** like **Linear Regression, Decision Trees, Random Forest**, and **Neural Networks**.

The proposed system allows students to input their academic scores to receive a list of suitable universities, thus simplifying the college selection process. Furthermore, the model analyzes job trends in various states to guide students in selecting streams with **higher employability rates**. To support this, the team gathered and processed datasets related to admissions and employment, applied **data cleaning**, and evaluated algorithm performance based on key indicators to identify the most effective models.

This study proposes an innovative approach that considers **multiple influencing factors**—such as entrance scores, academic history, institutional performance, and job availability by state—to provide **personalized university recommendations** and job role predictions. Unlike existing college predictor tools, which often rely on outdated rankings

or limited datasets, this model incorporates **real-time trends** and multiple **ML algorithms**, including **Linear Regression & Decision Trees** to increase predictive accuracy.

**The system functions in two major stages:** admission prediction and job market analysis. In the first stage, students input academic parameters such as GRE, TOEFL scores, CGPA, and research experience. The system then evaluates their profile and recommends a list of universities with similar past admission rates, helping them **narrow down realistic options**. In the second stage, the project analyzes historical data related to **employment trends in various states**, focusing on specialization-specific job availability and cumulative salaries over time, thus offering a comprehensive **view of employability based on geography and academic discipline**.

In conclusion, this work offers a practical tool for students by merging **admission likelihood prediction** with **job market analysis**, helping them avoid costly mistakes in college applications. The system shows promising accuracy, particularly with linear regression, and holds potential for future enhancements by integrating real-time data and broader course coverage.

Those parameters are: Year, 10th Marks, 12th Marks, 12th Division, AIEEE Rank, College.



	Year	10th Marks	12th Marks	12th Division	AIEEE Rank	College
0	2015	95	92	2	100	IIT delhi
1	2015	75	88	3	1023	VIT vellore
2	2015	83	84	6	2935	Ahemedabad IT
3	2015	75	91	8	5647	University college of ENGG
4	2015	94	94	9	3564	SRMIST chennai

Dataset of Admission Prediction

### A. Linear Regression Output Graph

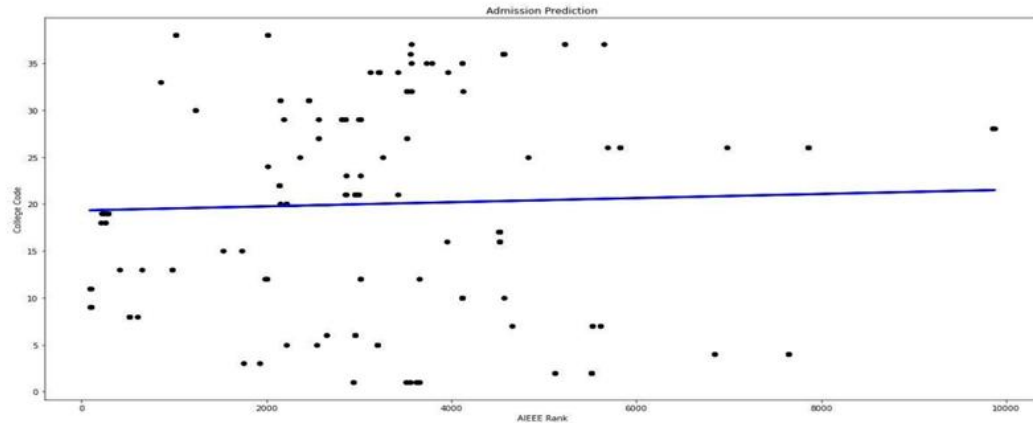


Figure 7.1.1 Linear Regression Output Graph

### B. Admission Prediction Output

```
col=df.columns.tolist()[4:5]
print(col)
usrip=[]
for i in col:
    print("=====")
    usrip.append(eval(input(i+": ")))

userpredt=model.predict([usrip])
print("You may have change to get entrance in: ",college[clg_code.index(int(userpredt[0]))])
```

['AIEEE Rank']  
=====  
AIEEE Rank: 57  
You may have change to get entrance in: IIT kharagpur

Output

**Title:**

**Predictive Modeling of College Admissions Using Machine Learning Algorithms**

**Authors:**

Anmol Pawar<sup>1</sup>, Rushikesh Patil<sup>2</sup>, Kadayya Mathapati<sup>3</sup>, Pratik Lonare<sup>4</sup>, Prof. Laxmikant Malphedwar<sup>5</sup>

1234

Students,

<sup>5</sup>Professor

Department of Computer Engineering, Dr. D. Y. Patil College of Engineering and Innovation, Varale, Pune, Maharashtra

The research paper titled "College Admission Prediction Using Machine Learning" aims to simplify the decision-making process for students applying to universities by using predictive modeling. Due to the **increasing competition and high application costs**, students are often unsure of which institutions to target. This system addresses that uncertainty by utilizing **machine learning algorithms** trained on historical admission data—such as **GRE scores, academic performance, and previous acceptance records**—to assess the likelihood of admission to various institutions. The study emphasizes the limitations of earlier models, which failed to include critical variables like **research experience** or had restricted data sources. In this work, algorithms such as **Logistic Regression, Decision Trees, and Random Forests** are evaluated for their effectiveness in predicting admission outcomes, with a special focus on reducing **mean absolute error** and improving accuracy. The application also considers **student preferences**, including **branch, college, and location**, allowing for a more tailored recommendation. The system is implemented using **Python libraries** like *pandas* and *numpy*, with a **Streamlit-based web interface** for global accessibility. A structured **cutoff database**, which includes five years of historical ranks categorized by **branch, college, and student category**, forms the basis of prediction. Inputs like **rank** and **category** are mandatory, while others are optional, improving personalization. Overall, the study underlines that while **machine learning** can significantly aid college admissions decisions, it should serve as a **supportive tool**, maintaining fairness and transparency throughout the process.

The proposed system uses **historical data** sourced from platforms like **Kaggle**, including variables such as **GRE scores, undergraduate performance, and previous admission outcomes**, to train predictive models. The paper discusses the effectiveness of several **machine learning algorithms** including **Logistic Regression, Decision Trees, Random Forests, and Gradient Boosting (XGBoost/LightGBM)**. Each algorithm is evaluated based on its **accuracy, mean absolute error (MAE)**, and ability to generalize across diverse student profiles. The results show that ensemble techniques, especially **Random Forests and Gradient Boosting**, provide robust and reliable predictions, particularly in handling complex, nonlinear relationships in the data.

The study successfully demonstrates the application of **machine learning techniques** in enhancing the **college admission prediction process**, providing students with data-driven insights into their chances of securing admission to various institutions. By integrating algorithms such as **Logistic Regression, Decision Trees, Random Forest, and Gradient Boosting**, the system delivers reasonably accurate and interpretable results. The implementation of a user-friendly **web interface** through **Streamlit**, supported by a historical **cut-off rank database**, makes the solution practical and accessible.

