UNIVERSITY ADMIT ELIGIBLITY PREDICTOR
PROJECT REPORT

Submitted by

NAVEEN PANDIAN P	**732119104051**
VIJAYRAGU M	**732119104082**
PRAVEEN KUMAR P M	**732119104055**
MAYILSAMY S	**732119104043**

TEAM ID : PNT2022TMID32168

NANDHA COLLEGE OF TECHNOLOGY

Perundurai Main Road, Erode - 638 052

TABLE OF CONTENTS

1 . INTRODUCTION

1.1 : Project Overview

✓ A Web page is designed for the students where they can check their Admission Eligibility for the respected University.

✓ This Web page can get some of the columns for the students marks that will calculate the students eligibility for that university.

1.2 : Purpose

The purpose of this project is

- ✓ To reduce the work load of the user and also the use of paper.
- ✓ To enable the online Eligibility Checking for the Universities.
- ✓ To reduce the work load of the Students.
- ✓ It will Automatically calculate the chance of the students.

2 . LITERATURE SURVEY

2.1 : Existing Problem

✓ Students who need to check their chances of getting admission in the Universities.

✓ Students can visit the web site and check their chances.

✓ And the Existing Problem can provide the high probability chances for the students who wish to get Admission in the University.

✓ This Existing Problem is well user friendly for the students who can visit this site.

2.2 : Problem Statement

✓ Students need to check their Admission Eligibilities in the respective Universities.

✓ Students can put their mark details that will calculate and provide the probable chances.

✓ Students need this platform to get the idea about the Admissions for the Universities.

