FINAL DELIVERABLES - PROJECT REPORT

UNIVERSITY ADMISSION ELIGIBILITY PREDICTOR

DOMAIN	APPLIED DATA SCIENCE
TOPIC	UNIVERSITY ADMIT ELIGIBILITY PREDICTOR
TEAM ID	PNT2022TMID35582
TEAM MEMBERS	1. PRADEEPA M - TEAM LEAD 2. SOWMYA D 3. SUJA S 4. VEDHA R
SUBMISIION DATE	19/11/2022

UNIVERSITY ADMISSION PREDICTION SYSTEM USING MACHINE LEARNING

TABLE OF CONTENTS

1. INTRODUCTION

1.1. PROJECT OVERVIEW

The project uses a machine-learning model to estimate, using information like marks and other details, whether the user is qualified for admission to the rating universities that have been chosen. The algorithm's operation ensures that the % of likelihood of admission is displayed when the user enters information such (GRE Score, TOEFL Score, University Rating, SOP, LOR, CGPA, Research). The user is given access to a user interface (UI) (Web-based application) where they can enter the above-mentioned information for prediction. The key benefit of this is that the user may use this programme to estimate eligibility and possibility of admission rather than going through the time-consuming procedure of manually determining eligibility for university admission.

1.2. PURPOSE

The goal of this project is to easily estimate an applicant's eligibility for admission to a rated university using a user interface and the given user information (GRE Score, TOEFL Score, University Rating, SOP, LOR, CGPA, Research). Additionally, this removes the chance of human mistake.

2. LITERATURE SURVEY

2.1. EXISTING PROBLEM

Previous studies in this field used the Naive Bayes algorithm to assess the likelihood that a student will be admitted to a particular university, but their main flaw was that they failed to take into account all the variables that would affect admission, such as TOEFL/IELTS, SOP, LOR, and undergraduate GPA. An evaluation network for the applications submitted by university's international students has been built using the Bayesian Networks technique. By comparing prospective students' scores to those of university students currently enrolled, this model was created to predict how well they will do. On the basis of various student scores, the model thus predicted whether the prospective student should be admitted to the university. This method won't be as accurate because comparisons are only made with students who were accepted into the universities, not with those whose admission was denied.

2.2 REFERENCES

- M. S. Acharya, A. Armaan, and A. S. Antony, “A Comparison of Regression Models for Prediction of Graduate Admissions,” Kaggle, 2018.
- C. López-Martín, Y. Villuendas-Rey, M. Azzeh, A. Bou Nassif, and S. Banitaan, “Transformed k-nearest neighborhood output distance minimization for predicting the defect density of software projects,” J. Syst. Softw., vol. 167, p. 110592, Sep. 2020.
- 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.

2.3. PROBLEM STATEMENT DEFINITION

Students are often worried about their chances of admission to University. The aim of this project is to help students in shortlisting universities with their profiles. The predicted output gives them a fair idea about their admission chances in a particular university. This analysis should also help students who are currently preparing or will be preparing to get a better idea.

3. IDEATION AND PROPOSED SOLUTION

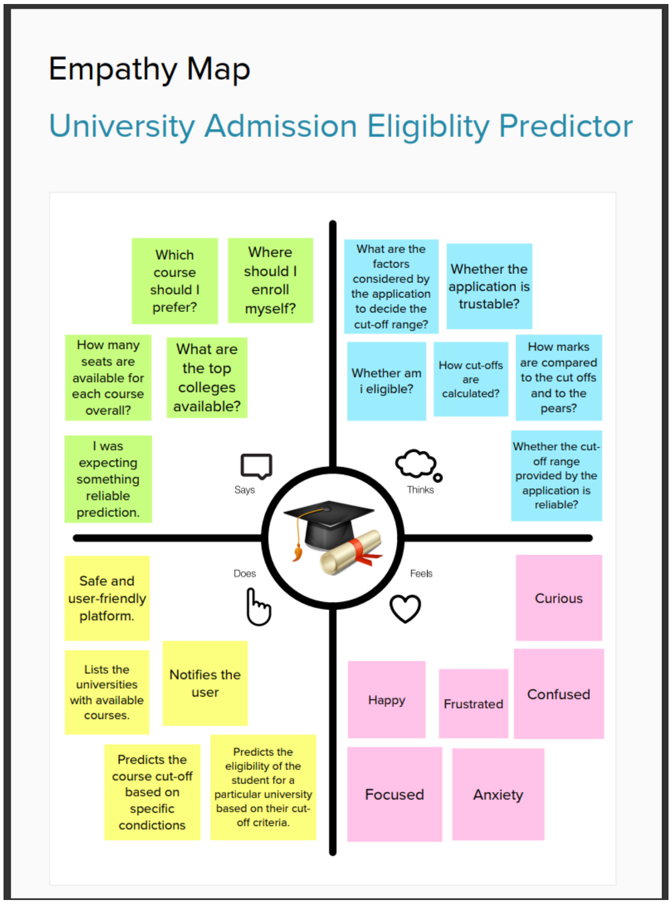
3.1. EMPATHY MAP CANVAS

An empathy map is a collaborative visualization used to articulate what we know about a particular type of user. It externalizes knowledge about users in order to

1) Create a shared understanding of user needs, and

2) Aid in decision making.

Traditional empathy maps are split into 4 quadrants (Says, Thinks, Does, and Feels), with the user or persona in the middle. Empathy maps provide a glance into who a user is as a whole and are not chronological or sequential.



Empathy Map

University Admission Eligiblity Predictor

Which course should I prefer?

Where should I enroll myself?

How many seats are available for each course overall?

What are the the top colleges available?

I was expecting something reliable prediction.

What are the factors considered by the application to decide the cut-off range?

Whether the application is trustable?

Whether am i eligible?

How cut-offs are calculated?

How marks are compared to the cut offs and to the pears?

Whether the cut-off range provided by the application is reliable?

Safe and user-friendly platform.

Lists the universities with available courses.

Notifies the user

Predicts the course cut-off based on specific condictions

Predicts the eligibility of the student for a particular university based on their cut-off criteria.

Curious

Happy

Frustrated

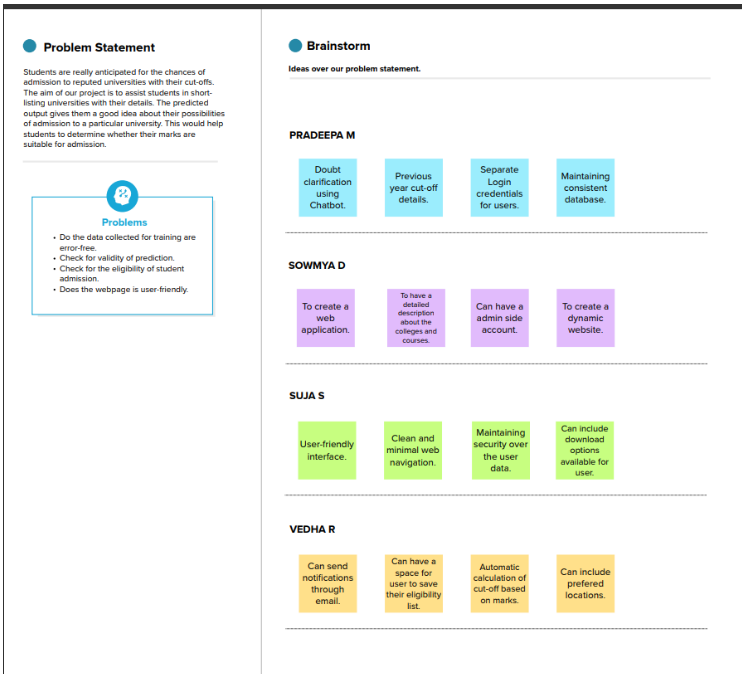
Confused

Focused

Anxiety

3.2. IDEATION & BRAINSTORMING

Brainstorming is a method design teams use to generate ideas to solve clearly defined design problems. In controlled conditions and a free-thinking environment, teams approach a problem by such means as “How Might We” questions. They produce a vast array of ideas and draw links between them to find potential solutions.



Problem Statement

Students are really anticipated for the chances of admission to reputed universities with their cut-offs. The aim of our project is to assist students in short-listing universities with their details. The predicted output gives them a good idea about their possibilities of admission to a particular university. This would help students to determine whether their marks are suitable for admission.

Problems

- Do the data collected for training are error-free.
- Check for validity of prediction.
- Check for the eligibility of student admission.
- Does the webpage is user-friendly.

Brainstorm

Ideas over our problem statement.

PRADEEPA M

- Doubt clarification using Chatbot.
- Previous year cut-off details.
- Separate Login credentials for users.
- Maintaining consistent database.

SOWMYA D

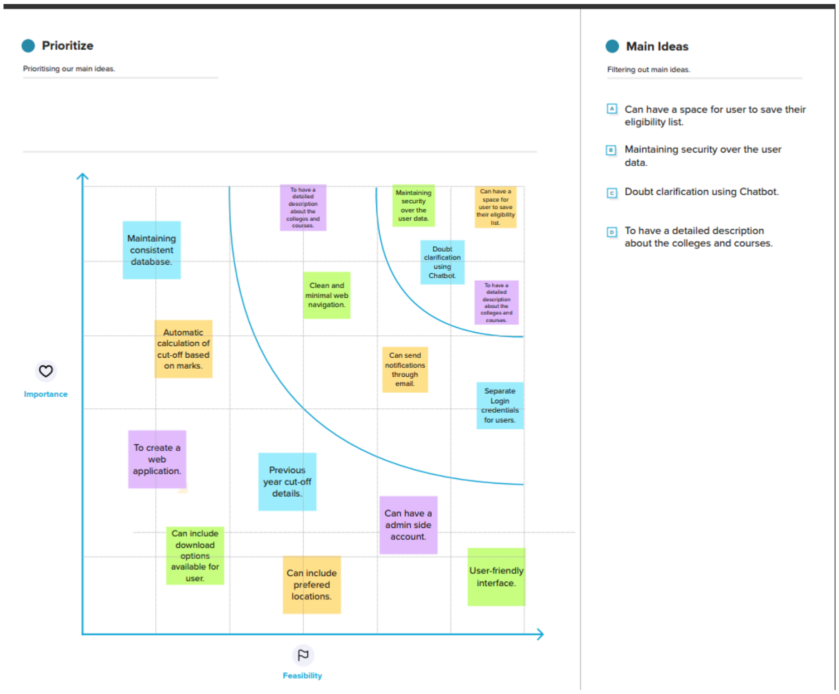
- To create a web application.
- To have a detailed description about the colleges and courses.
- Can have a admin side account.
- To create a dynamic website.

SUJA S

- User-friendly interface.
- Clean and minimal web navigation.
- Maintaining security over the user data.
- Can include download options available for user.

VEDHA R

- Can send notifications through email.
- Can have a space for user to save their eligibility list.
- Automatic calculation of cut-off based on marks.
- Can include prefered locations.



Prioritize

Prioritising our main ideas.

To have a detailed description about the colleges and courses.
Maintaining security over the user data.
Can have a space for user to save their eligibility list.
Maintaining consistent database.
Doubt clarification using Chatbot.
Clean and minimal web navigation.
To have a detailed description about the colleges and courses.
Automatic calculation of cut-off based on marks.
Can send notifications through email.
Importance
Separate Login credentials for users.
To create a web application.
Previous year cut-off details.
Can have a admin side account.
Can include download options available for user.
Can include prefered locations.
User-friendly interface.
Feasibility

Main Ideas

Filtering out main ideas.

- A Can have a space for user to save their eligibility list.
- B Maintaining security over the user data.
- C Doubt clarification using Chatbot.
- D To have a detailed description about the colleges and courses.

3.3. PROPOSED SOLUTION

The aim of the proposed system is to address the limitations of the current system. The requirements for the system have been gathered from the defects recorded in the past and also based on the feedback from users of previous metrics tools. Following are the objectives of the proposed system: • Reach to geographically scattered student. • Reducing time in activities • Paperless admission with reduced man power • Operational efficiency

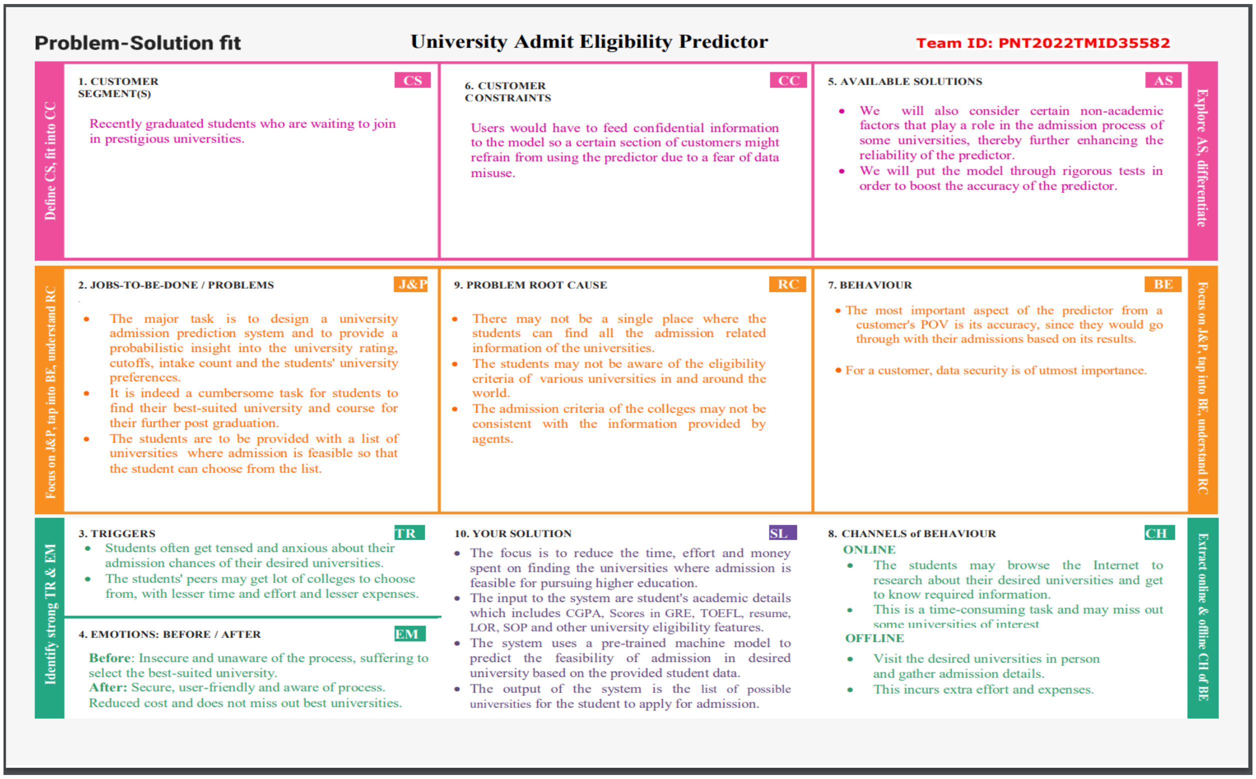
These problems can be resolved by using regression algorithms /classification algorithms as they can consider most of the features for prediction. Linear regression / KNN classification / Random Forest Regressor can be used as the machine learning model for the model. XG boost model can also be used which performs better on small to medium scale datasets but the model giving accurate and desired results only will be selected. The aim of the proposed system is to address the limitations of the current system. The requirements for the system have been gathered from the defects recorded in the past and also based on the feedback from users of previous metrics tools.