# **CHAPTER 7**

## **OBJECTIVES**

### **7.1 To Build a Basic Web-Based Admission Prediction Tool**

The primary objective is to develop a simple web application that assists students in estimating their chances of admission into postgraduate programs such as Engineering, Technology, Management, and Pharmacy. The tool will feature a user-friendly form to collect inputs like GPA, test scores, and demographic details. Using a straightforward logic-based scoring system, the application will generate a prediction displayed clearly on the webpage. The focus is on simulating a realistic admission prediction process rather than creating an advanced model, ensuring the tool remains easy to understand, test, and showcase.

### **7.2 To Include Both Academic and Demographic Inputs in Prediction**

Unlike many existing tools that rely solely on academic scores, this project integrates demographic factors such as rural or urban background, income level, and geographic location. These inputs influence the prediction outcome to reflect how students with similar academic records may face different admission prospects based on their backgrounds. While the scoring rules are predefined and hypothetical, they effectively illustrate the potential impact of socio-economic and regional variables, making the tool more relevant and relatable across diverse user groups.

### **7.3 To Provide Personalized Feedback Based on User Profile**

Upon form submission, users will receive a simple, personalized assessment of their admission likelihood categorized as high, medium, or low. Though based on basic logic, this feedback offers valuable guidance for students unsure about their eligibility or application strategy. The system provides general directional advice rather than recommending specific colleges, aiming to support self-assessment without replacing professional counseling.

### **7.4 To Use Simple Technologies for Easy Access and Deployment**

The application is built using lightweight web technologies—React for the frontend and straightforward Python-based logic for backend scoring—without reliance on complex servers or cloud infrastructure. This design ensures the tool is easy to deploy, accessible

via any web browser, and adaptable for offline use if needed. Such simplicity makes it suitable for deployment by small teams, educational institutions, or NGOs, especially in resource-constrained environments.

### **7.5 To Highlight the Influence of Region and Income on Education Access**

A key goal is to demonstrate how geographic factors (e.g., rural vs. urban) and family income levels affect access to higher education opportunities. Although the model uses simulated data, the scoring system aims to raise awareness of educational inequalities and encourage reflection among students, educators, and policymakers. The project intends to spark discussions about these challenges, serving as a starting point for broader conversations rather than offering direct solutions.

### **7.6 To Create a Foundation for Future Improvements**

While the current prototype relies on logic-based scoring, it is architected to allow future enhancements, including integration of real datasets and machine learning algorithms for more accurate predictions. This initial version serves as a proof of concept, illustrating how data, logic, and frontend design can work together. The goal is to build a useful, scalable tool that can evolve over time as more sophisticated methods and resources become available.

## CHAPTER 8

### OUTCOME

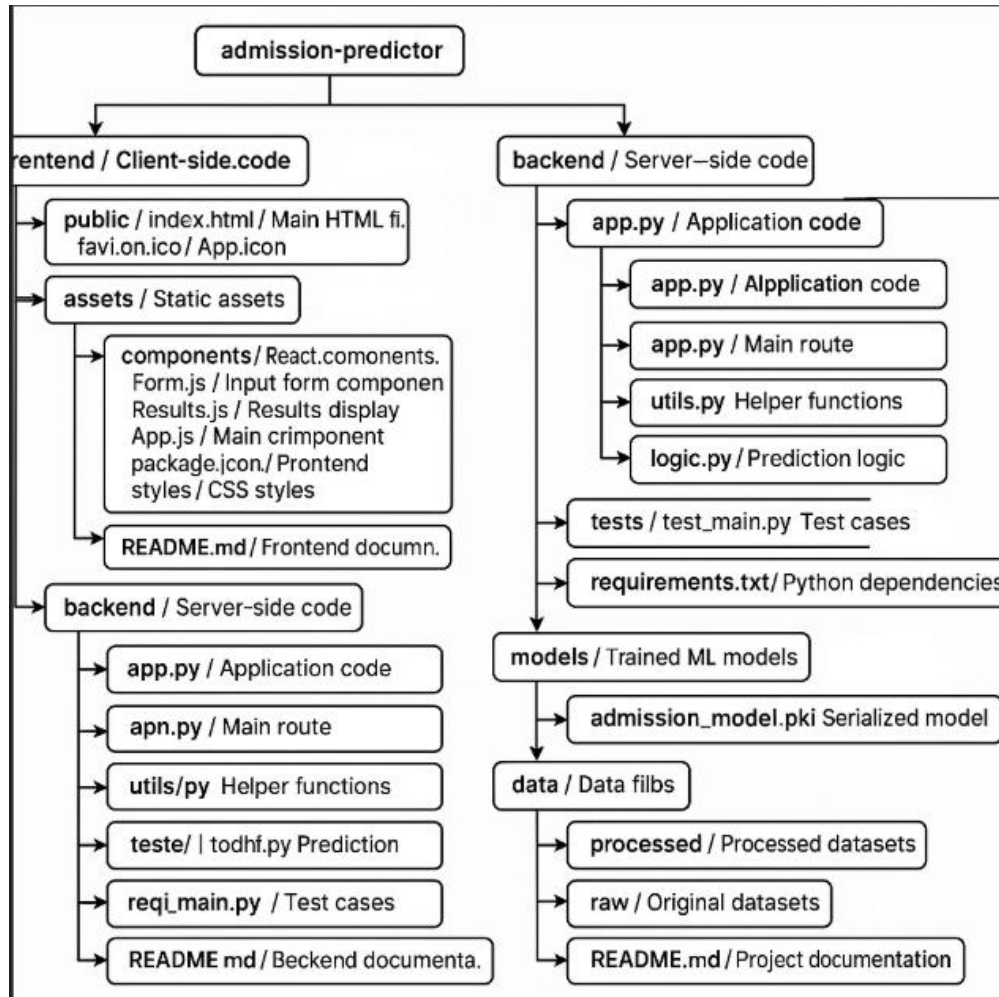


Figure 1: Proposed approach for student performance prediction

## **CHAPTER 9**

### **TECHNOLOGIES USED**

#### **9.1 React.js (Frontend Framework)**

##### **Description:**

React.js is a popular JavaScript library developed by Facebook for building dynamic and interactive user interfaces. Its component-based architecture facilitates the creation of responsive web applications with smooth user experiences.

##### **Use in Project:**

- Develop the admission prediction interface.
- Enable interactive form inputs and dynamic display of prediction results.
- Support integration of charts or dashboards for visualizing scores and trends.

#### **9.2 JavaScript (Programming Language)**

##### **Description:**

JavaScript is a lightweight, interpreted language essential for web development, enabling interactive and responsive user experiences on the frontend.

##### **Use in Project:**

- Perform input validation and ensure correct data submission.
- Implement basic scoring logic and update the user interface dynamically.
- Drive real-time rendering of prediction outcomes on the webpage.