3. IDEATION & PROPOSED SOLUTION

3.1 : Empathy Map Canvas

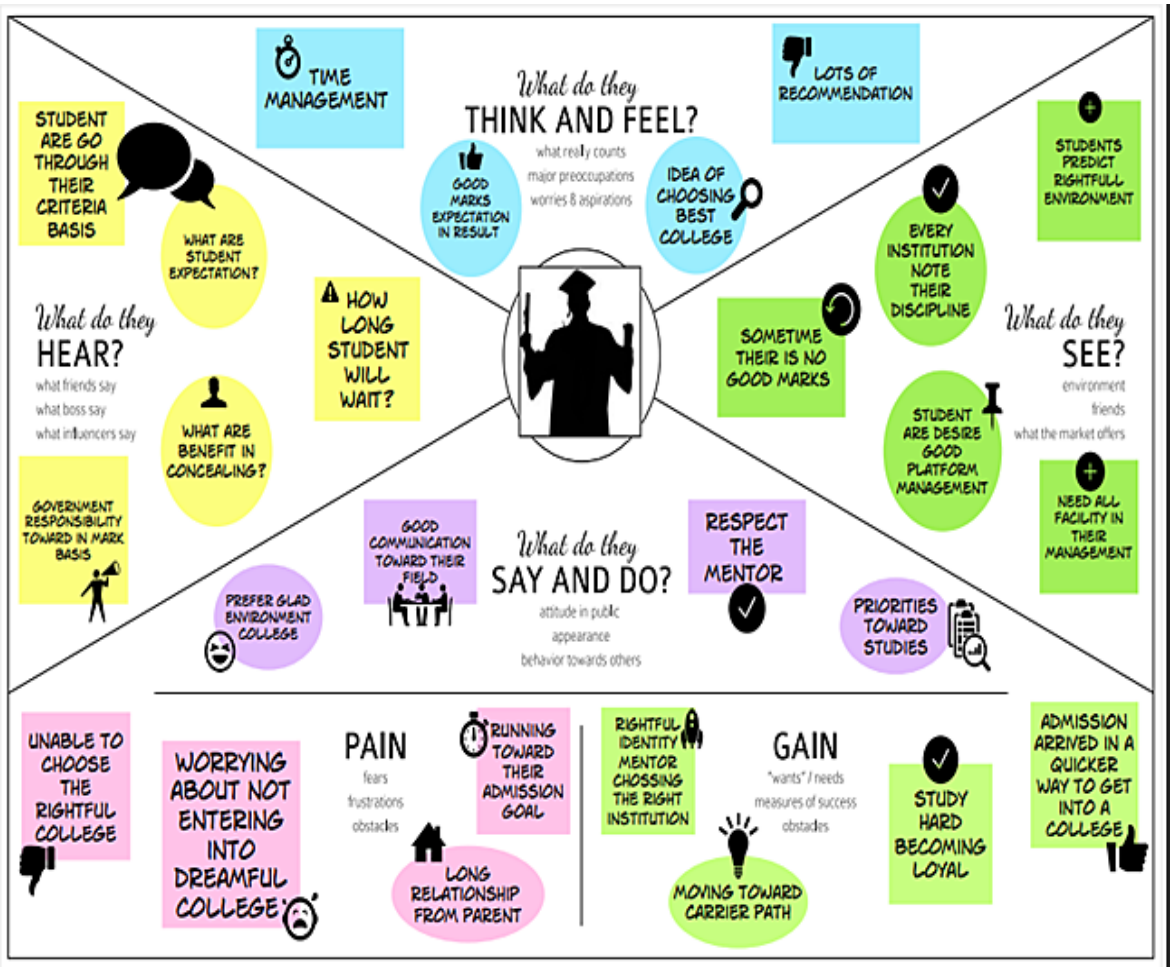

fig 1 : Empathy map (Using canvas)

3.2 : Ideation and Brainstorming

fig 2 : Ideation & brainstorming

3.3 : Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	➢ To Get the student preferred University under the chances of the admission to University.
2.	Idea / Solution description	➢ Prediction calculator. ➢ Free access to every students or users. ➢ Online Supports are provided.
3.	Novelty / Uniqueness	➢ Guide for the online supports. ➢ Easy to understand while we preforming the respected actions.
4.	Social Impact / Customer Satisfaction	➢ Idea of choosing best University. ➢ Admission arrived in a quicker way to get into the University.
5.	Business Model (Revenue Model)	➢ Advertising about the platform. ➢ Able to choose the rightful University that increases the users for using this platform.
6.	Scalability of the Solution	➢ Creating the rightful future. ➢ Maximization of increasing the productivity. ➢ Student are desire good platform management.

3.4 : Problem Solution Fit

fig 3 : problem solution fit

4. REQUIREMENT ANALYSIS

4.1 : Functional requirements

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	• Registration through Form • Registration through Gmail • Registration through LinkedIN
FR-2	User Confirmation	• Confirmation via Email • Confirmation via OTP
FR-3	User Requirements	• Based on the user details the system would scrape the all necessary information. • Upload all the relevant documents.
FR-4	User Details	• GRE score • SOP score • CV • TOEFL score

4.2: Non-Functional requirements

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	➢ Easy to use and User friendly. ➢ The UI would focus on recognize over recall . ➢ Student preferred University under the chances of the admission to University.
NFR-2	**Security**	➢ The Authenticated user would be able to utilize the services of this web site.
NFR-3	**Reliability**	➢ We can access the application at any time . ➢ Guide for the online supports also provided at any time.
NFR-4	**Performance**	➢ Guide for the online supports. ➢ Easy to understand while we preforming the respected actions.
NFR-5	**Availability**	➢ Prediction calculator. ➢ Free access to every students or users. ➢ Online Supports are provided.
NFR-6	**Scalability**	➢ Creating the rightful future. ➢ Maximization of increasing the productivity. ➢ Student are desire good platform management.