S.No.	**Parameter**	**Description**
1.	Problem Statement (Problem to be solved)	Students are really anticipated for the chances of admission to reputed universities with their cut-offs. The aim of our project is to assist students in short-listing universities with their details. The predicted output gives them a good idea about their possibilities of admission to a particular university. This would help students to determine whether their marks are suitable for admission.
2.	Idea / Solution description	It takes a lot of time and effort to conduct university and college research, which is one of the requirements for applying to universities. This issue, which is a major one for students, has not yet been resolved. There are reputable websites that rank the top colleges and universities according to factors like location, cost of attendance, degree offered, and major, but none of them utilise a machine learning algorithm to do it. As a result, we conducted this research to partially address that problem using data mining approaches.
3.	Novelty / Uniqueness	This website is going to analyse Indian universities. additionally provide numerous university-related information. the universities included in the ranking list.

4.	Social Impact / Customer Satisfaction	The webpage will lessen pupils' anxiety and ignorance. Their time, travel, and expenses will all be cut. It will provide an exact or close-to-precise prediction based on the pupils' secondary school grades.
5.	Business Model (Revenue Model)	Universities shall find the websites in order to maintain it. This website will predict and display the exact results to the students
6.	Scalability of the Solution	A future update shall have chat space comprising faculty, current students and alumni. It can be scaled for universities all around the world.

3.4 PROBLEM SOLUTION FIT

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem.


Define CS, fit into CC
Explore AS, differentiate
Focus on J&P, tap into BE, understand RC
Focus on J&P, tap into BE, understand RC
Identify strong TR & EM
Extract online & offline CH of BE

Problem-Solution fit **University Admit Eligibility Predictor** **Team ID: PNT2022TMID35582**

1. CUSTOMER SEGMENT(S) CS

Recently graduated students who are waiting to join in prestigious universities.

6. CUSTOMER CONSTRAINTS CC

Users would have to feed confidential information to the model so a certain section of customers might refrain from using the predictor due to a fear of data misuse.

5. AVAILABLE SOLUTIONS AS

- We will also consider certain non-academic factors that play a role in the admission process of some universities, thereby further enhancing the reliability of the predictor.
- We will put the model through rigorous tests in order to boost the accuracy of the predictor.

2. JOBS-TO-BE-DONE / PROBLEMS J&P

- The major task is to design a university admission prediction system and to provide a probabilistic insight into the university rating, cutoffs, intake count and the students' university preferences.
- It is indeed a cumbersome task for students to find their best-suited university and course for their further post graduation.
- The students are to be provided with a list of universities where admission is feasible so that the student can choose from the list.

9. PROBLEM ROOT CAUSE RC

- There may not be a single place where the students can find all the admission related information of the universities.
- The students may not be aware of the eligibility criteria of various universities in and around the world.
- The admission criteria of the colleges may not be consistent with the information provided by agents.

7. BEHAVIOUR BE

- The most important aspect of the predictor from a customer's POV is its accuracy, since they would go through with their admissions based on its results.
- For a customer, data security is of utmost importance.

3. TRIGGERS TR

- Students often get tensed and anxious about their admission chances of their desired universities.
- The students' peers may get lot of colleges to choose from, with lesser time and effort and lesser expenses.

4. EMOTIONS: BEFORE / AFTER EM

Before: Insecure and unaware of the process, suffering to select the best-suited university.
After: Secure, user-friendly and aware of process. Reduced cost and does not miss out best universities.

10. YOUR SOLUTION SL

- The focus is to reduce the time, effort and money spent on finding the universities where admission is feasible for pursuing higher education.
- The input to the system are student's academic details which includes CGPA, Scores in GRE, TOEFL, resume, LOR, SOP and other university eligibility features.
- The system uses a pre-trained machine model to predict the feasibility of admission in desired university based on the provided student data.
- The output of the system is the list of possible universities for the student to apply for admission.

8. CHANNELS of BEHAVIOUR CH

ONLINE

- The students may browse the Internet to research about their desired universities and get to know required information.
- This is a time-consuming task and may miss out some universities of interest

OFFLINE

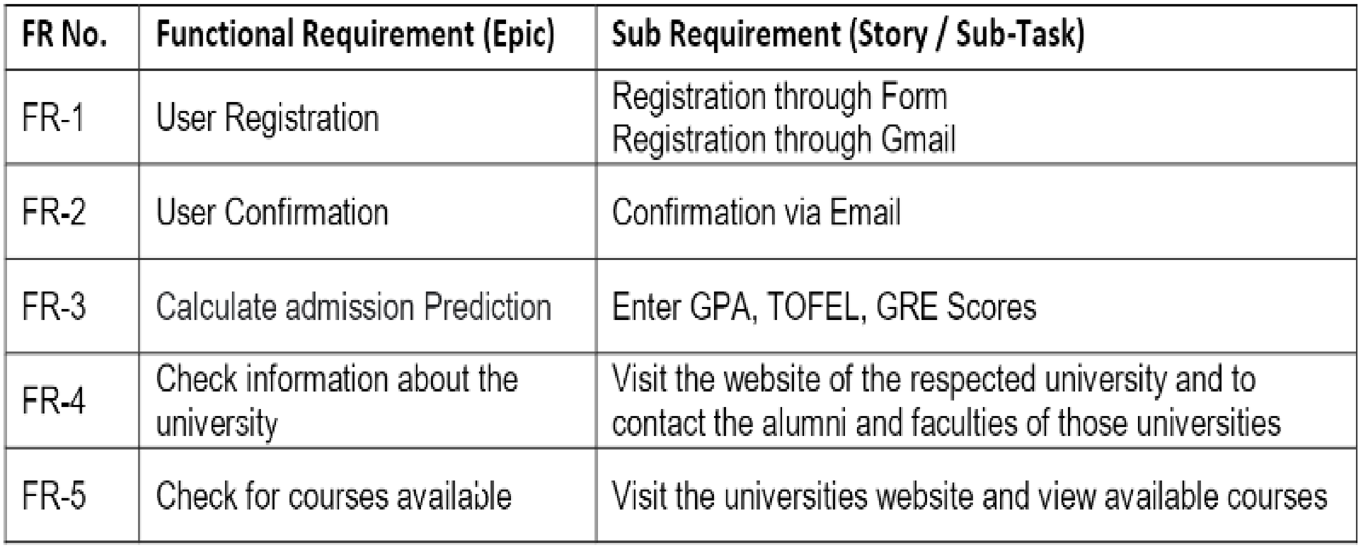
- Visit the desired universities in person and gather admission details.
- This incurs extra effort and expenses.

1. REQUIREMENT ANALYSIS

1.1. FUNCTIONAL REQUIREMENT

Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describe all the cases where the system uses the functional requirements, these are captured in use cases.

Following are the functional requirements of the proposed solution

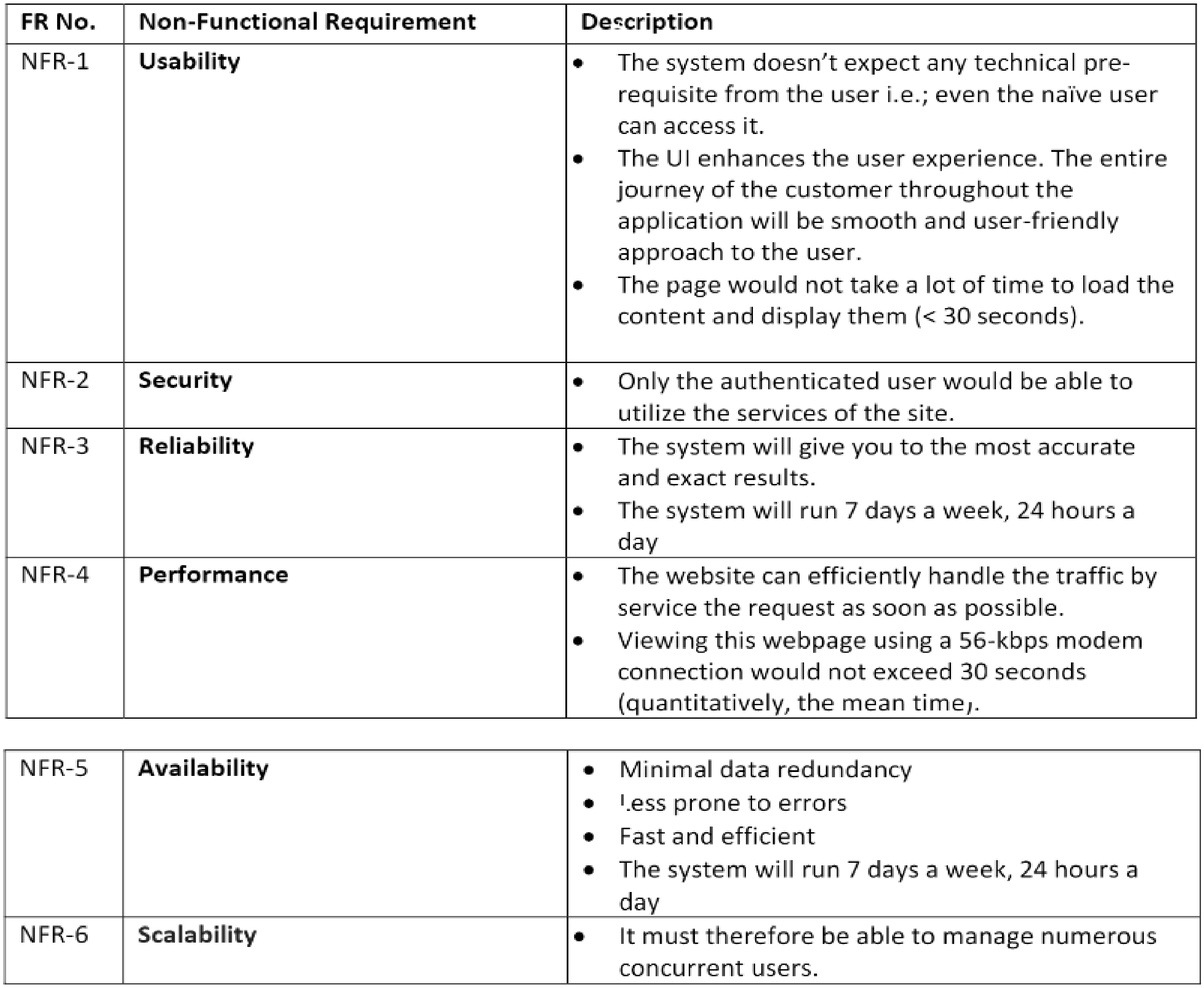


FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email
FR-3	Calculate admission Prediction	Enter GPA, TOFEL, GRE Scores
FR-4	Check information about the university	Visit the website of the respected university and to contact the alumni and faculties of those universities
FR-5	Check for courses available	Visit the universities website and view available courses

1.1. NON-FUNCTIONAL REQUIREMENTS

In systems engineering and requirements engineering, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours.

Following are the non-functional requirements of the proposed solution.

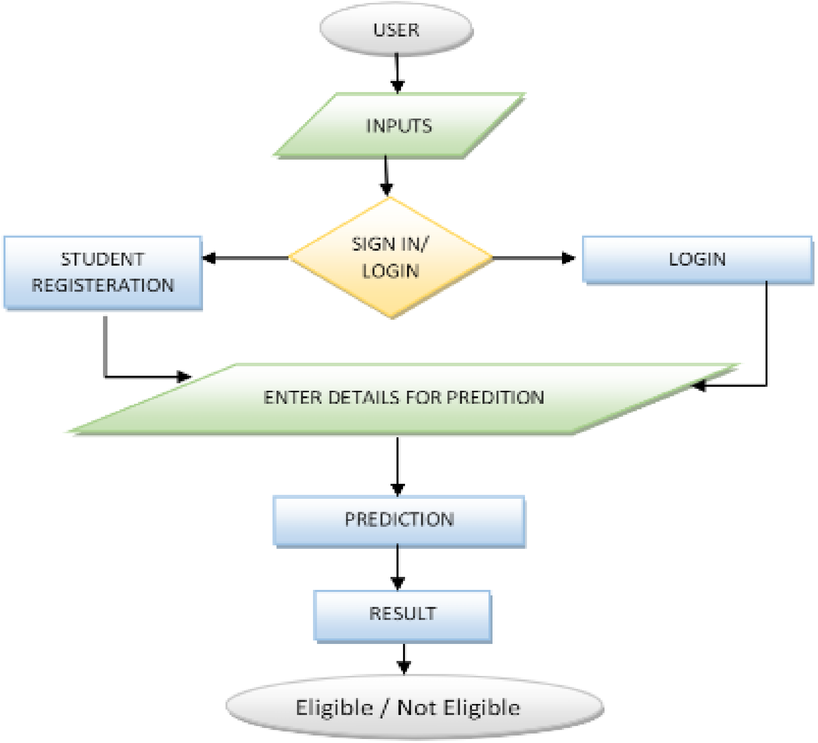


FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	• The system doesn't expect any technical pre-requisite from the user i.e.; even the naïve user can access it. • The UI enhances the user experience. The entire journey of the customer throughout the application will be smooth and user-friendly approach to the user. • The page would not take a lot of time to load the content and display them (< 30 seconds).
NFR-2	**Security**	• Only the authenticated user would be able to utilize the services of the site.
NFR-3	**Reliability**	• The system will give you to the most accurate and exact results. • The system will run 7 days a week, 24 hours a day
NFR-4	**Performance**	• The website can efficiently handle the traffic by service the request as soon as possible. • Viewing this webpage using a 56-kbps modem connection would not exceed 30 seconds (quantitatively, the mean time).
NFR-5	**Availability**	• Minimal data redundancy • Less prone to errors • Fast and efficient • The system will run 7 days a week, 24 hours a day
NFR-6	**Scalability**	• It must therefore be able to manage numerous concurrent users.