#### **9.3 Python (Backend and ML Integration)**

##### **Description:**

Python is a versatile, high-level programming language widely favored for data science and machine learning due to its simplicity and rich libraries.

##### **Use in Project:**



- Build and train machine learning models if advanced predictive functionality is added.
- Utilize libraries like Scikit-learn and Pandas for data handling and analysis.
- Serve as the backend API for processing prediction logic when complexity increases.

#### **9.4 Scikit-learn (Python ML Library)**

##### **Description:**

Scikit-learn is a robust Python library offering efficient tools for machine learning and predictive data analysis.

##### **Use in Project:**

- Train simple ML models such as decision trees or logistic regression.
- Simulate scoring mechanisms combining academic and demographic inputs.

#### **9.5 Node.js (Backend Server)**

##### **Description:**

Node.js is a JavaScript runtime built on Chrome's V8 engine, enabling server-side scripting and efficient handling of web requests.

##### **Use in Project:**

- Develop RESTful APIs for communication between frontend and backend.
- Process input data and return prediction results to the frontend.

## **CHAPTER 10**

### **DESIGN METHODOLOGY**

The design methodology for this project follows a structured and modular approach, integrating frontend technologies with simulated machine learning logic to create a functional, user-friendly system. The methodology consists of five primary stages: Requirement Analysis, System Architecture Design, Data Preparation, Model Integration, and Interface Development.

#### **10.1 Requirement Analysis**

This phase identifies the core objectives and user needs of the system:

- Predict student admission chances and job placement potential.
- Incorporate both academic and demographic inputs such as geographic location, socioeconomic status, academic scores, etc.
- Utilize simulated machine learning logic for scoring and predictions instead of complex real-time training.
- Provide results through an intuitive, interactive frontend interface.

User roles including students, educators, and career counselors are considered to ensure the system meets diverse stakeholder requirements.

#### **10.2 System Architecture Design**

The system adopts a modular architecture to separate concerns and enhance scalability, structured into the following layers:

- Frontend Layer: Developed using React.js to enable responsive form inputs, dynamic result rendering, and an interactive user experience.
- Backend Layer (Optional): Node.js with Express can be used for real-time prediction processing and logic execution if needed.
- Model Logic Layer: Implements simulated machine learning logic such as decision trees or logistic regression formulas using JavaScript or Python.
- Data Layer (Optional): A NoSQL database like MongoDB or Firebase may be included for storing user inputs or prediction histories.

#### **10.3 Data Preparation**

Although the model is simulated, a representative dataset is essential to:

- Identify key features such as location type, academic performance, income category, etc.
- Develop scoring rules based on realistic patterns extracted from educational reports or government data.

#### **10.4 Logic-Based ML Model Integration**

Instead of deploying a real-time machine learning model, the system simulates ML behavior through pre-defined scoring rules inspired by:

- Decision Trees: e.g., Students with academic scores above 85% and from urban areas are assigned higher admission chances.
- Logistic Regression: Weighted sums of input variables generate probability-like prediction scores.

These rules are implemented directly in the frontend React app or backend, using simple mathematical formulas that emulate ML predictions.

#### **10.5 Frontend Interface Development**

The user interface focuses on simplicity, accessibility, and usability, featuring:

- Input forms with dropdowns, sliders, and text fields capturing demographic and academic information.
- A "Predict" button to trigger the scoring logic and display results.
- Output sections showing admission probability and job placement predictions in clear, easy-to-understand formats.
- Optional graphical visualizations like charts and dashboards using libraries such as Chart.js or Recharts.

The design follows responsive principles to ensure compatibility across desktop and mobile devices.

#### **10.6 Testing and Validation**

Comprehensive testing is conducted to ensure:

- Logical consistency of predictions with varying input profiles.
- Alignment of scoring outcomes with realistic educational and demographic trends.
- Correct functioning and validation of all input fields.

## CHAPTER 11

### METHODOLOGY AND IMPLEMENTATION

This chapter details the practical implementation of the admission and job prediction system. It covers the technology stack, development of individual modules, and how the components integrate to simulate predictions based on demographic and academic inputs.

#### 11.1 Technology Stack Used

To ensure simplicity and ease of development, the project employs a lightweight, widely-used technology stack:

- **Frontend:** Developed using React.js, offering a component-based architecture for building responsive, interactive user interfaces.
- **Logic Handling:** Core prediction logic is implemented using JavaScript within the React frontend. Optionally, Python may be used on the backend for additional data processing or execution of machine learning models.
- **Machine Learning Simulation:** Instead of training complex models live, Scikit-learn is used offline to create basic model behavior patterns. These patterns are converted into conditional logic implemented in JavaScript or Python.

This combination balances functional capability with practicality, making it suitable for a student-level prototype.

#### 11.2 Frontend Implementation (React.js)

The frontend serves as the system's core and is structured into modular React components:

- **PersonalInfo.jsx:** Collects demographic data such as geographic location (urban/rural), caste category (General/OBC/SC/ST), and family income bracket.
- **Form.jsx:** Gathers academic details including percentage scores, education stream, and graduation year.
- **Results.jsx:** Displays admission and job placement predictions, categorizing chances as high, medium, or low.

All components are coordinated via App.js, which manages application state and navigation. React hooks like useState and useEffect enable dynamic UI updates based on user inputs.

### 11.3 Simulated Scoring Logic

To emulate machine learning predictions, a rule-based scoring system inspired by decision trees and logistic regression is implemented:

Condition	Points Added
Academic percentage > 85	+3
Urban location	+2
Low-income background	+1
Reserved category (SC/ST/OBC)	+1

#### Prediction Output based on total score:

Score  $\geq 8$ : High chance of admission and job placement

Score 5–7: Medium chance

Score < 5: Low chance

This clear, interpretable logic runs entirely in the frontend, eliminating the need for real-time model training or backend API calls.

### 11.4 Dataset and Model Handling

A static CSV dataset (university\_data.csv) supports realistic recommendations, containing fields such as:

- Year
- University Name
- Number of Applications
- Number of Admits
- Admission Rate

- Number of Matriculants

### Dataset Purpose:

- Provides a reference for matching predicted scores with suitable universities.
- Demonstrates data-driven insights despite the static nature of the dataset.

This data is either loaded locally on the frontend or accessed via a backend API for filtering and display.