5 . PROJECT DESIGN

5.1 : Data Flow Diagrams

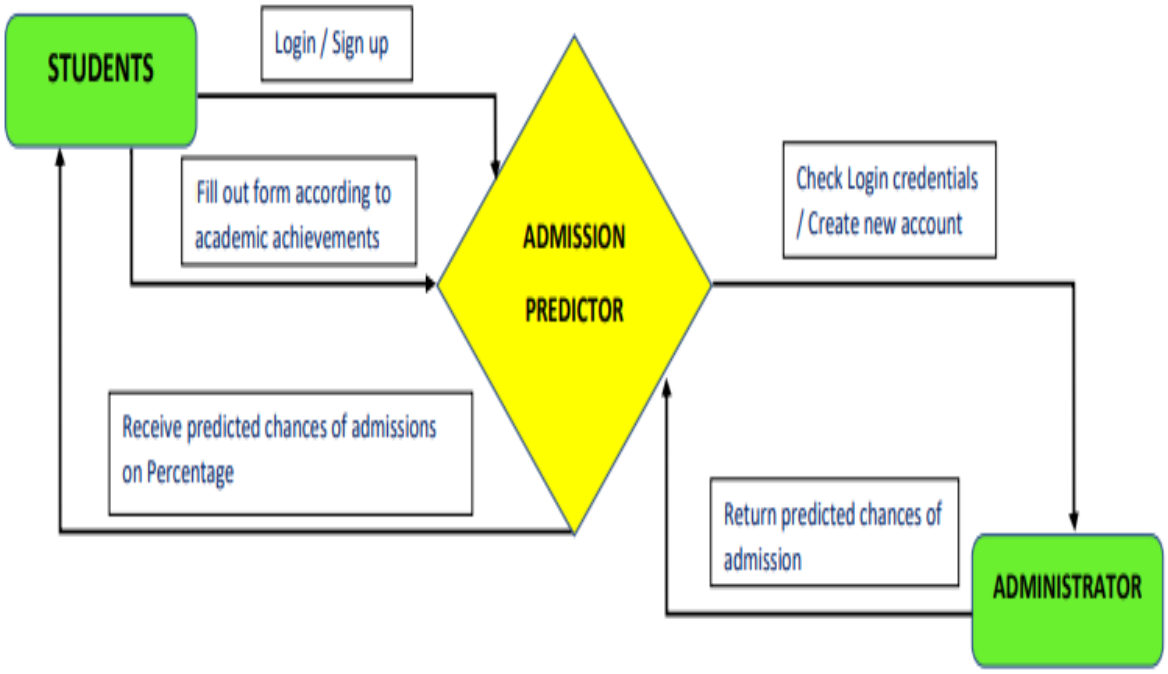

fig 4 : data flow diagram (Level -0)

5.2 : Solution &Technical Architecture

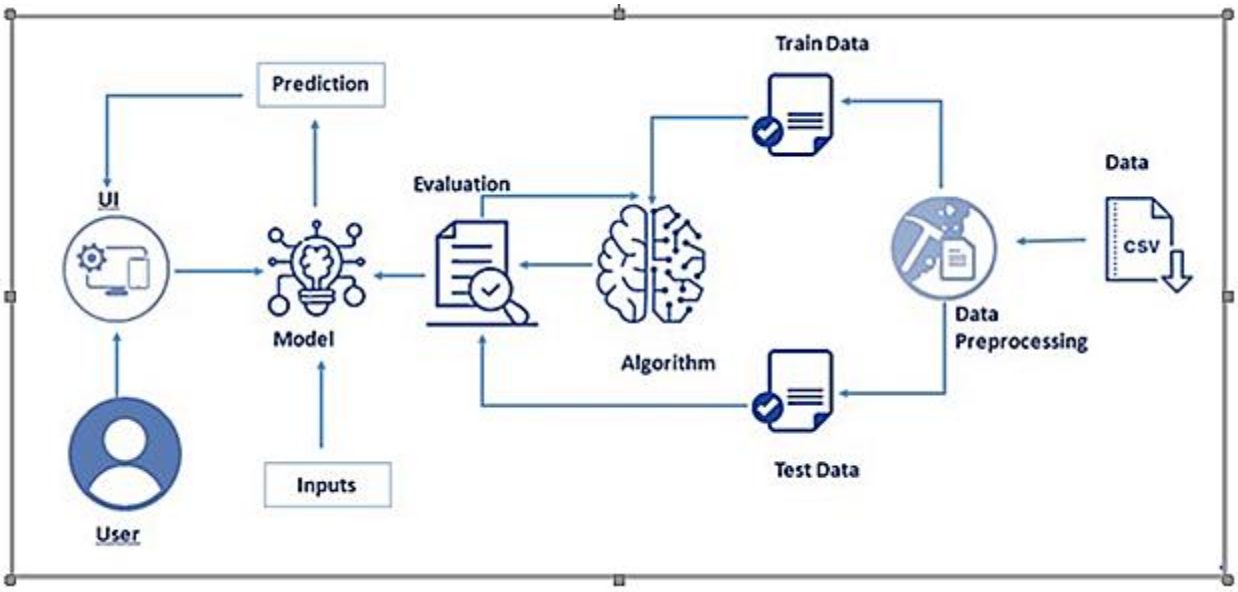

fig 5 : Solution & technical Architecture

5.3 : User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	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	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
	Login	USN-4	As a user, I can register for the application through Gmail	I can access my account	Medium	Sprint-1
	Dashboard	USN-5	As a user, I can log into the application by entering email & password	I can choose colleges	High	Sprint-1
Customer (Web user)	Selection	USN-6	As a user, I can confirm the available college or re-apply to other college	I can select the college	Medium	Sprint-3
	Queries	USN-7	As a user, I can ask queries to the system regarding the help / support or technical issues	I can ask queries	High	Sprint-4
Administrator	Authentication	USN-8	As a admin, I can authenticate the login credentials of user	I can access all the user details	High	Sprint-1
	Dashboard	USN-9	As a admin , I can verify the details of the user	I can confirm the user updating details	High	Sprint-2
	Prediction	USN-10	As a admin , I can train the user details with ML algorithm	I can train the data	High	Sprint-2
	Chances	USN-11	As a admin ,I can solve the queries of users	I can provide chances	High	Sprint-3
	Solution	USN-12	As a admin, I can update the university database depends on the user confirmation	I can solve the queries	High	Sprint-4