1. PROJECT DESIGN

1.1. DATA FLOW DIAGRAMS

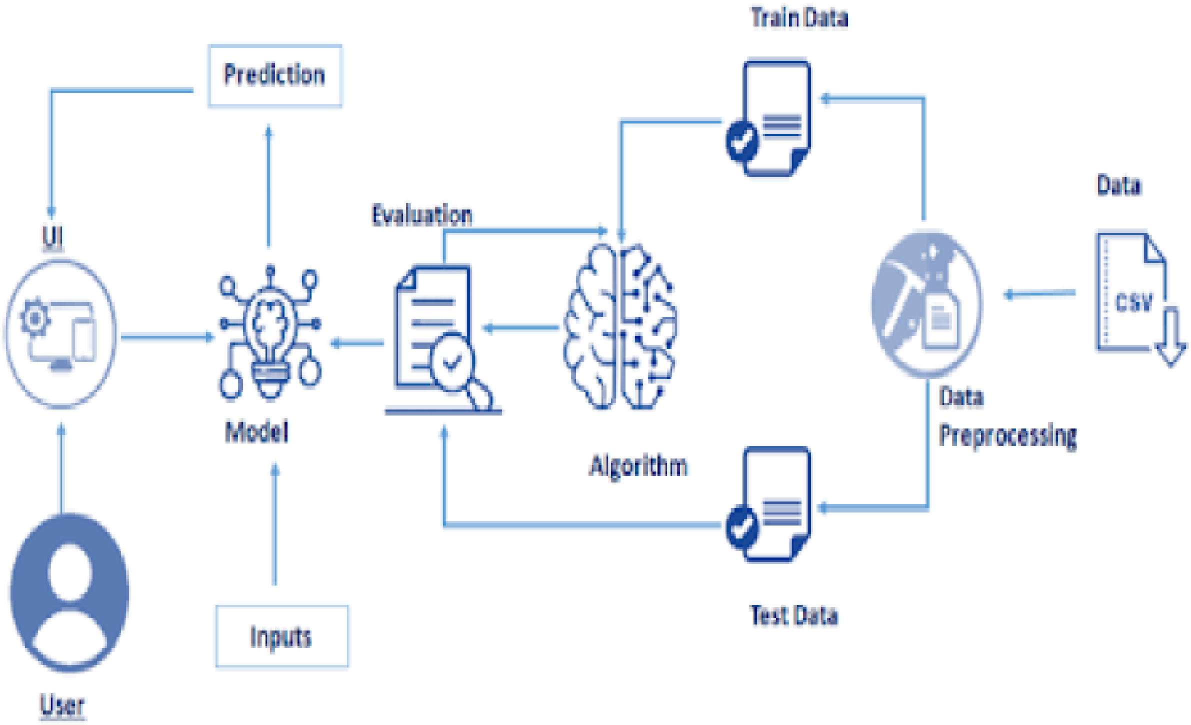
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



1.1. SOLUTION & TECHNICAL ARCHITECTURE

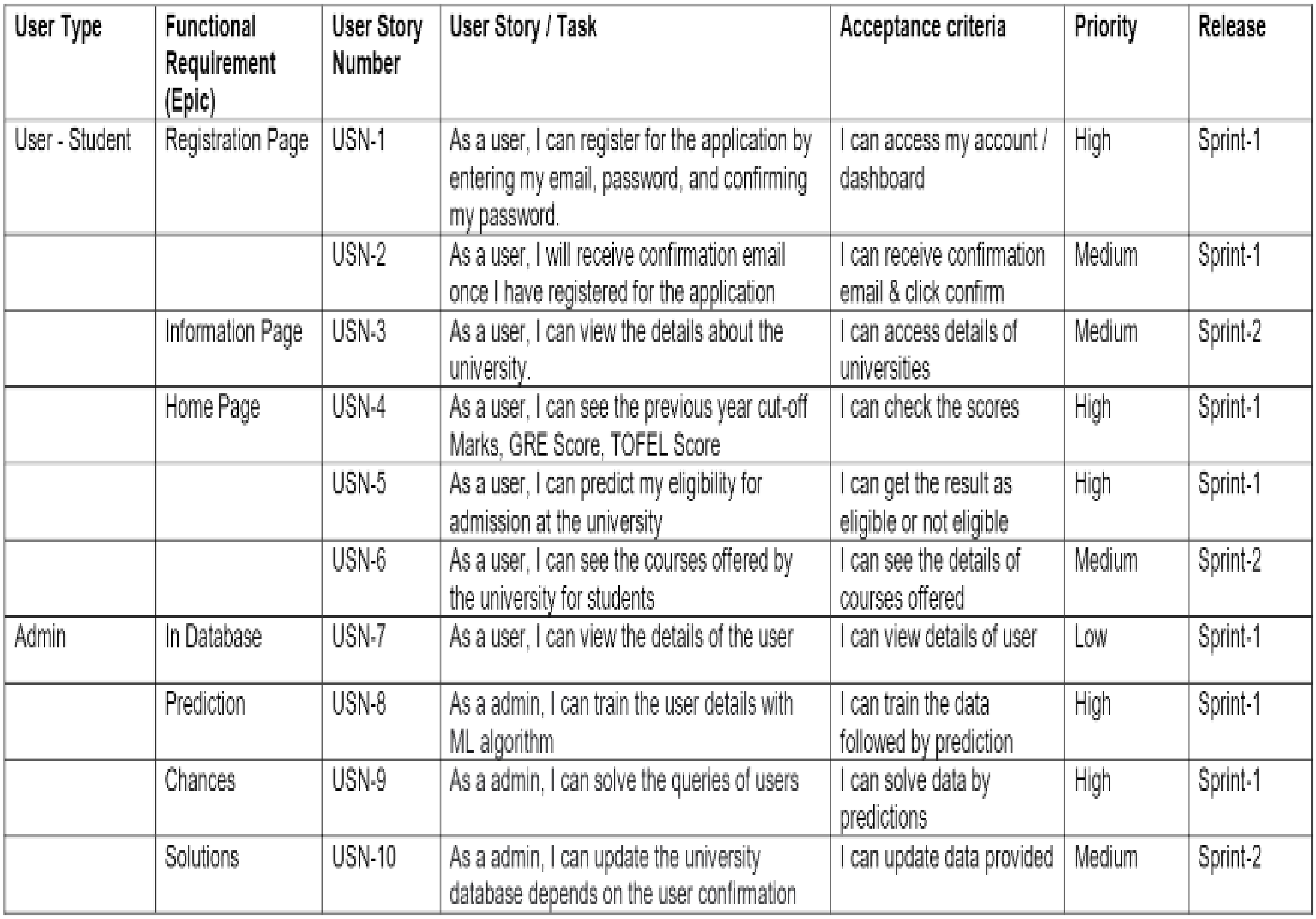
Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions.
Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.



1.1. USER STORIES

A user story is an informal, general explanation of a software feature written from the perspective of the end user or customer. The purpose of a user story is to articulate how a piece of work will deliver a particular value back to the customer

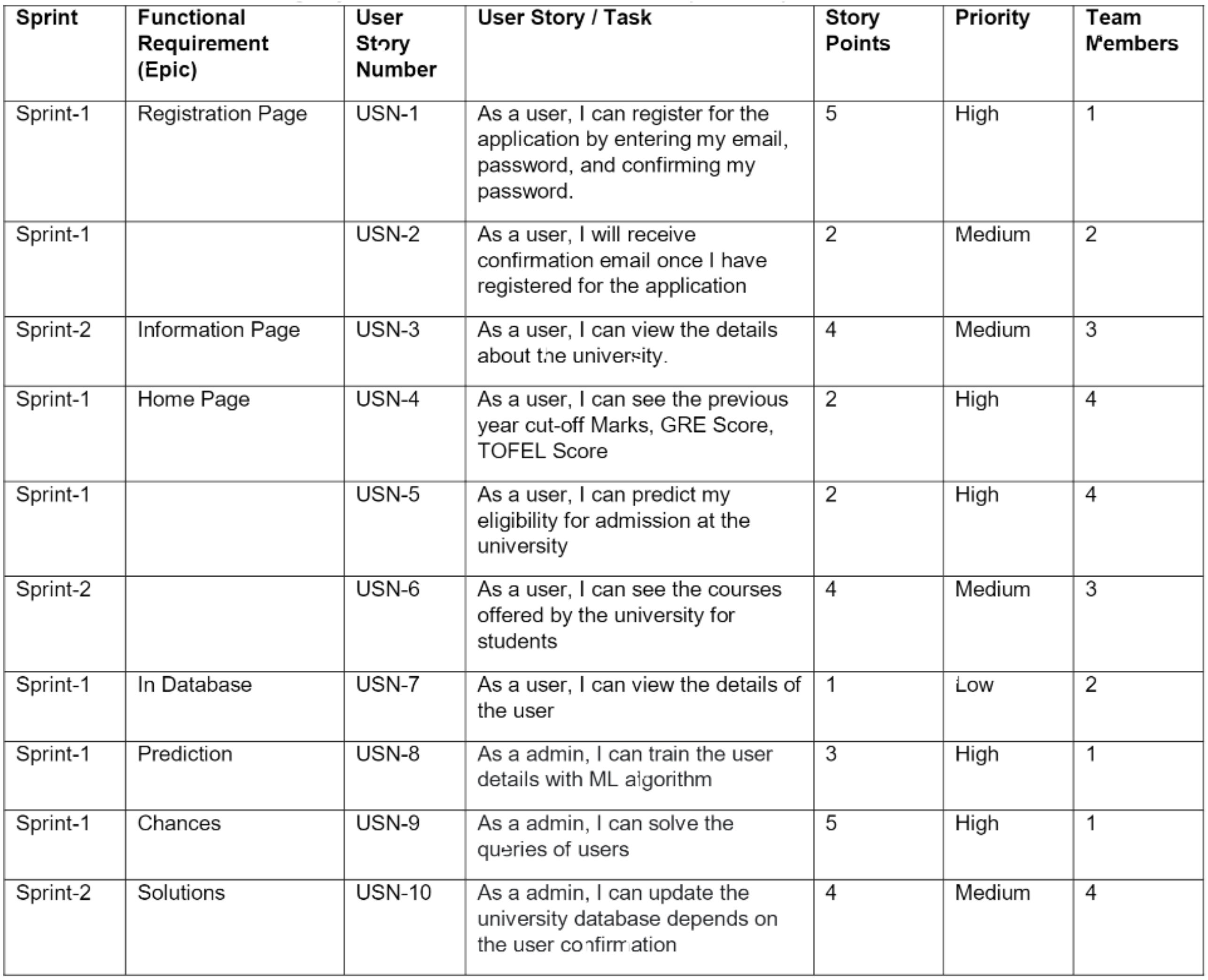


User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
User - Student	Registration Page	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	Medium	Sprint-1
	Information Page	USN-3	As a user, I can view the details about the university.	I can access details of universities	Medium	Sprint-2
	Home Page	USN-4	As a user, I can see the previous year cut-off Marks, GRE Score, TOFEL Score	I can check the scores	High	Sprint-1
		USN-5	As a user, I can predict my eligibility for admission at the university	I can get the result as eligible or not eligible	High	Sprint-1
		USN-6	As a user, I can see the courses offered by the university for students	I can see the details of courses offered	Medium	Sprint-2
Admin	In Database	USN-7	As a user, I can view the details of the user	I can view details of user	Low	Sprint-1
	Prediction	USN-8	As a admin, I can train the user details with ML algorithm	I can train the data followed by prediction	High	Sprint-1
	Chances	USN-9	As a admin, I can solve the queries of users	I can solve data by predictions	High	Sprint-1
	Solutions	USN-10	As a admin, I can update the university database depends on the user confirmation	I can update data provided	Medium	Sprint-2

PROJECT PLANNING & SCHEDULING

1.1. SPRINT PLANNING & ESTIMATION

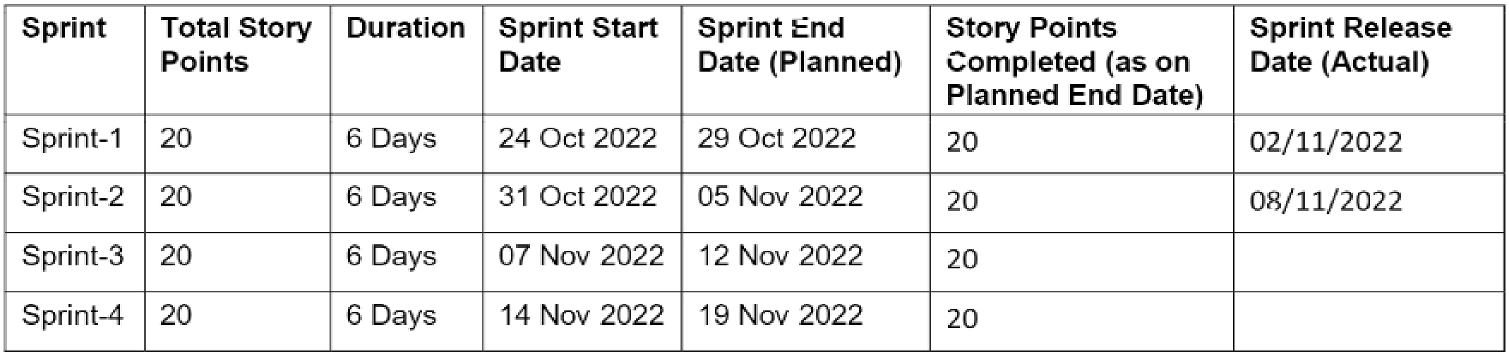
In Scrum Projects, Estimation is done by the entire team during Sprint Planning Meeting. The objective of the Estimation would be to consider the User Stories for the Sprint by Priority and by the Ability of the team to deliver during the Time Box of the Sprint.



Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration Page	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	5	High	1
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	2	Medium	2
Sprint-2	Information Page	USN-3	As a user, I can view the details about the university.	4	Medium	3
Sprint-1	Home Page	USN-4	As a user, I can see the previous year cut-off Marks, GRE Score, TOFEL Score	2	High	4
Sprint-1		USN-5	As a user, I can predict my eligibility for admission at the university	2	High	4
Sprint-2		USN-6	As a user, I can see the courses offered by the university for students	4	Medium	3
Sprint-1	In Database	USN-7	As a user, I can view the details of the user	1	Low	2
Sprint-1	Prediction	USN-8	As a admin, I can train the user details with ML algorithm	3	High	1
Sprint-1	Chances	USN-9	As a admin, I can solve the queries of users	5	High	1
Sprint-2	Solutions	USN-10	As a admin, I can update the university database depends on the user confirmation	4	Medium	4

1.1. SPRINT DELIVERY SCHEDULE

A sprint schedule is a document that outlines sprint planning from end to end. It's one of the first steps in the agile sprint planning process—and something that requires adequate research, planning, and communication.