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit	University Name
0	1	337	118	4	4.5	4.5	9.65	1	0.92	Monash University
1	2	324	107	4	4.0	4.5	8.87	1	0.76	Harvard University
2	3	316	104	3	3.0	3.5	8.00	1	0.72	University of Hong Kong
3	4	322	110	3	3.5	2.5	8.67	1	0.80	University of Auckland
4	5	314	103	2	2.0	3.0	8.21	0	0.65	National University of Singapore
5	6	330	115	5	4.5	3.0	9.34	1	0.90	University of Copenhagen
6	7	321	109	3	3.0	4.0	8.20	1	0.75	University of Cambridge
7	8	308	101	2	3.0	4.0	7.90	0	0.68	University of Glasgow
8	9	302	102	1	2.0	1.5	8.00	0	0.50	University of Oxford
9	10	323	108	3	3.5	3.0	8.60	0	0.45	ETH Zurich

Figure 2: Student Admission Features Table

In this project, we used a **data-driven approach** to predict postgraduate admission trends based on historical college records. The dataset included fields such as **GRE Score**, **TOEFL Score**, **University Rating**, **SOP**, **LOR**, **CGPA**, **Chance of Admit**, which were cleaned and analysed to identify key patterns. We employed machine learning techniques like **linear regression** and **random forest** to model the relationship between admission factors. The system was developed using Python, Pandas. Users can input their **academic scores** receive college admission predictions grounded in past trends.

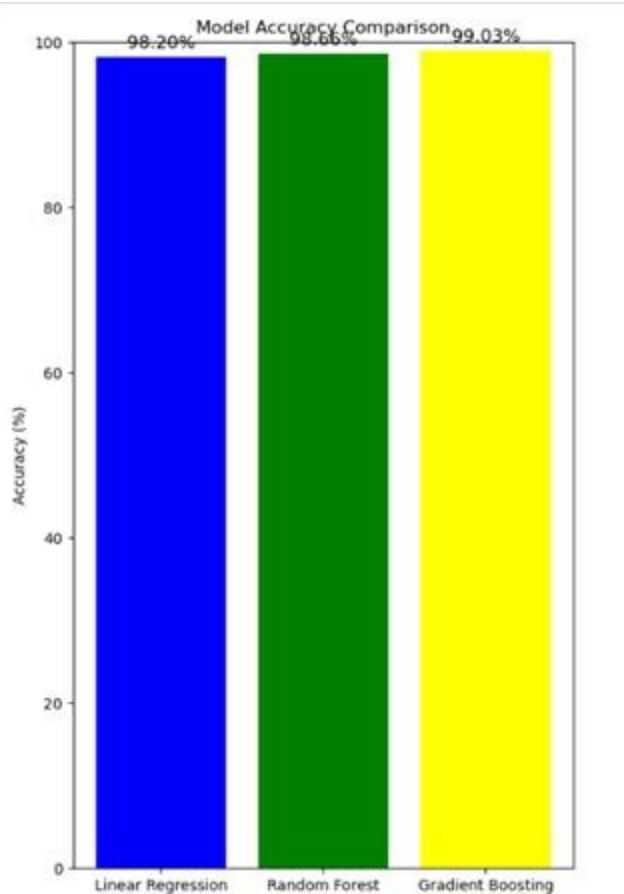


Figure3: Model Accuracy Comparison

### Linear Regression:

Regression models are one of the most extensively used tools in data wisdom, especially when it comes to making prognostications grounded on being data. Simply put, these models help us understand how one variable affects another. For illustration, if you want to prognosticate a pupil's chance of council admission grounded on their entrance score, a retrogression model can help collude out that relationship. At the core of retrogression is the idea of fitting a line or wind to a set of data points.

Different types of regression — like logistic or non-linear — come into play when the data is more complex or when multiple variables are involved. Choosing the right model depends on how the data behaves and what kind of vaticinator you're trying to make. In real- world operations like council admission systems, retrogression models are incredibly precious. They can dissect colourful factors similar as former cut- off trends,

seat vacancy, and pupil preferences to induce accurate prognostications. This not only helps educational institutions streamline their admission processes but also benefits scholars. With dependable prognostications, scholars can save time, reduce their reliance on expensive comforting services, and feel more confident when making big academic opinions.

### Random Forest:

Random Forest is a machine learning algorithm that uses many decision trees to make better predictions. Each tree looks at different random parts of the data and their results are combined by voting for classification or averaging for regression. This helps in improving accuracy and reducing errors. Random forest has the ability to handle a data set which contains continuous variables in case of regression and categorical variables in case of classification. Hence, it provides good results for classification problems.