6 . PROJECT PLANNING & SCHEDULING

6.1 : Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story points	Priority	Team Members
Sprint-1	Pre requisities	USN-1	Install Anaconda, install python packages, prior knowledge	4	Low	PRAVEEN KUMAR P M, MAYILSAMY S
Sprint-1	Pre requisities	USN-2	Project flow, project objectives, project structure	4	Low	PRAVEEN KUMAR P M, MAYILSAMY S
Sprint-1	Data collection	USN-3	Download the dataset, importing The Libraries, Reading The Dataset	4	Medium	VIJAYRAGU M
Sprint-1	Data collection	USN-4	Analyze The Data, Handling Missing Values	4	Medium	NAVEEN PANDIAN P
Sprint-1	Data collection	USN-5	Data Visualization	4	High	PRAVEEN KUMAR P M, MAYILSAMY S
Sprint-2	Data collection	USN-6	Splitting Dependent And Independent Columns, Splitting the Data into train and test	5	Medium	NAVEEN PANDIAN P

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story points	Priority	Team Members
Sprint-2	Model Building	USN-7	Training and Testing The Model	5	Medium	VIJAYRAGU M
Sprint-2	Model Building	USN-8	Model Evaluation	5	High	PRAVEEN KUMAR P M
Sprint-2	Model Building	USN-9	Save the model	5	Low	MAYILSAMY S
Sprint-3	Application Building	USN-10	Build HTML Code	10	High	VIJAYRAGU M, MAYILSAMY S
Sprint-3	Application Building	USN-11	Build HTML Code	10	High	NAVEEN PANDIAN P, PRAVEEN KUMAR P M
Sprint-4	Train the Model on IBM	USN-12	Register for IBM Cloud	4	Medium	MAYILSAMY S, PRAVEEN KUMAR P M
Sprint-4	Train the Model on IBM	USN-13	Train Machine Learning Model on IBM Watson	8	High	NAVEEN PANDIAN P
Sprint-4	Train the Model on IBM	USN-14	Integrate Flask with scoring endpoint	8	High	VIJAYRAGU M

6.2 : Sprint Delivery Schedule

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	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

Velocity:

Velocity: We have 6-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day).