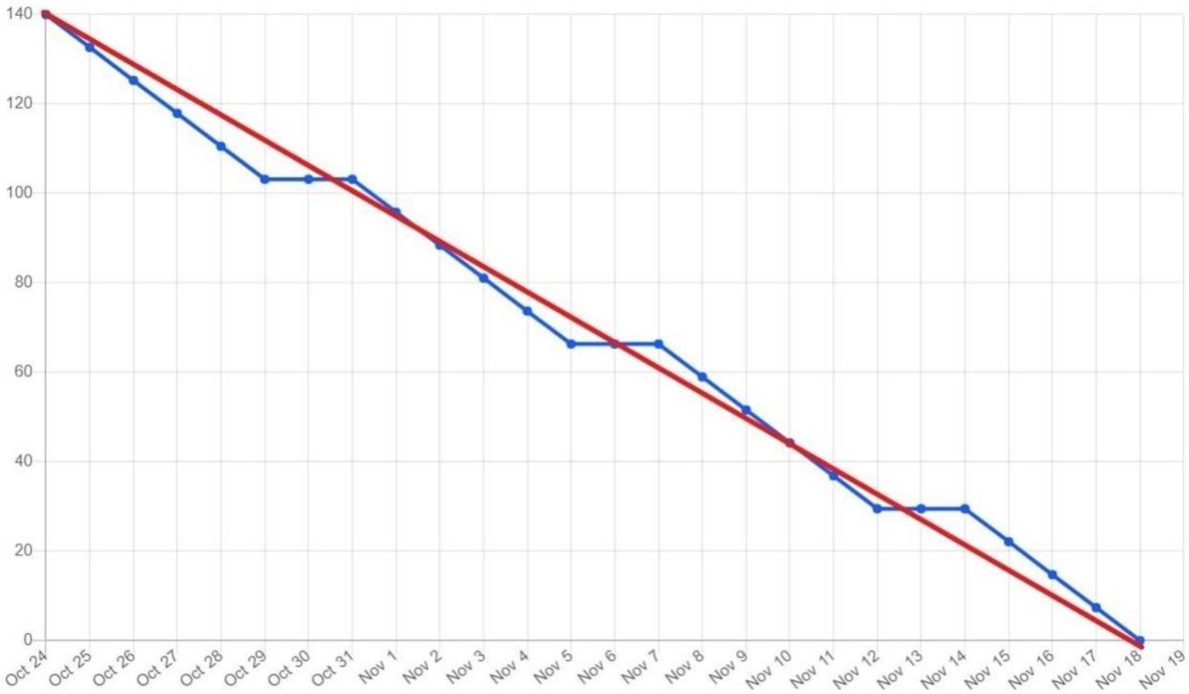
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	02/11/2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	08/11/2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	

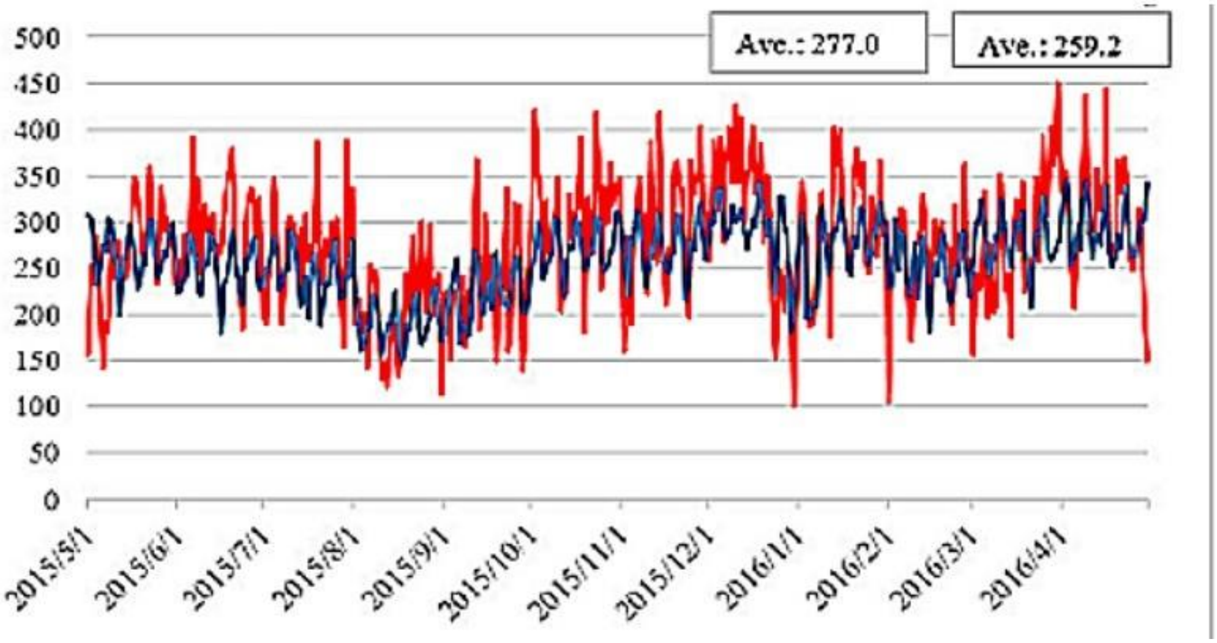
1.1. REPORTS FROM JIRA

Jira helps teams plan, assign, track, report, and manage work and brings teams together for everything from agile software development and customer support to startups and enterprises. Software teams build better with Jira Software, the #1 tool for agile teams.

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.





2. CODING & SOLUITONING

2.1DATA DICTIONARY

We have used the dataset for building the ML Model and training it. Some of the datas from our dataset are tabulated below.

Serial No.	**GRE Score**	**TOEFL Score**	**University Rating**	**SOP**	**LOR**	**CGPA**	**Research**	**Chance of Admit**
1	337	118	4	4.5	4.5	9.65	1	0.92
2	324	107	4	4	4.5	8.87	1	0.76
3	316	104	3	3	3.5	8	1	0.72

4	322	110	3	3.5	2.5	8.67	1	0.8
5	314	103	2	2	3	8.21	0	0.65
6	330	115	5	4.5	3	9.34	1	0.9
7	321	109	3	3	4	8.2	1	0.75
8	308	101	2	3	4	7.9	0	0.68
9	302	102	1	2	1.5	8	0	0.5
10	323	108	3	3.5	3	8.6	0	0.45

2.1. LIBRARIES USED

- Pandas
- Numpy
- Scikit learn
- Matplotlib
- Seaborn

2.2. TECHNOLOGIES USED

- Software
    - Python
    - Anaconda
    - Jupyter Notebook
    - Windows 11
    - IBM Watson Studio
- Hardware
    - Processor - Quad Core
    - Hard Disk and SSD
    - Memory - 2 GB and Above RAM

2.3. EVALUATION METRIC

The evaluation metric for this competition is 100*RMSLE where RMSLE is Root of Mean Squared Logarithmic Error across all entries in the test set.

2.4. INITIAL APPROACH

- Simple Linear Regression model without any feature engineering and data transformation which gave a RMSE : 196.402.
- Without feature engineering and data transformation, the model did not perform well and could'nt give a good score.
- Post applying feature engineering and data transformation (log and log1p transformation), Linear Regression model gave a RMSLE score of 0.734.

2.5. ADVANCED MODELS

- With improvised feature engineering, built advanced models using Ensemble techniques and other Regressor algorithms.
- Decision Tree Regressors performed well on the model which gave much reduced RMSLE.
- With proper hyper-parameter tuning, Decision Tree Regressor performed well on the model and gave the lease RMSLE of 0.5237.

3. TESTING

3.1. TEST CASES

Test case ID	Feature Type	Component	Test Scenario
LoginPage_TC 001	Functional	Home Page	Verify user is able tosee the Login/Signup popup when user clicked onMy account button
LoginPage_TC_002	UI	Home Page/ Dem0 2	Verify the UI elementsin home page
LoginPage_TC_003	Functional	Chance	Verify that the Candidate Having chance to Admit/Not
LoginPage_TC_004	Functional	NoChance	Verify that the Candidate Having chance to Admit/Not

Execution Steps	**Tested URL**
1. Run the Python Flask code 2. Open your browser 3. Enter the URL and click go 4. Enter the required details 5. Click the predict button	http://127.0.0.1:5000/
6. As per the Student details, the score will be calculated 7. If score is above 50%, then the Chance page will appear	http://127.0.0.1:5000/chance/99.13900931466966
8. If predicted score is below 50%, then the nochance page will appear	http://127.0.0.1:5000/nochance/38.72796854402236

S. No.	**Expected Result**	**Actual Result**	**Status**	**Comments**
1.	Working as expected	Working as expected	Pass	Perfect Working
2.	Working as expected	Working as expected	Pass	Perfect Working
3.	Working as expected	Working as expected	Pass	Perfect Working
4.	Working as expected	Working as expected	Pass	Perfect Working

3.2. USER ACCEPTANCE TESTING

Defect Analysis:

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved.

Resolution	Severity1	Severity2	Severity3	Severity4	Sub Total
By Design	7	3	2	3	15
Duplicate	1	0	3	0	4
Extemal	2	3	0	1	6
Fixed	7	3	2	17	29
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won'tFix	0	3	2	1	6
Totals	17	12	11	23	63

Test Case Analysis:

This report shows the number of test cases that have passed, failed and untested.

Section	**TotalCases**	**Not Tested**	**Fail**	**Pass**
PrintEngine	7	0	0	7
ClientAppIication	51	0	0	51
Security	2	0	0	2
OutsourceShipping	3	0	0	3
ExceptionReporting	9	0	0	9
FinalReportOutput	4	0	0	4
VersionControI	2	0	0	2

4. RESULTS

4.1. PERFORMANCE METRICS

S. No.	Parameter	Values	Screenshot
1.	Metrics	Regression Model: MAE -0.03790692243018498, MSE-O.0030587534363O7664, RMSE — 0.05530599819465936, R2 score — 0.8647260941958439 Classification Model: Confusion Matrix - , Accuray Score- & Classification Report	
2.	Tune the Model	Hyperparameter Tuning: 0.9166666666666666 Validation Method —LinearRegression model	

5. ADVATAGES & DISADVANTAGES

Advantages:

- It helps student for making decision for choosing a right college.
- Here the chance of occurrence of error is less when compared with the existing system.
- Avoids data redundancy and inconsistency.
- It is fast, efficient and reliable.

Disadvantages:

- Machine errors are unavoidable when occurred. (Hardware failure, network failure, others).
- The predictions made are not 100% accurate but accurate to an acceptable value.

6. APPLICATIONS

- Reach to geographically scattered student.
- Reducing time in activities

- Paperless admission with reduced man power
- Operational efficiency

7. CONCLUSION

The project employs a Random forest regressor to forecast the output, and a web application is created utilising a variety of technologies, including Python, HTML5, CSS, Flask, Scikit, Matplot, Numpy, Pandas, Seaborn, and other libraries, to make the user interface (UI) more accessible and simple. The web application may be viewed from any location with an internet connection after it has been deployed. With this project, you can estimate your eligibility for admission to a ranked university in a fraction of the time.

8. FUTURE SCOPE

Some of the future scopes of this project are:

- This can be implemented quickly and properly during admission process.
- This can be accessed anytime, anywhere, since it is a web application provided only an internet connection.
- The user does not need to travel a long distance for the admission and his/her time is also saved as a result of this automated system.

9. APPENDIX

Source Code:

1. HTML Codes:

a. INDEX.HTML:

```
<!DOCTYPE html>
<html>
   <head>
      <title>University Eligiblity Predictor</title>
      <meta charset="Utf-8">
      <meta name="keyword" content="Eligiblity Predictor">
      <meta name="description" content="gre, tofel, University Eligiblity, Predictor">
      <meta name="author" content="pradeepa sowmya suja vedha">
      <meta name="viewport" content="width=device-width ,initial_selected=1.0">
      <link rel="stylesheet" href="css/style.css">
      <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

   </head>
```

```
  <body>
    <div class="transbox">
      <form method="POST" action="">
        <br><br>
        <div class="heading">University Admission Prediction System</div><br>
        <div class="heading">Enter your details: </div><br><br><br>
        Enter GRE Score     : <input type="number" name="gre"><br><br>
        Enter TOEFL Score   : <input type="number" name="tofel" ><br><br>
        <lable for="University Rating">
        University Rating   : </lable>
        <input type="radio" name="urating" value=1>1</input>
        <input type="radio" name="urating" value=2>2</input>
        <input type="radio" name="urating" value=3>3</input>
        <input type="radio" name="urating" value=4>4</input>
        <input type="radio" name="urating" value=5>5</input><br>
        <br>
        Enter SOP          : <input type="number" name="sop"><br><br>
        Enter LOR          : <input type="number" name="lor" ><br><br>
        Enter CGPA          : <input type="number" name="cgpa" ><br><br>
        <lable>Are you interesed in Research :</lable>
        <input type="radio" name="research" value="1">Interested</input>
        <input type="radio" name="research" value="2">Not Interested</input><br><br>
        <br><br>
        <input type="submit" name="predict" id="Predict" value="Predict">
      </form>
    </div>
</div>

  <div id="popup" class="pop">
    <div class="popup-content">
    <span class="close">&times;</span>
    <p>Result of Prediction</p>
  </div>

    <script>
      var pop = document.getElementById("popup");
      var btn = document.getElementById("Predict");
      var span = document.getElementsByClassName("close")[0];
```

```
            btn.onclick = function() {
                event.preventDefault()
                pop.style.display = "block";
            }

            span.onclick = function() {
                pop.style.display = "none";
            }

            window.onclick = function(event) {
                if (event.target == pop) {
                    pop.style.display = "none";
                }
            }
        </script>
    </body>
</html>
```

b.CSS file

```
div.heading{
  font-size:24px;
    text-align: center;
}

body{
  background-image:url('images/university.jpg');
  background-position:center;
  background-size:cover;
    background-color: rgb(27, 27, 27);
  font-family:Cambria;
  font-size:16px;
  color:rgb(255, 253, 253);
}

form{
 padding: 5px;
 margin-left:auto;
 margin-right:auto;
 border-radius:25px;
 width:500px;
 height:600px;
 table-layout:fixed;
}

input[type=submit]{
    width:15%;
    height:30px;
    border-radius:50px;
    background-color:rgb(237, 242, 243);
}

div.transbox {
 margin-left:auto;
 margin-right:auto;
 border-radius:25px;
 width:500px;
 height:600px;
  background-color: rgba(255,255,255,0.3);/*rgba(0,151,19,0.1)*/
  border: 1px solid rgb(255, 255, 255);
 }


 .pop {
    display: none; /* Hidden by default */
    position: fixed; /* Stay in place */
    z-index: 1; /* Sit on top */
    padding-top: 100px; /* Location of the box */
    left: 0;
    top: 0;
    width: 100%; /* Full width */
    height: 100%; /* Full height */
    overflow: auto; /* Enable scroll if needed */
    background-color: rgb(104, 102, 102); /* Fallback color */
    background-color: rgba(75, 75, 75, 0.4); /* Black w/ opacity */
}
```