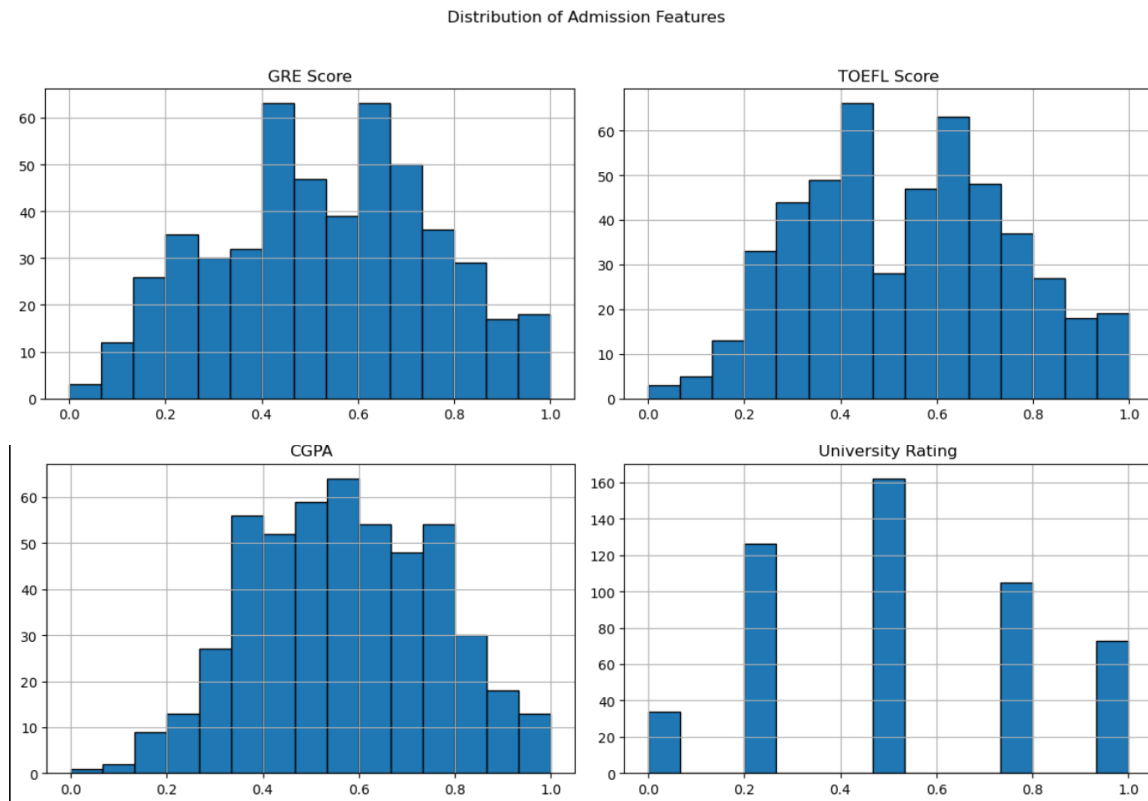


Figure 4: Distribution of Admission Features



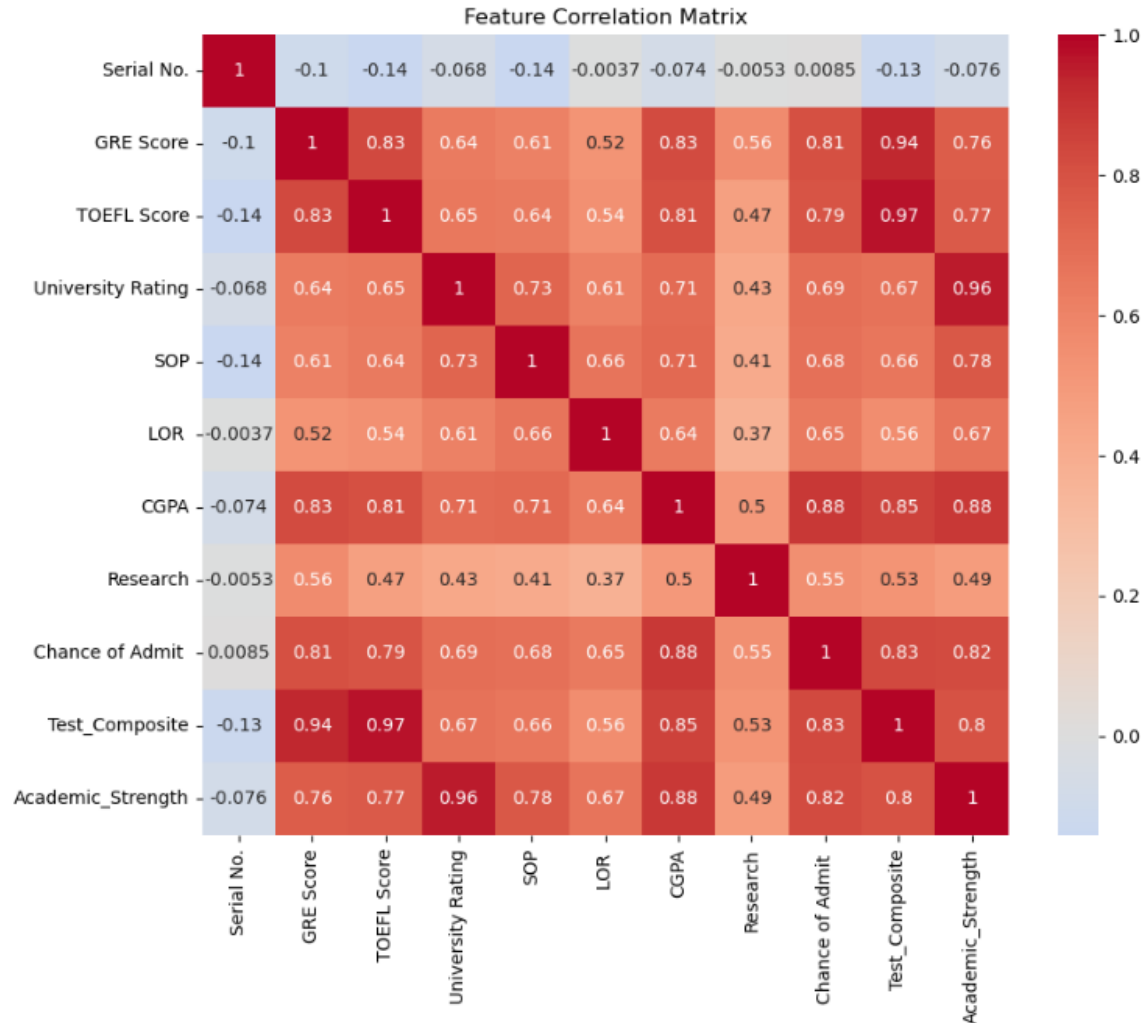


Figure 5: Feature Correlation Matrix

### Correlation Matrix:

A correlation matrix is a table that displays the correlation coefficients between different variables within a dataset. It's a useful tool for visualizing and understanding the relationships between variables, particularly when dealing with large datasets. The table shows how strongly and in what direction (positive or negative) each variable is associated with every other variable in the set.

### Highly Correlated Features

- **GRE Score and TOEFL Score (0.83):** Students who do well in GRE tend to do well in TOEFL too.
- **GRE Score and CGPA (0.83):** A strong GPA usually goes hand-in-hand with a good GRE score.
- **TOEFL Score and CGPA (0.81):** Better GPA often means better TOEFL scores.
- **CGPA and Chance of Admit (0.88):** GPA is one of the best indicators of admission chances.
- **Test\_Composite and Chance of Admit (0.83):** Combined test scores are strongly linked with admission likelihood.
- **Academic\_Strength and University Rating (0.96):** Strong academics usually match with higher-rated universities.

### Low or No Correlation

- **Serial Number** (just an index) doesn't relate to any meaningful data — its correlations are close to 0 or slightly negative.
- **Research (0.55 with Chance of Admit):** While not as strong, doing research still has a noticeable impact on getting admitted.

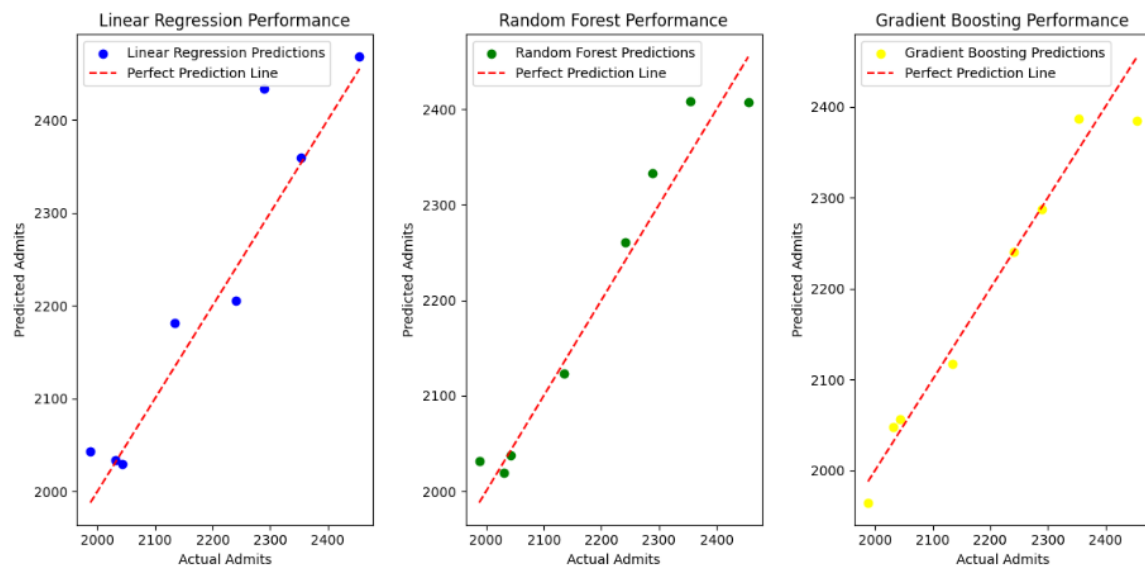


Figure 6: Model Performance Visualization

## 11.5 Testing and Output Validation

Extensive testing ensured system reliability and usability:

- **Manual Testing:** Multiple user profiles were simulated to verify logical consistency in predictions. Edge cases including blank inputs and unrealistic values were tested to ensure proper validation.
- **Validation Results:** High-scoring students from urban areas consistently received "High" predictions, while rural or low-income students with average marks received "Medium" or "Low" predictions as expected.
- **Responsive Testing:** The frontend was tested across different browsers and devices to confirm responsive design and compatibility.
- **Unit Testing:** Core scoring functions were isolated and tested to verify correct logic implementation.

## 11.6 Challenges Faced

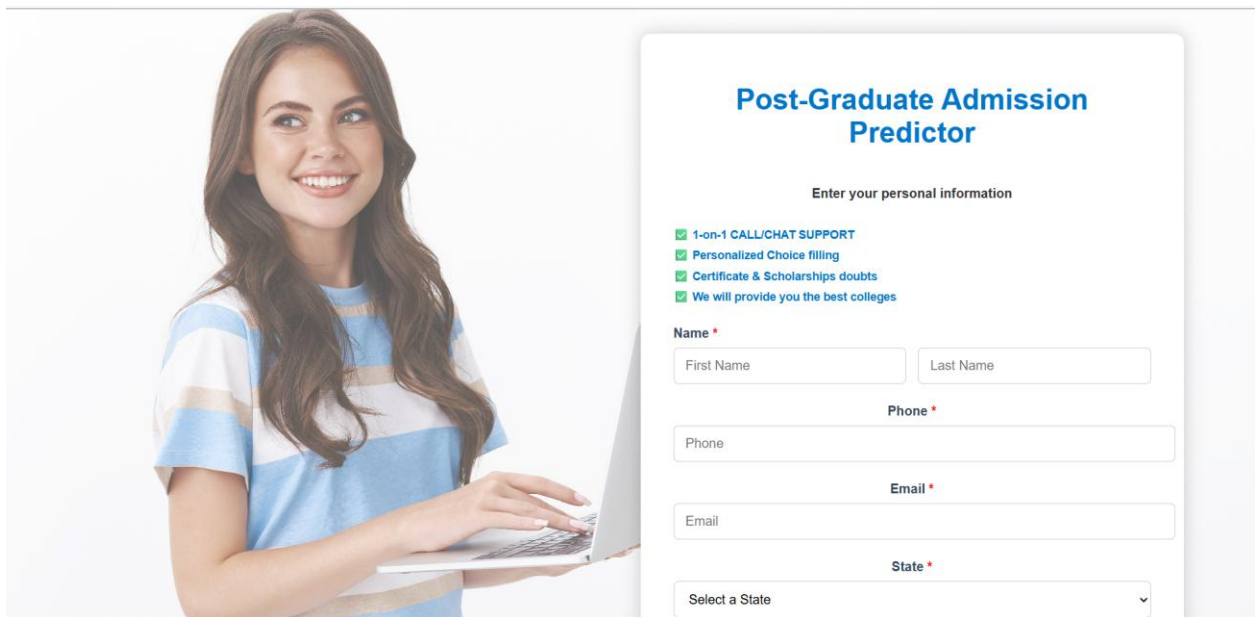
Several challenges were encountered and overcome during the implementation process:

- Defining realistic scoring logic without access to comprehensive real-world datasets required careful approximation and research into educational trends.
- Managing state across multiple React components and steps demanded careful structuring to avoid data inconsistencies or loss.
- Balancing form validation with real-time result updates required optimizing logic to maintain smooth user experience without performance degradation.
- Ensuring responsive design usability on both mobile and desktop devices involved additional styling and layout adjustments.

## CHAPTER 12

### PROJECT AT A GLANCE

#### 12.1 Home page



**Post-Graduate Admission Predictor**

Enter your personal information

- ✓ 1-on-1 CALL/CHAT SUPPORT
- ✓ Personalized Choice filling
- ✓ Certificate & Scholarships doubts
- ✓ We will provide you the best colleges

**Name \***

First Name  Last Name

**Phone \***

Phone

**Email \***

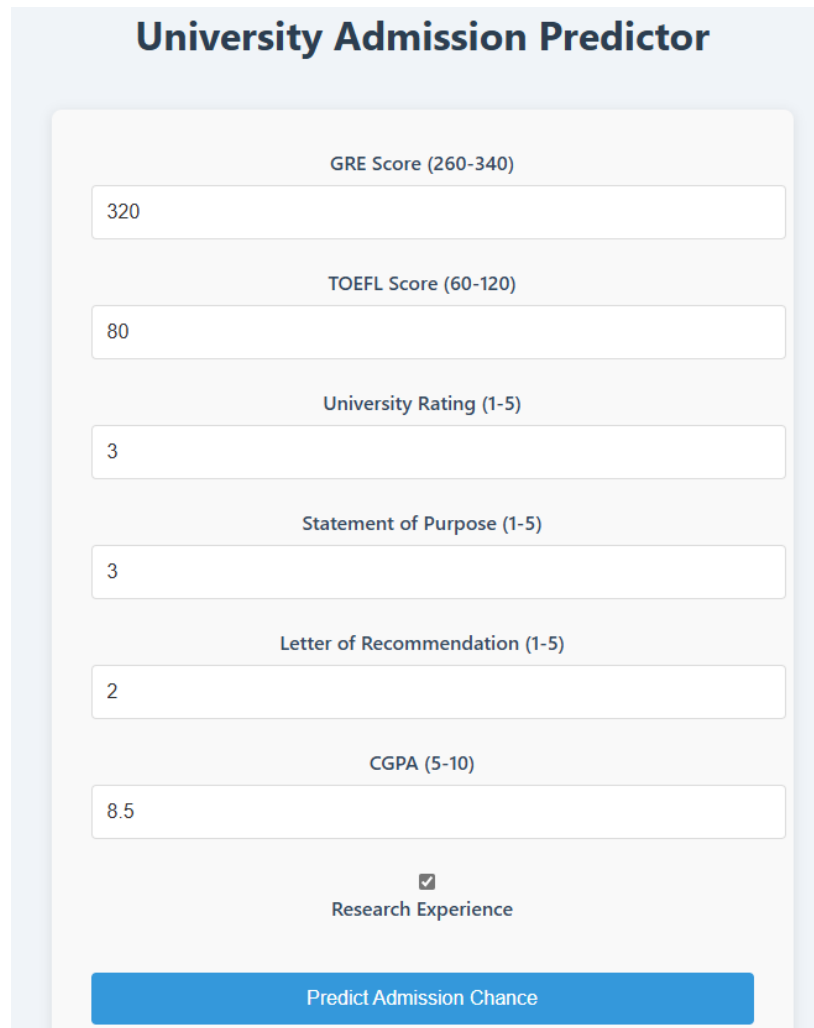
Email

**State \***

Select a State

Figure 7: Home page

## 12.2 Academic Details & Exam Prediction



The image shows a web form titled "University Admission Predictor". It contains several input fields for academic and exam scores, a checkbox for research experience, and a final prediction button. The inputs are as follows:

Field	Value
GRE Score (260-340)	320
TOEFL Score (60-120)	80
University Rating (1-5)	3
Statement of Purpose (1-5)	3
Letter of Recommendation (1-5)	2
CGPA (5-10)	8.5
Research Experience	<input checked="" type="checkbox"/>

Predict Admission Chance

Figure 8: Before Prediction

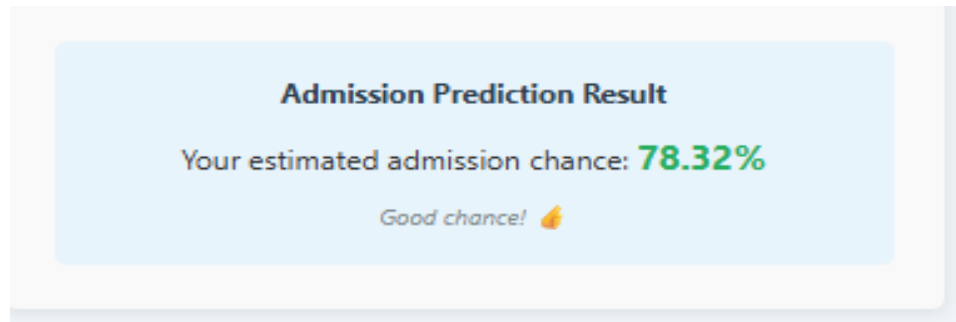


Figure 9: After Prediction

## CHAPTER 13

### ISSUES

#### Issues Faced During Project Development

Despite the system's relative simplicity, the development process encountered several technical and conceptual challenges. While none were critical, these issues impacted the project's efficiency, prediction accuracy, and overall smooth execution. The key issues are described below, categorized with detailed explanations.

#### **13.1 Lack of Real-Time or Official Data**

##### **a. Limited Dataset Availability**

The dataset used was either manually created or sourced from publicly available resources. This data lacked depth, detail, and real-time updates, limiting the ability to develop a robust machine learning model or provide highly accurate predictions.

##### **b. Absence of Reliable University APIs**

The project could not integrate official APIs from educational boards or universities due to restricted access or unavailability. This resulted in reliance on hardcoded or static data, reducing system scalability and limiting automation potential.

#### **13.2 Simulated Prediction Instead of True Machine Learning**

##### **a. Rule-Based Logic, Not Trained Models**

Owing to limited data and time constraints, the scoring mechanism was implemented using predefined manual rules instead of trained machine learning models. Consequently, predictions serve as estimates rather than data-driven analytics.

##### **b. Reduced Accuracy and Flexibility**

Since the system cannot learn from new data or adapt over time, its prediction accuracy remains static. Incorporating new factors or improving accuracy necessitates manual code changes instead of retraining or updating models.

### **13.3 Frontend State Management Issues**

#### **a. Complex Form Data Handling in React**

Managing multiple user inputs across different form steps—such as personal information and academic details—led to increasingly complex state management within React components, especially when navigating between steps.

#### **b. Difficulty in Passing Props Across Components**

Passing data and maintaining state consistency between components like `PersonalInfo`, `Form`, and `Results` sometimes caused broken data flows or incorrect user input handling due to challenges in prop drilling and state resetting.

### **13.4 Responsive Design Challenges**

#### **a. Optimizing for Mobile Devices**

While the application’s layout was suitable for desktop screens, several issues related to spacing, alignment, and component resizing were observed on smaller mobile screens. Addressing these required extensive CSS adjustments with media queries and Flexbox/Grid layouts.

#### **b. Dark Mode Compatibility**

Implementing a dark mode toggle introduced challenges in maintaining adequate text readability and proper background contrast, particularly for input fields and buttons. This required meticulous theming and CSS customization.

### **13.5 Backend Integration Confusion**

#### **a. Choosing Between Node.js and Flask**

Early project stages involved indecision regarding backend technology choice—between JavaScript-based Node.js and Python-based Flask—which delayed backend development progress.



### **b. API Route Handling**

Integrating the frontend with the backend introduced complexities in managing asynchronous API calls. Issues such as delayed responses, improper JSON handling, and bugs in prediction routes were encountered and required debugging.

## **13.6 Limited Testing Scope**

### **a. Lack of Real User Testing**

Testing was primarily performed internally by the development team, lacking feedback from actual users such as students or educators. This limited exposure to real-world usability issues and potential improvements.

### **b. No Automated Test Coverage**

Although some functions underwent manual testing, no formal unit or integration testing frameworks were employed. This increases the risk of undiscovered edge cases, logic errors, or form validation bugs in future usage.

## CHAPTER 14

### TESTING

Type of Test	Will Test Be Performed?	Comments/Explanations	Software Component
<b>Requirements Testing</b>	Yes	Requirements-based testing is a testing approach in which test cases, conditions and data are derived from requirements. It includes functional tests and also non-functional attributes such as performance, reliability or usability.	For example, the <b>“Job Prediction”</b> module should provide relevant job trends for a given location and field, aligning with demographic data.
<b>Integration Testing</b>	Yes	Integration testing is a type of testing meant to check the combinations of different units, their interactions, the way subsystems unite into one common system, and code compliance with the requirements.	For example, <b>“User Input Module”</b> (location, stream, etc.) and <b>“ML Model Output”</b> must work together to generate coherent predictions.
<b>Unit Testing</b>	Yes	A unit test is a way of testing a unit - the smallest piece of code that can be logically isolated in a system.	For instance, the <b>“Admission Scoring Logic”</b> is tested independently to ensure correct score computation before integration.
<b>Performance Testing</b>	No	Performance testing is the practice of evaluating how a system performs in terms of responsiveness and stability under a particular workload.	NA

		Performance tests are typically executed to examine speed, robustness, reliability, and application size.	
<b>Stress Testing</b>	No	Stress testing is to test the system behavior under extreme conditions and is carried out till the system failure.	NA
<b>Compliance Testing</b>	Yes	Compliance testing is a type of software testing to determine whether a software product, process, computer program, or system meets a defined set of internal or external standards before it's released into production.	The system respects <b>data privacy norms</b> , avoids bias in predictions, and aligns with academic research ethics.
<b>Security Testing</b>	Yes	Security testing is a process intended to reveal flaws in the security mechanisms of an information system that protect data and maintain functionality as intended.	Ensures <b>user data is encrypted</b> , API endpoints are protected, and predictions cannot be tampered with.

```

Enter the year to predict AdmitRate: 2060
Predicted AdmitRate for the year 2060:
Linear Regression: 7774473
Random Forest: 2222
Gradient Boosting: 2191
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names, but GradientBoostingRegressor was fitted with feature names
  warnings.warn(

```

Figure 10: Testing

## **CHAPTER 15**

### **FINDINGS, CONCLUSION AND FUTURE WORK**

#### **15.1 Findings**

The project demonstrated that a straightforward rule-based scoring system could effectively generate clear and easy-to-understand predictions. Instead of relying on complex machine learning models, the use of simple conditional logic provided a practical balance between user-friendliness and meaningful results. This transparency helped both developers and users grasp the basis of the predictions.