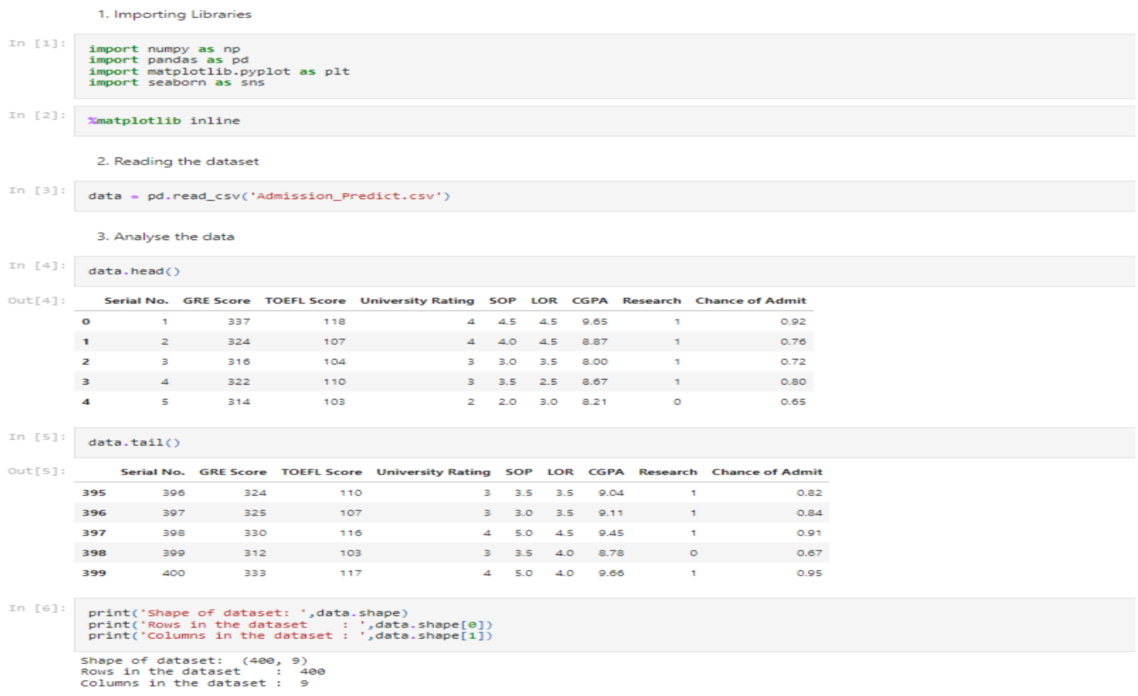
```
/* Modal Content */
.popup-content {
   background-color: #111111;
   margin: auto;
   padding: 20px;
   border: 1px solid #888;
   width: 80%;
}

/* The Close Button */
.close {
   color: #aaaaaa;
   float: right;
   font-size: 28px;
   font-weight: bold;
}

.close:hover,
.close:focus {
   color: #000;
   text-decoration: none;
   cursor: pointer;
}
```

2. PYTHON CODE:

a. APP.PY:

```
import pandas as pd
from flask import Flask,request, jsonify, render_template, redirect, url_for
import requests
import json

API_KEY = "13S6-gvuJHw0EgY7HAmtl8ae5tQlGcbahHYBYAacEOQn"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey":API_KEY,"grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
app = Flask(_name_, template_folder='templates')

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['GET', 'post'])
def predict():
    GRE_Score = int(request.form['GRE Score'])
    TOEFL_Score = int(request.form['TOEFL Score'])
    University_Rating = int(request.form['University Rating'])
    SOP = float(request.form['SOP'])
    LOR = float(request.form['LOR'])
    CGPA = float(request.form['CGPA'])
    Research = int(request.form['research_radio'])
    final_features = [[GRE_Score, TOEFL_Score, University_Rating, SOP, LOR, CGPA]]
    payload_scoring = {'input_data': [
        {'field': [["GRE Score", "TOEFL Score", "University Rating", "SOP", "LOR ", "CGPA"]],
         'values': final_features}]}
    print("hello")
    response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/3ef17bf1-d7c8-475c-88b4-
1216d3e59253/predictions?version=2022-11-17',
        json=payload_scoring,
        headers={'Authorization': 'Bearer ' + mltoken})
    print("scoring response")
    pred = response_scoring.json()
    print(pred)
    output = pred['predictions'][0]['values'][0][0]
    print(output)
    if output > 0.5:
        return redirect(url_for('chance', percent=output * 100))
    else:
        return redirect(url_for('no_chance', percent=output * 100))

@app.route("/chance/<percent>")
def chance(percent):
    return render_template("chance.html", content=[percent])

@app.route("/nochance/<percent>")
def no_chance(percent):
    return render_template("nochance.html", content=[percent])

if __name___== "_main_":
    app.run(debug=True)
```

b. REGRESSION METHODS.IPYNB



1. Importing Libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
%matplotlib inline
```

2. Reading the dataset

In [3]:

```
data = pd.read_csv('Admission_Predict.csv')
```

3. Analyse the data

In [4]:

```
data.head()
```

Out[4]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

In [5]:

```
data.tail()
```

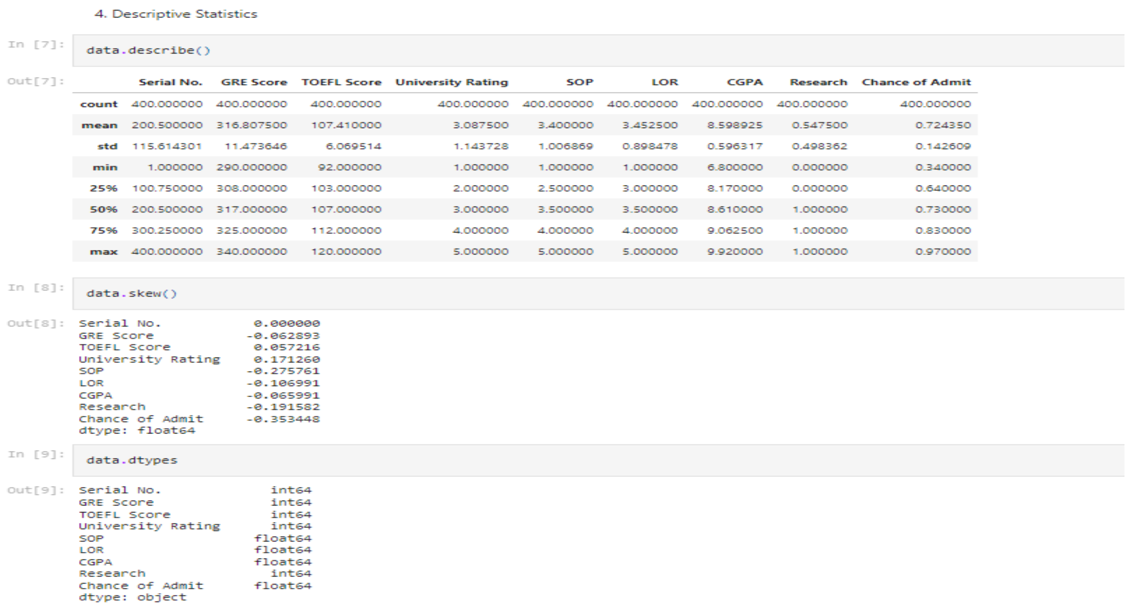
Out[5]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
395	396	324	110	3	3.5	3.5	9.04	1	0.82
396	397	325	107	3	3.0	3.5	9.11	1	0.84
397	398	330	116	4	5.0	4.5	9.45	1	0.91
398	399	312	103	3	3.5	4.0	8.78	0	0.67
399	400	333	117	4	5.0	4.0	9.66	1	0.95

In [6]:

```
print('Shape of dataset: ',data.shape)
print('Rows in the dataset     : ',data.shape[0])
print('Columns in the dataset : ',data.shape[1])
```

```
Shape of dataset:  (400, 9)
Rows in the dataset     :  400
Columns in the dataset :  9
```



4. Descriptive Statistics

In [7]:

```
data.describe()
```

Out[7]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	200.500000	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.547500	0.724350
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.498362	0.142609
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	0.340000
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.000000	0.640000
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.000000	0.730000
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.000000	0.830000
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.970000

In [8]:

```
data.skew()
```

Out[8]:

```
Serial No.            0.000000
GRE Score            -0.062893
TOEFL Score           0.057216
University Rating     0.171260
SOP                  -0.275761
LOR                  -0.106991
CGPA                 -0.065991
Research             -0.191582
Chance of Admit      -0.353448
dtype: float64
```

In [9]:

```
data.dtypes
```

Out[9]:

```
Serial No.             int64
GRE Score              int64
TOEFL Score            int64
University Rating      int64
SOP                  float64
LOR                  float64
CGPA                 float64
Research               int64
Chance of Admit      float64
dtype: object
```

5. Working with missing values

```
In [10]: print("Looking for missing values:")
         data.isnull().any()
```

```
         Looking for missing values:
Out[10]: Serial No.           False
         GRE Score            False
         TOEFL Score          False
         University Rating    False
         SOP                  False
         LOR                  False
         CGPA                 False
         Research             False
         Chance of Admit      False
         dtype: bool
```

```
In [11]: data.columns
```

```
Out[11]: Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
                'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
               dtype='object')
```

```
In [12]: data.drop('Serial No.', inplace=True, axis=1)
```

```
In [13]: data.columns
```

```
Out[13]: Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',
                'Research', 'Chance of Admit '],
               dtype='object')
```

```
In [14]: data.describe()
```

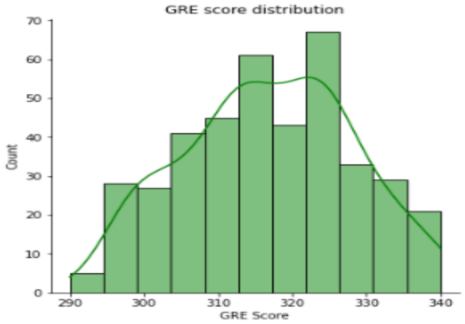
Out[14]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.547500	0.724350
std	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.498362	0.142609
min	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	0.340000
25%	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.000000	0.640000
50%	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.000000	0.730000
75%	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.000000	0.830000
max	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.970000

7. Data Visualization

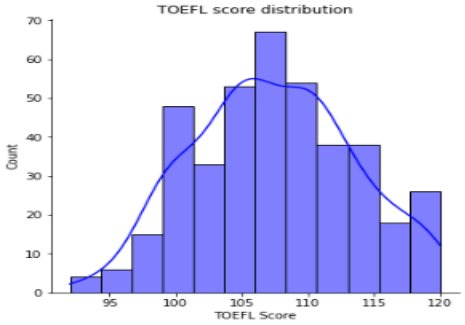
UNIVARIATE ANALYSIS

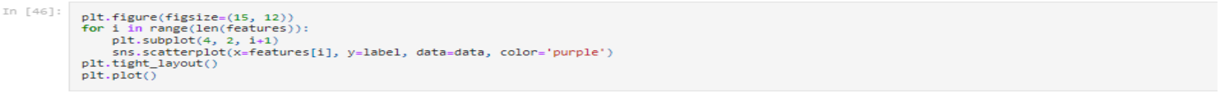
```
In [21]: sns.displot(x=data["GRE Score"], kde=True, color='green')
         plt.title("GRE score distribution");
```



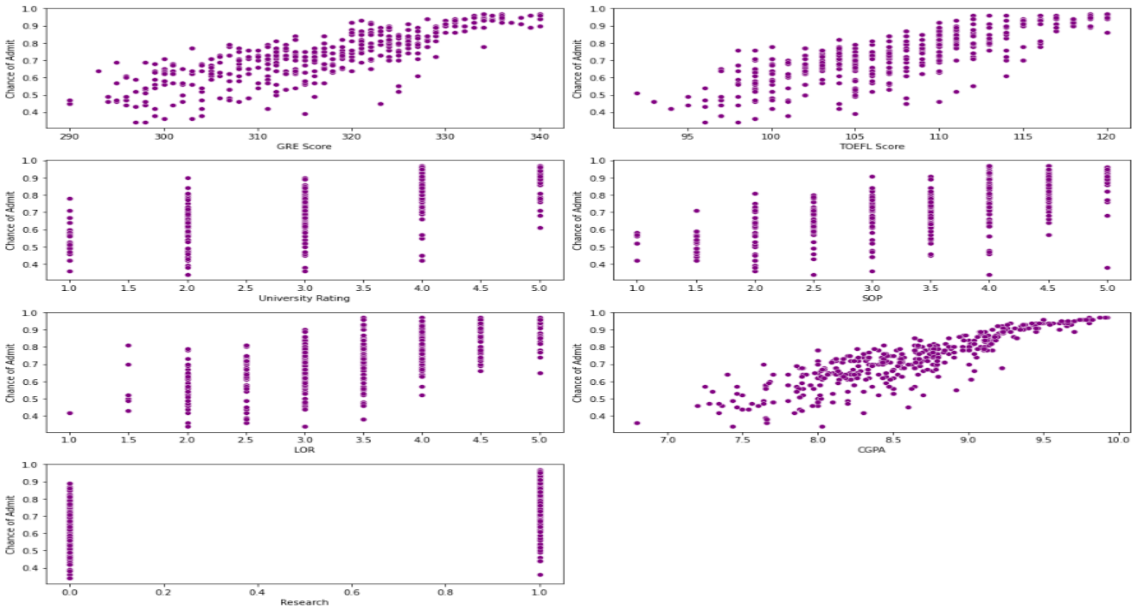
```
In [22]: # Observation for GRE Score: score is distributed in the range of 290 to 340
         # Peaks are observed between 310 to 330 indicating most applicants have scored in this range
```

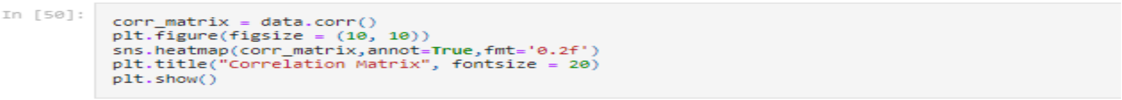
```
In [23]: sns.displot(x=data["TOEFL Score"], kde=True, color='blue')
         plt.title("TOEFL score distribution");
```



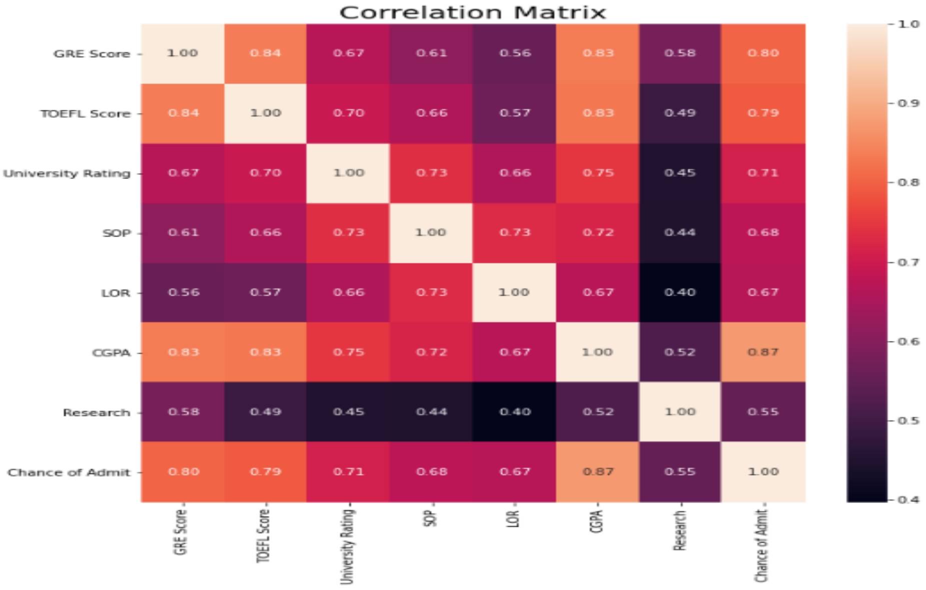


```
In [46]:  plt.figure(figsize=(15, 12))
           for i in range(len(features)):
               plt.subplot(4, 2, i+1)
               sns.scatterplot(x=features[i], y=label, data=data, color='purple')
           plt.tight_layout()
           plt.plot()

Out[46]:  []
```



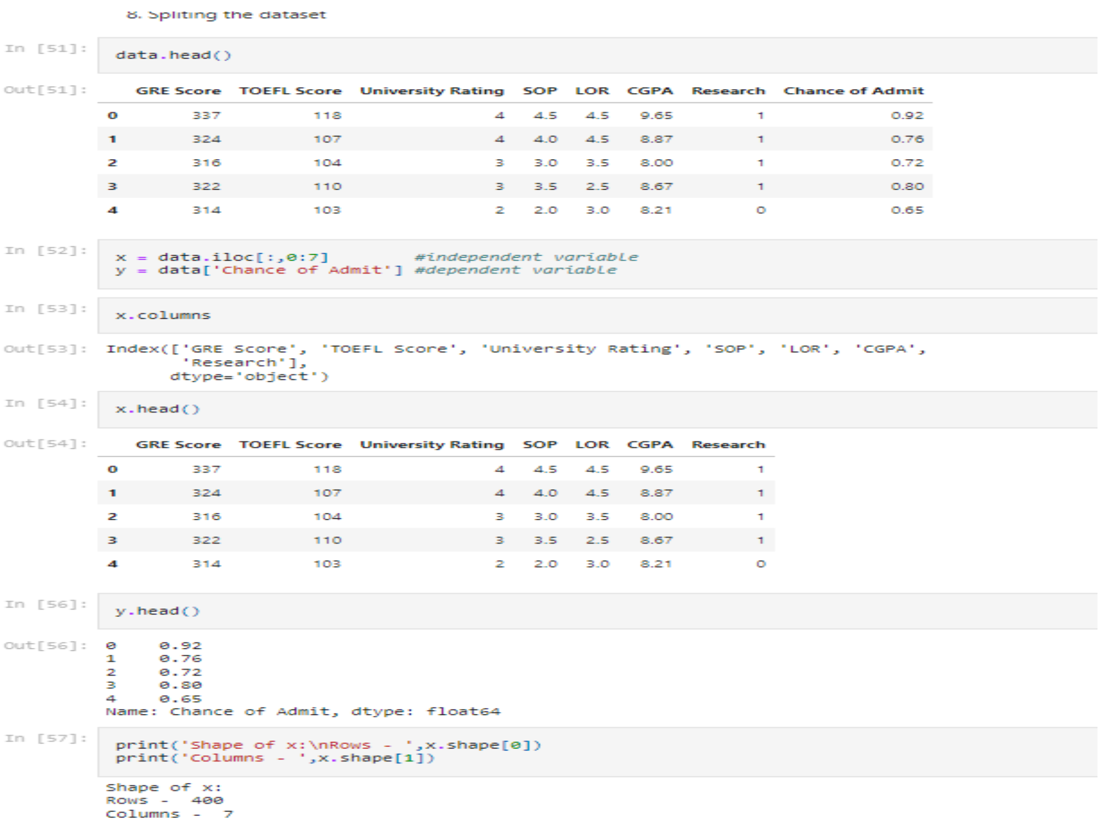


```
In [50]:  corr_matrix = data.corr()
           plt.figure(figsize = (10, 10))
           sns.heatmap(corr_matrix,annot=True,fmt='0.2f')
           plt.title("Correlation Matrix", fontsize = 20)
           plt.show()
```



Correlation Matrix

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
GRE Score	1.00	0.84	0.67	0.61	0.56	0.83	0.58	0.80
TOEFL Score	0.84	1.00	0.70	0.66	0.57	0.83	0.49	0.79
University Rating	0.67	0.70	1.00	0.73	0.66	0.75	0.45	0.71
SOP	0.61	0.66	0.73	1.00	0.73	0.72	0.44	0.68
LOR	0.56	0.57	0.66	0.73	1.00	0.67	0.40	0.67
CGPA	0.83	0.83	0.75	0.72	0.67	1.00	0.52	0.87
Research	0.58	0.49	0.45	0.44	0.40	0.52	1.00	0.55
Chance of Admit	0.80	0.79	0.71	0.68	0.67	0.87	0.55	1.00



8. Spliting the dataset

```
In [51]: data.head()
```

Out[51]:

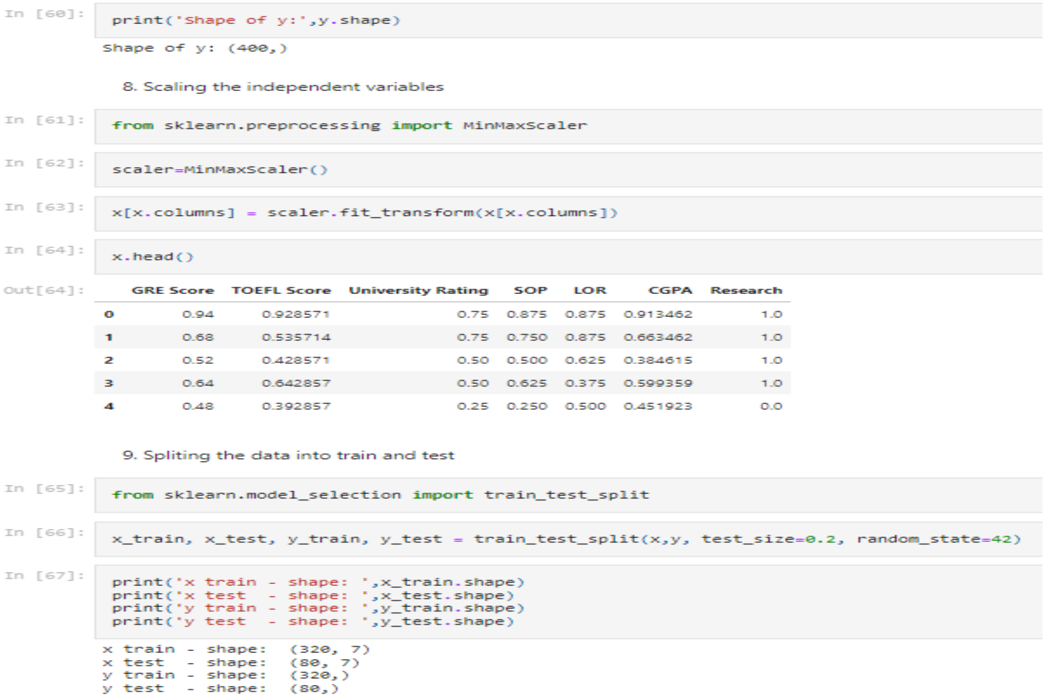
	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65

```
In [52]: x = data.iloc[:,0:7]         #independent variable
         y = data['Chance of Admit'] #dependent variable

In [53]: x.columns

Out[53]: Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR', 'CGPA',
                'Research'],
               dtype='object')

In [54]: x.head()
```

Out[54]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	337	118	4	4.5	4.5	9.65	1
1	324	107	4	4.0	4.5	8.87	1
2	316	104	3	3.0	3.5	8.00	1
3	322	110	3	3.5	2.5	8.67	1
4	314	103	2	2.0	3.0	8.21	0

```
In [56]: y.head()

Out[56]: 0    0.92
         1    0.76
         2    0.72
         3    0.80
         4    0.65
         Name: Chance of Admit, dtype: float64

In [57]: print('Shape of x:\nRows - ',x.shape[0])
         print('Columns - ',x.shape[1])

         Shape of x:
         Rows -  400
         Columns -  7
```



```
In [60]: print('Shape of y:',y.shape)

         Shape of y: (400,)
```

8. Scaling the independent variables

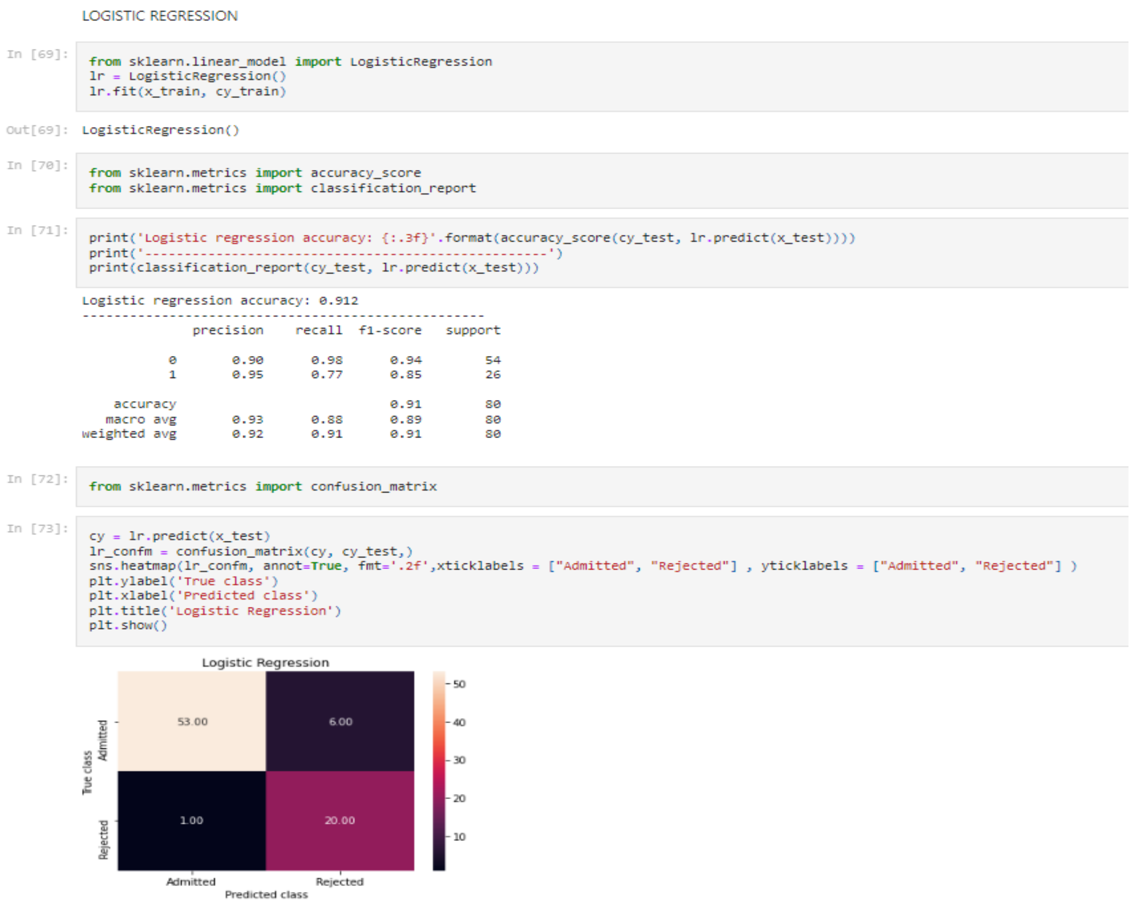
```
In [61]: from sklearn.preprocessing import MinMaxScaler

In [62]: scaler=MinMaxScaler()

In [63]: x[x.columns] = scaler.fit_transform(x[x.columns])