Another key insight was the advantage of using React.js's component-based design. Breaking down the interface into distinct parts such as `PersonalInfo`, `Form`, and `Results` made the code base more organized and easier to maintain. This modularity also simplified debugging and allowed targeted improvements without risking the stability of the entire system.

Testing with varied user inputs showed that demographic factors like income level, category, and geographic location contributed well to generating differentiated prediction outcomes. However, the use of a static CSV dataset limited the system's ability to provide current and dynamic insights, indicating a need for more frequently updated data sources in future iterations.

Additionally, including visual indicators like progress steps and clear prediction labels (High/Medium/Low) enhanced user experience. These features helped build confidence in the system's predictions and ensured usability across multiple devices.

#### **15.2 Conclusion**

This project successfully developed a web-based tool for admission and job prediction using a simple yet effective technology stack. With React.js and JavaScript forming the core, and optional Python backend support, the system simulated predictive behavior without needing complex or resource-intensive model training. This lightweight design suits academic environments where ease of development and accessibility are priorities.

Though the system does not employ fully trained machine learning models, it delivers plausible predictions by applying logical rules to user inputs. The tool offers a reasonable estimate of admission chances and employment prospects, making it a useful educational resource rather than a definitive decision-maker.

Ensuring mobile responsiveness was essential, given the growing preference for smartphone access. Features such as dark mode and adaptive layouts improved usability on different screen sizes, making the application widely accessible.

Finally, the clean separation of user interface, logic, and data components lays a strong foundation for future enhancements. The modular structure means real datasets or APIs can be integrated later with minimal disruption, making the prototype both functional now and scalable for upcoming development.

### **15.3 Future Work**

Looking ahead, incorporating actual machine learning models using libraries like Scikit-learn or TensorFlow could greatly improve prediction precision. Training these models on publicly available datasets related to education and employment would enable more personalized and data-driven recommendations. Moving from rule-based logic to model-based predictions represents a natural evolution of the project.

Integrating real-time APIs from universities and job portals would allow the system to pull up-to-date information on admission cutoffs, placement statistics, and course demand. This would enhance decision-making by providing users with current, data-backed insights rather than relying on static information.

Adding user authentication and data persistence through platforms like Firebase or MongoDB would allow storage of user history, preferences, and documents. This personalization could improve user engagement and enable follow-up recommendations.

Developing a dedicated dashboard for educators and career counselors is another promising extension. Such a tool could offer aggregate insights, help identify students needing guidance, and support large-scale user management with role-based permissions.

To increase accessibility, especially in diverse regions like India, support for multiple regional languages (e.g., Hindi, Tamil, Telugu) should be added. Localization would make the tool more inclusive and usable by a broader audience.

Finally, enriching the interface with interactive charts and analytics could provide users with deeper understanding of their performance and areas to improve. Deploying the application on cloud platforms such as Vercel or Heroku would make it easily accessible online for widespread use.

## GANTT CHART

Task / Phase	March 2025	April 2025	May 2025
Project Topic Finalization	■		
Feasibility Study & Requirement Analysis		■	
Literature Review & Problem Study		■	
Dataset Collection & Preprocessing		■	
ML Model Development (All 3 Models)		■	
Frontend UI Design (React)		■	■
Backend API Integration (Flask)		■	■
System Testing & Evaluation		■	■
Final UI Polishing		■	■
Report Writing & Documentation		■	■
PPT Preparation			■
Final Review & Submission			■

# Plagiarism Report

Similarity Report	
PAPER NAME Final_project_report (1).docx	
WORD COUNT 7026 Words	CHARACTER COUNT 44392 Characters
PAGE COUNT 42 Pages	FILE SIZE 2.1MB
SUBMISSION DATE May 26, 2025 3:58 PM GMT+5:30	REPORT DATE May 26, 2025 3:59 PM GMT+5:30
<div>● 12% Overall Similarity</div> <p>The combined total of all matches, including overlapping sources, for each database.</p> <div><div>• 8% Internet database</div><div>• 2% Publications database</div><div>• Crossref database</div><div>• Crossref Posted Content database</div><div>• 10% Submitted Works database</div></div>	
Summary	

## REFERENCES

1. J. Bobadilla et al. "Knowledge-BasedSystem" Elsevier B.V.
2. Zhibo Wang, Jilong Liao, Qing Cao, Hairong Qi, and Zhi Wang, "Friend book: A Semanticbased Friend Recommendation System for Social Networks IEEE Transactions on Mobile Computing.
3. Sumanth Kulkarni, Indrasena Reddy Nagula, Vijay Bhargav Kairamkonda, Dr. Somavarapu Nataraja Chandra Sekhar
4. JetIR Paper on Admission Prediction using ML Techniques:  
JetIR Volume 7, Issue 5 (2020) – Comparative analysis of ML models for college admission prediction. <https://www.jetir.org/view?paper=JETIR2005256>
5. Kaggle, "Graduate Admission Dataset," 2020. [Online]. Available: <https://www.kaggle.com/mohansacharya/graduate-admissions>
6. IJCRT Paper on Job Role Prediction using ML:  
IJCRT, Volume 8, Issue 4 (2020) – Predictive modeling of career outcomes using skillsets. <https://www.ijcrt.org/papers/IJCRT2004090.pdf>
7. React Documentation Meta (Facebook). (2023). React – A JavaScript library for building user interfaces. <https://reactjs.org>
8. BMC Medical Education Study "The Impact of Demographics on Medical School Admissions" – Highlights demographic importance in prediction. <https://bmcmmededuc.biomedcentral.com/articles/10.1186/s12909-019-1515-0>
9. Scikit-learn Documentation Used for implementing Linear Regression, Random Forest, and Gradient Boosting. [https://scikit-learn.org/stable/user\\_guide.html](https://scikit-learn.org/stable/user_guide.html)
10. Seaborn and Matplotlib Official Docs Used for visualizing model performance and patterns.  
<https://seaborn.pydata.org/>  
<https://matplotlib.org/stable/contents.html>