In [64]: x.head()
```

Out[64]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	0.94	0.928571	0.75	0.875	0.875	0.913462	1.0
1	0.68	0.535714	0.75	0.750	0.875	0.663462	1.0
2	0.52	0.428571	0.50	0.500	0.625	0.384615	1.0
3	0.64	0.642857	0.50	0.625	0.375	0.599359	1.0
4	0.48	0.392857	0.25	0.250	0.500	0.451923	0.0

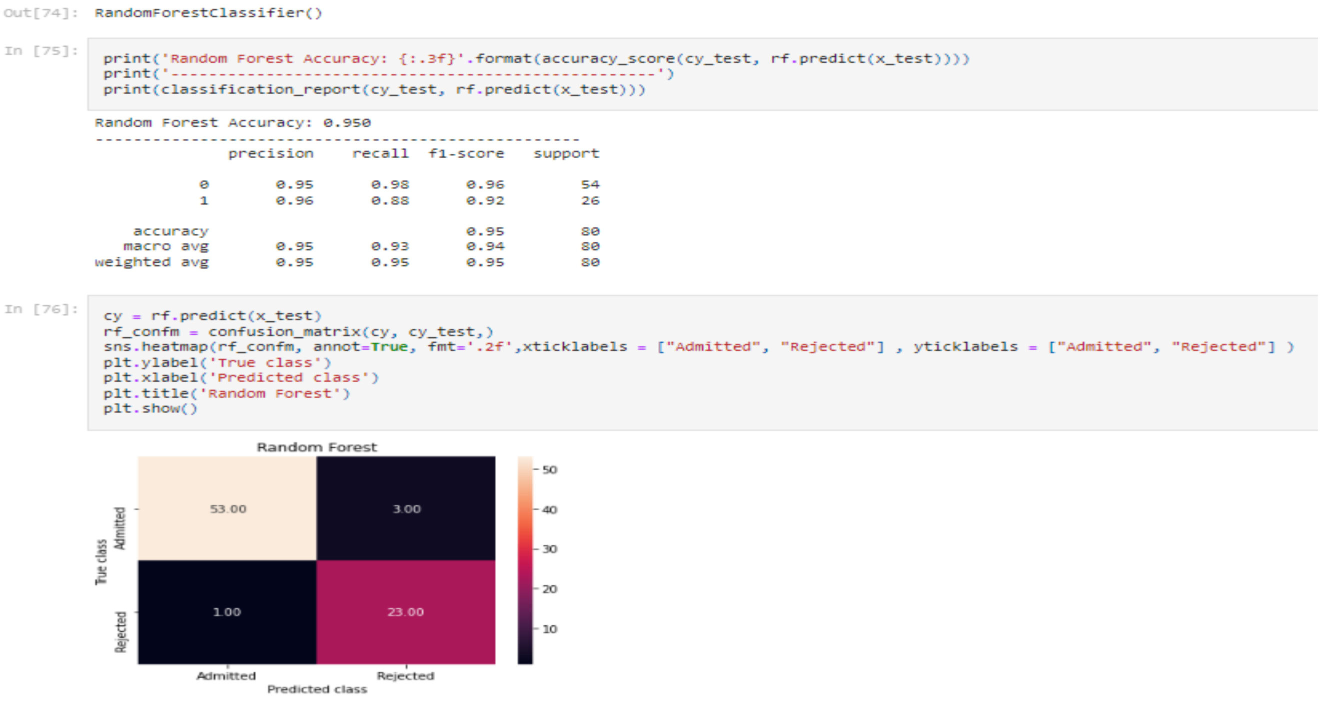
9. Spliting the data into train and test

```
In [65]: from sklearn.model_selection import train_test_split

In [66]: x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=42)

In [67]: print('x train - shape: ',x_train.shape)
         print('x test  - shape: ',x_test.shape)
         print('y train - shape: ',y_train.shape)
         print('y test  - shape: ',y_test.shape)

         x train - shape:  (320, 7)
         x test  - shape:  (80, 7)
         y train - shape:  (320,)
         y test  - shape:  (80,)
```

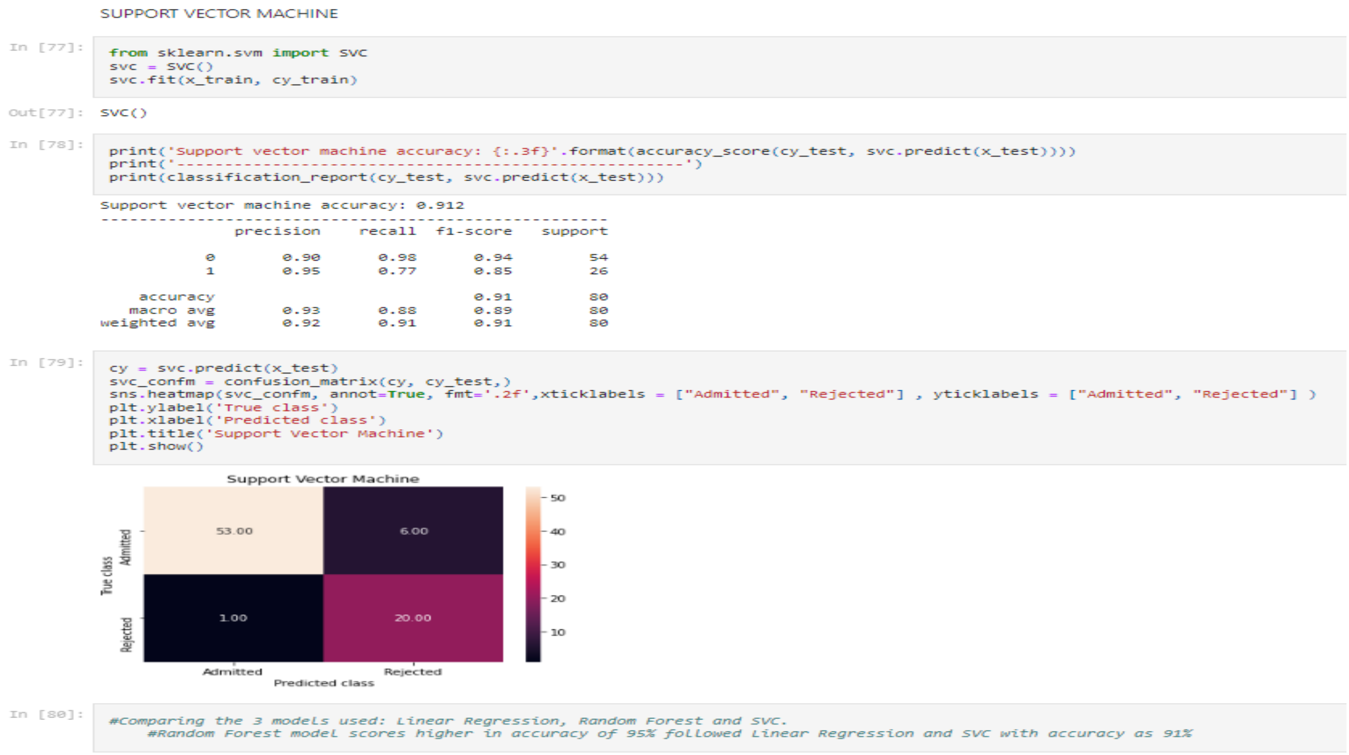


LOGISTIC REGRESSION

```
In [69]: from sklearn.linear_model import LogisticRegression
         lr = LogisticRegression()
         lr.fit(x_train, cy_train)

Out[69]: LogisticRegression()

In [70]: from sklearn.metrics import accuracy_score
         from sklearn.metrics import classification_report

In [71]: print('Logistic regression accuracy: {:.3f}'.format(accuracy_score(cy_test, lr.predict(x_test))))
         print('----------------------------------------------------')
         print(classification_report(cy_test, lr.predict(x_test)))

Logistic regression accuracy: 0.912
----------------------------------------------------
              precision    recall  f1-score   support

           0       0.90      0.98      0.94        54
           1       0.95      0.77      0.85        26

    accuracy                           0.91        80
   macro avg       0.93      0.88      0.89        80
weighted avg       0.92      0.91      0.91        80

In [72]: from sklearn.metrics import confusion_matrix

In [73]: cy = lr.predict(x_test)
         lr_confm = confusion_matrix(cy, cy_test,)
         sns.heatmap(lr_confm, annot=True, fmt='.2f',xticklabels = ["Admitted", "Rejected"] , yticklabels = ["Admitted", "Rejected"] )
         plt.ylabel('True class')
         plt.xlabel('Predicted class')
         plt.title('Logistic Regression')
         plt.show()
```



```
Out[74]: RandomForestClassifier()

In [75]: print('Random Forest Accuracy: {:.3f}'.format(accuracy_score(cy_test, rf.predict(x_test))))
         print('----------------------------------------------------')
         print(classification_report(cy_test, rf.predict(x_test)))

Random Forest Accuracy: 0.950
----------------------------------------------------
              precision    recall  f1-score   support

           0       0.95      0.98      0.96        54
           1       0.96      0.88      0.92        26

    accuracy                           0.95        80
   macro avg       0.95      0.93      0.94        80
weighted avg       0.95      0.95      0.95        80

In [76]: cy = rf.predict(x_test)
         rf_confm = confusion_matrix(cy, cy_test,)
         sns.heatmap(rf_confm, annot=True, fmt='.2f',xticklabels = ["Admitted", "Rejected"] , yticklabels = ["Admitted", "Rejected"] )
         plt.ylabel('True class')
         plt.xlabel('Predicted class')
         plt.title('Random Forest')
         plt.show()
```



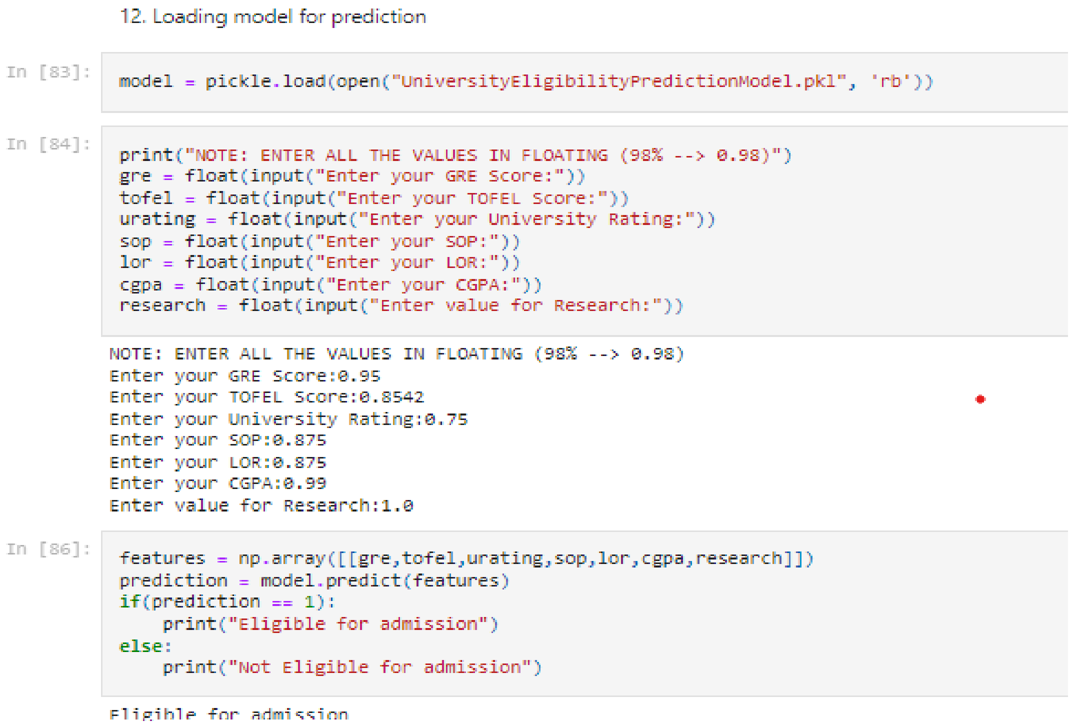
SUPPORT VECTOR MACHINE

```
In [77]: from sklearn.svm import SVC
         svc = SVC()
         svc.fit(x_train, cy_train)

Out[77]: SVC()
```

```
In [78]: print('Support vector machine accuracy: {:.3f}'.format(accuracy_score(cy_test, svc.predict(x_test))))
         print('-----------------------------------------------------')
         print(classification_report(cy_test, svc.predict(x_test)))

Support vector machine accuracy: 0.912
-----------------------------------------------------
              precision    recall  f1-score   support

           0       0.90      0.98      0.94        54
           1       0.95      0.77      0.85        26

    accuracy                           0.91        80
   macro avg       0.93      0.88      0.89        80
weighted avg       0.92      0.91      0.91        80
```

```
In [79]: cy = svc.predict(x_test)
         svc_confm = confusion_matrix(cy, cy_test,)
         sns.heatmap(svc_confm, annot=True, fmt='.2f',xticklabels = ["Admitted", "Rejected"] , yticklabels = ["Admitted", "Rejected"] )
         plt.ylabel('True class')
         plt.xlabel('Predicted class')
         plt.title('Support Vector Machine')
         plt.show()
```

```
In [80]: #Comparing the 3 models used: Linear Regression, Random Forest and SVC.
             #Random Forest model scores higher in accuracy of 95% followed Linear Regression and SVC with accuracy as 91%
```

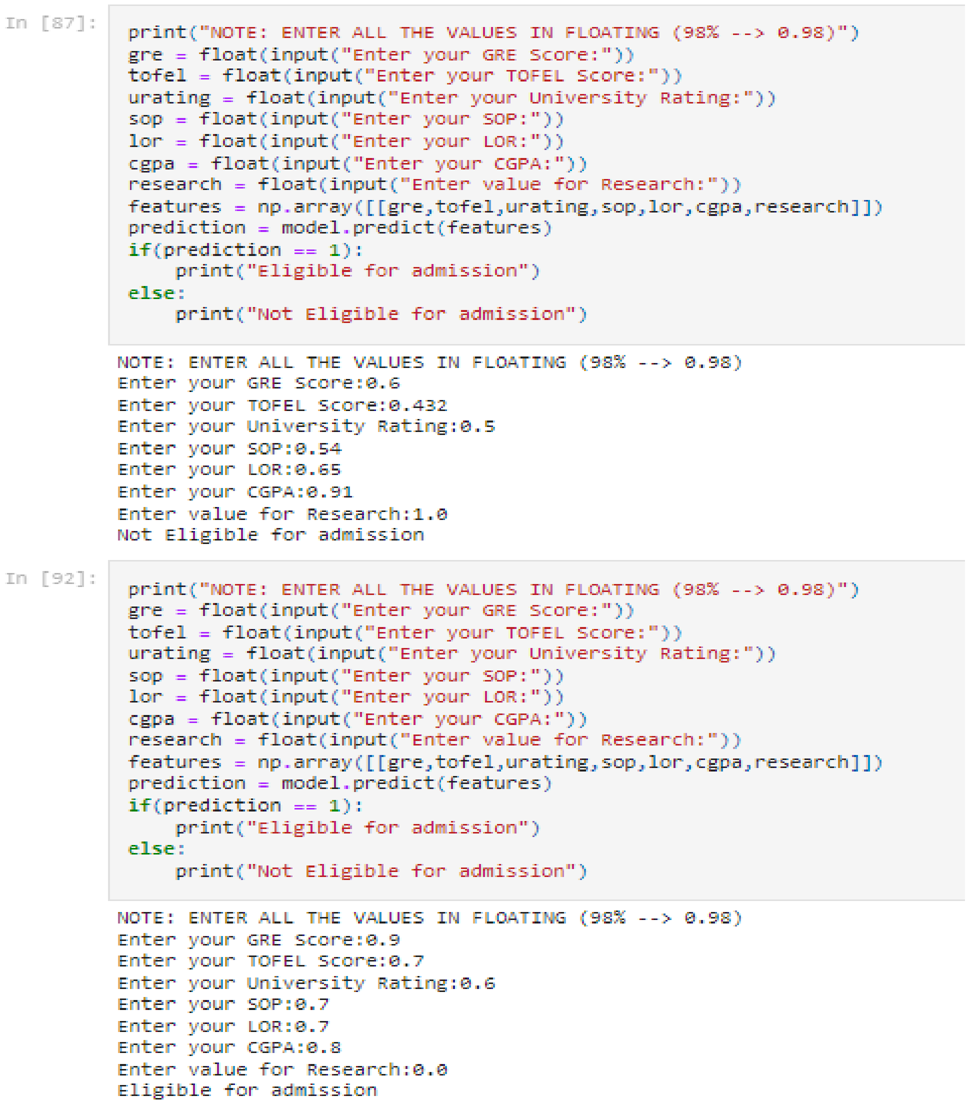


12. Loading model for prediction

```
In [83]: model = pickle.load(open("UniversityEligibilityPredictionModel.pkl", 'rb'))
```

```
In [84]: print("NOTE: ENTER ALL THE VALUES IN FLOATING (98% --> 0.98)")
         gre = float(input("Enter your GRE Score:"))
         tofel = float(input("Enter your TOFEL Score:"))
         urating = float(input("Enter your University Rating:"))
         sop = float(input("Enter your SOP:"))
         lor = float(input("Enter your LOR:"))
         cgpa = float(input("Enter your CGPA:"))
         research = float(input("Enter value for Research:"))

NOTE: ENTER ALL THE VALUES IN FLOATING (98% --> 0.98)
Enter your GRE Score:0.95
Enter your TOFEL Score:0.8542
Enter your University Rating:0.75
Enter your SOP:0.875
Enter your LOR:0.875
Enter your CGPA:0.99
Enter value for Research:1.0
```

```
In [86]: features = np.array([[gre,tofel,urating,sop,lor,cgpa,research]])
         prediction = model.predict(features)
         if(prediction == 1):
             print("Eligible for admission")
         else:
             print("Not Eligible for admission")

Eligible for admission
```



```
In [87]:
print("NOTE: ENTER ALL THE VALUES IN FLOATING (98% --> 0.98)")
gre = float(input("Enter your GRE Score:"))
tofel = float(input("Enter your TOFEL Score:"))
urating = float(input("Enter your University Rating:"))
sop = float(input("Enter your SOP:"))
lor = float(input("Enter your LOR:"))
cgpa = float(input("Enter your CGPA:"))
research = float(input("Enter value for Research:"))
features = np.array([[gre,tofel,urating,sop,lor,cgpa,research]])
prediction = model.predict(features)
if(prediction == 1):
    print("Eligible for admission")
else:
    print("Not Eligible for admission")
```

```
NOTE: ENTER ALL THE VALUES IN FLOATING (98% --> 0.98)
Enter your GRE Score:0.6
Enter your TOFEL Score:0.432
Enter your University Rating:0.5
Enter your SOP:0.54
Enter your LOR:0.65
Enter your CGPA:0.91
Enter value for Research:1.0
Not Eligible for admission
```

```
In [92]:
print("NOTE: ENTER ALL THE VALUES IN FLOATING (98% --> 0.98)")
gre = float(input("Enter your GRE Score:"))
tofel = float(input("Enter your TOFEL Score:"))
urating = float(input("Enter your University Rating:"))
sop = float(input("Enter your SOP:"))
lor = float(input("Enter your LOR:"))
cgpa = float(input("Enter your CGPA:"))
research = float(input("Enter value for Research:"))
features = np.array([[gre,tofel,urating,sop,lor,cgpa,research]])
prediction = model.predict(features)
if(prediction == 1):
    print("Eligible for admission")
else:
    print("Not Eligible for admission")
```

```
NOTE: ENTER ALL THE VALUES IN FLOATING (98% --> 0.98)
Enter your GRE Score:0.9
Enter your TOFEL Score:0.7
Enter your University Rating:0.6
Enter your SOP:0.7
Enter your LOR:0.7
Enter your CGPA:0.8
Enter value for Research:0.0
Eligible for admission
```

GitHub Link:

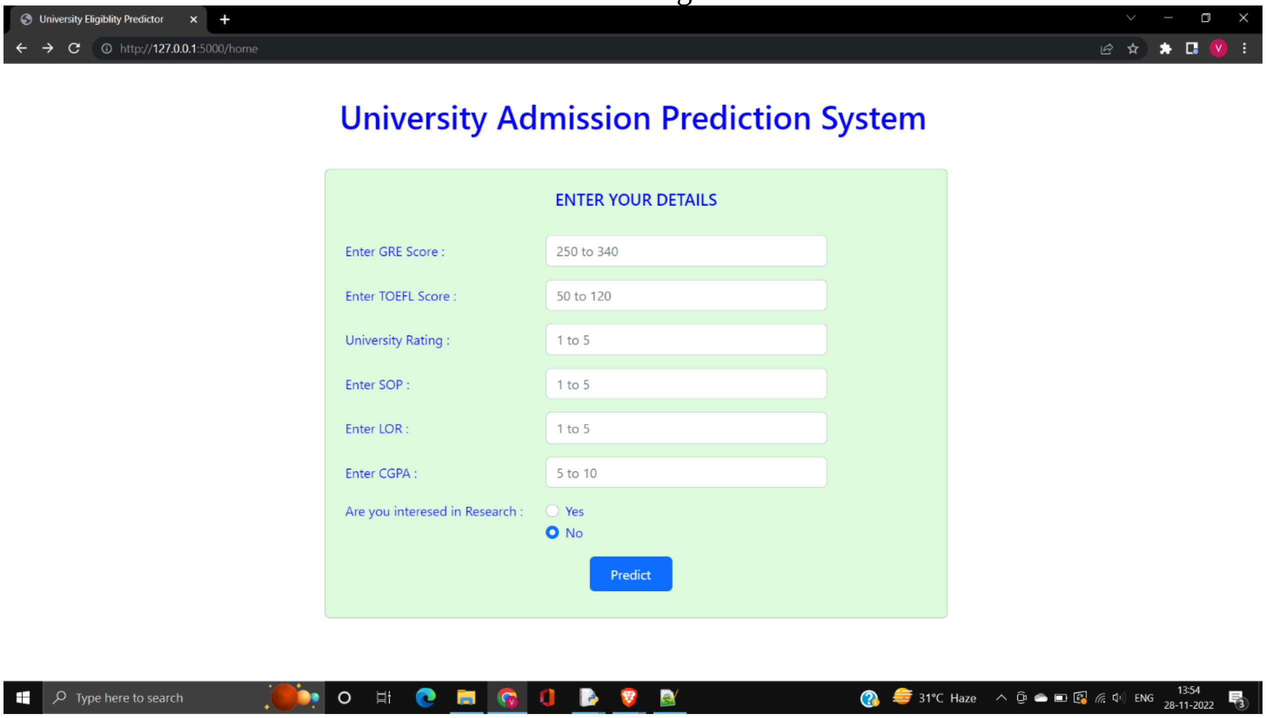
https://github.com/IBM-EPBL/IBM-Project-6693-1658834634

Project Demo Video Link:

https://drive.google.com/drive/folders/1M0sQ_Z845F9Oap2f3oL0hY7KWb2CZ2EF?usp=share_link

Output Screenshots:

Index Page



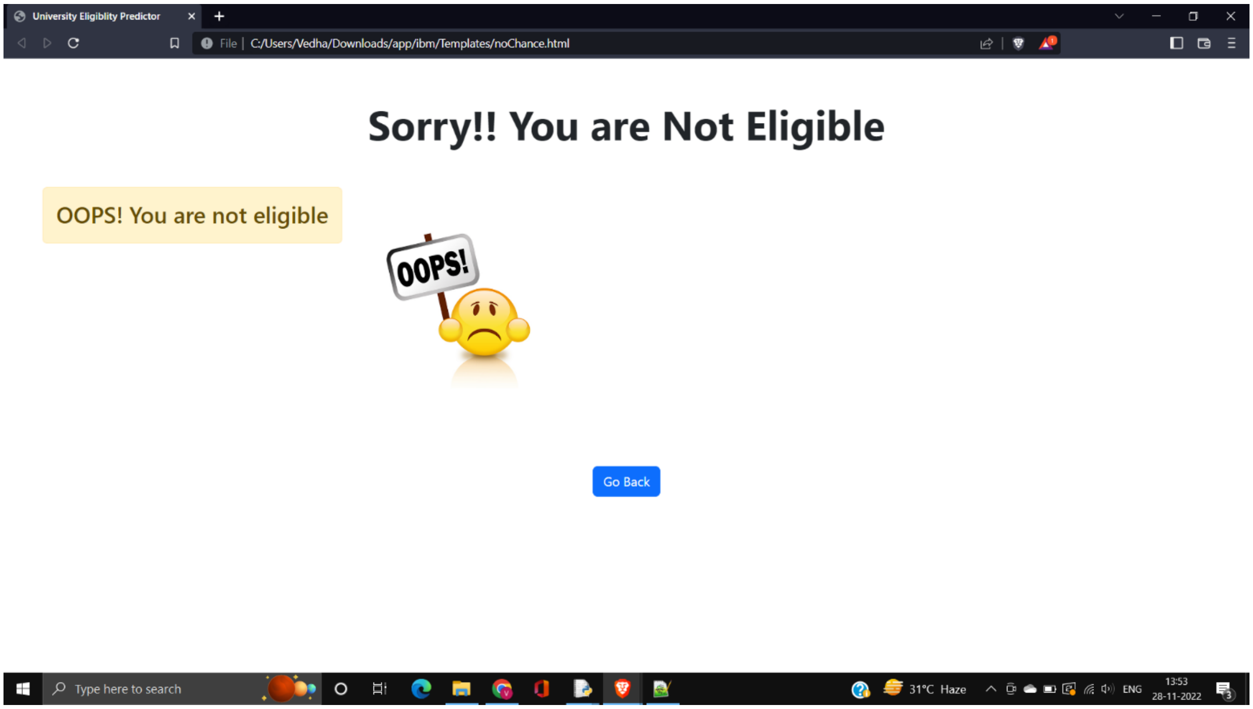
University Admission Prediction System

ENTER YOUR DETAILS

Enter GRE Score :	250 to 340
Enter TOEFL Score :	50 to 120
University Rating :	1 to 5
Enter SOP :	1 to 5
Enter LOR :	1 to 5
Enter CGPA :	5 to 10
Are you interesed in Research :	○ Yes ● No

Predict

NoChance.html



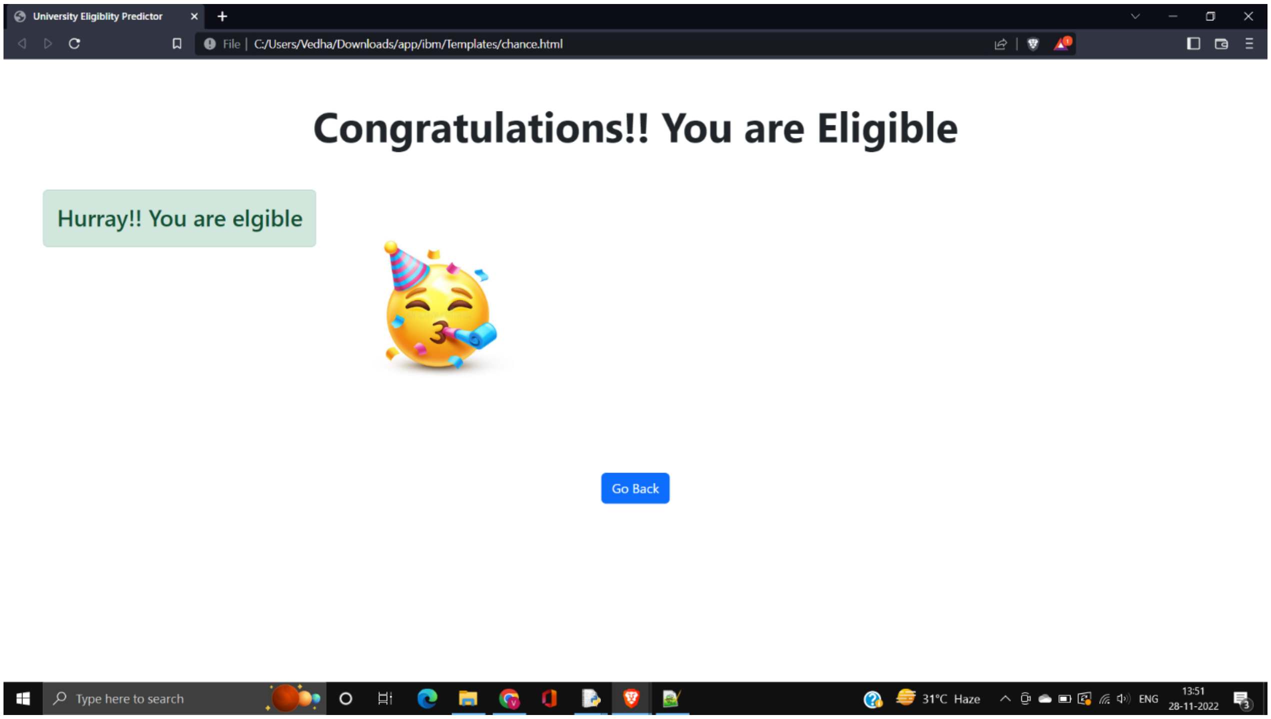
University Eligiblity Predictor
File | C:/Users/Vedha/Downloads/app/ibm/Templates/noChance.html
OOPS!
Type here to search
31°C Haze
ENG
13:53
28-11-2022

Sorry!! You are Not Eligible

OOPS! You are not eligible

Go Back

Chance.html



University Eligiblity Predictor
File | C:/Users/Vedha/Downloads/app/ibm/Templates/chance.html

Congratulations!! You are Eligible

Hurray!! You are elgible

Go Back
Type here to search
31°C Haze
ENG
13:51
28-11-2022