$$AV = 20/6 = 3.33$$

6.3 : Reports From JIRA

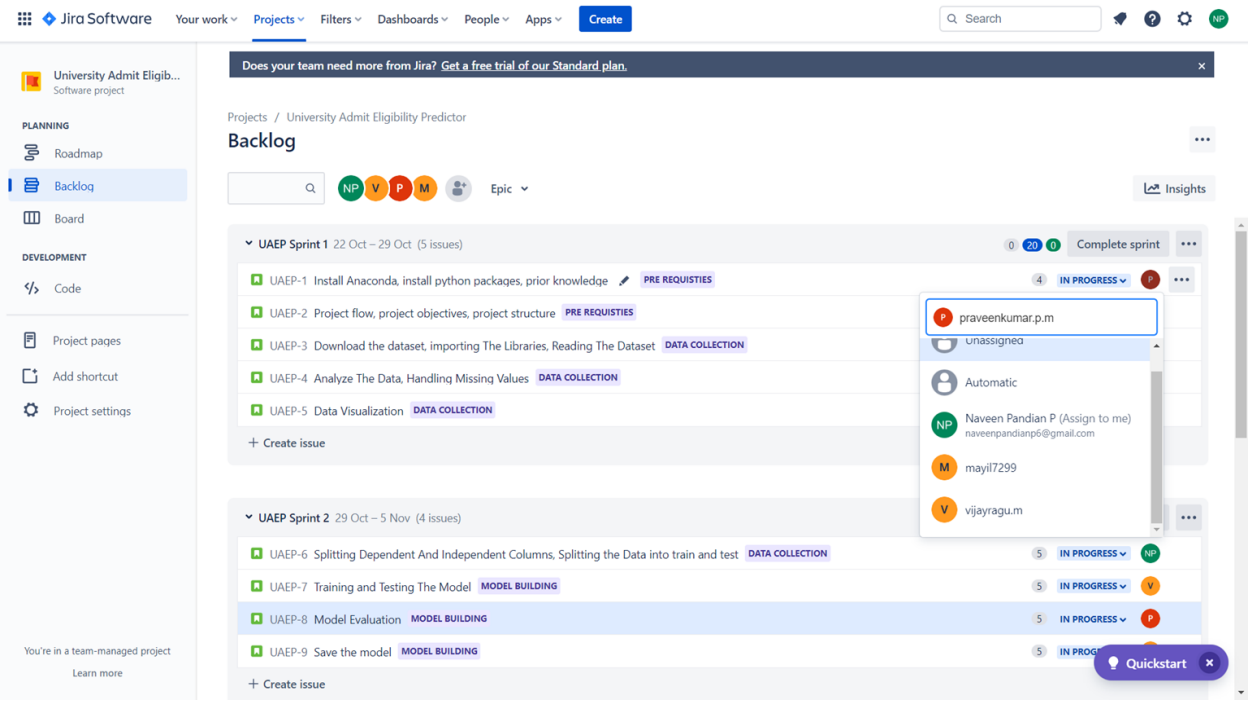

fig 6 : Assign task to team

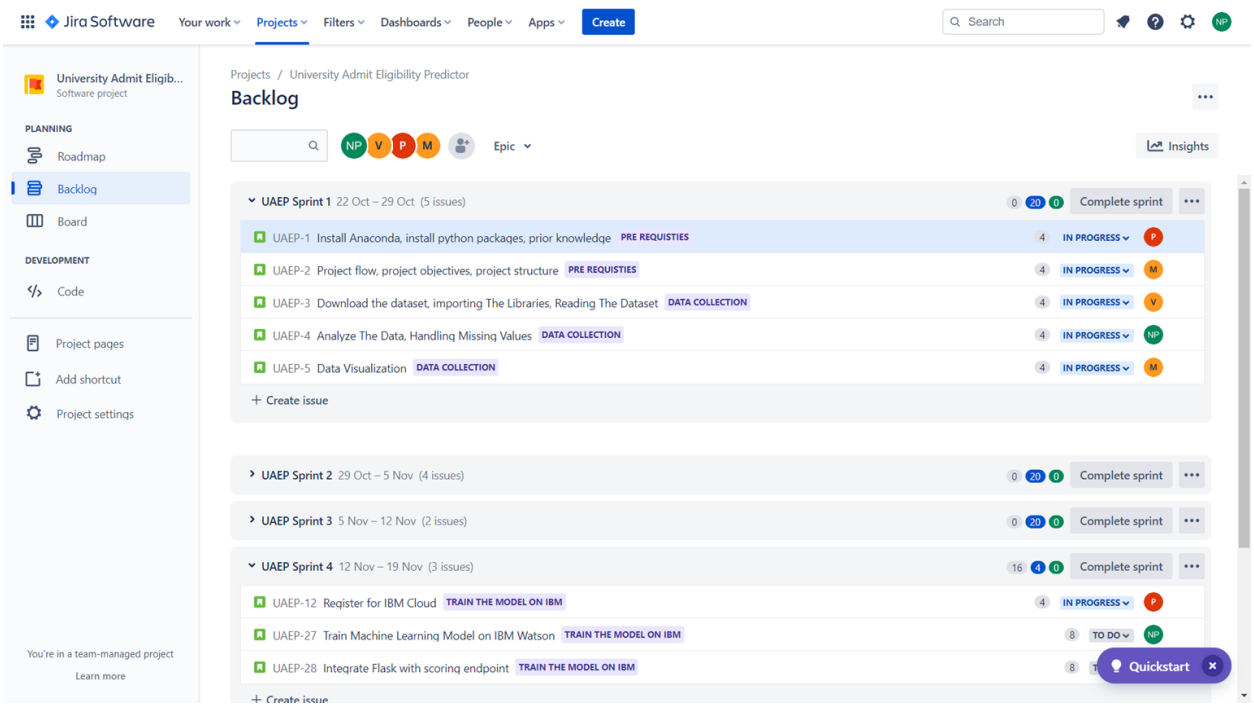

fig 7 : Backlog

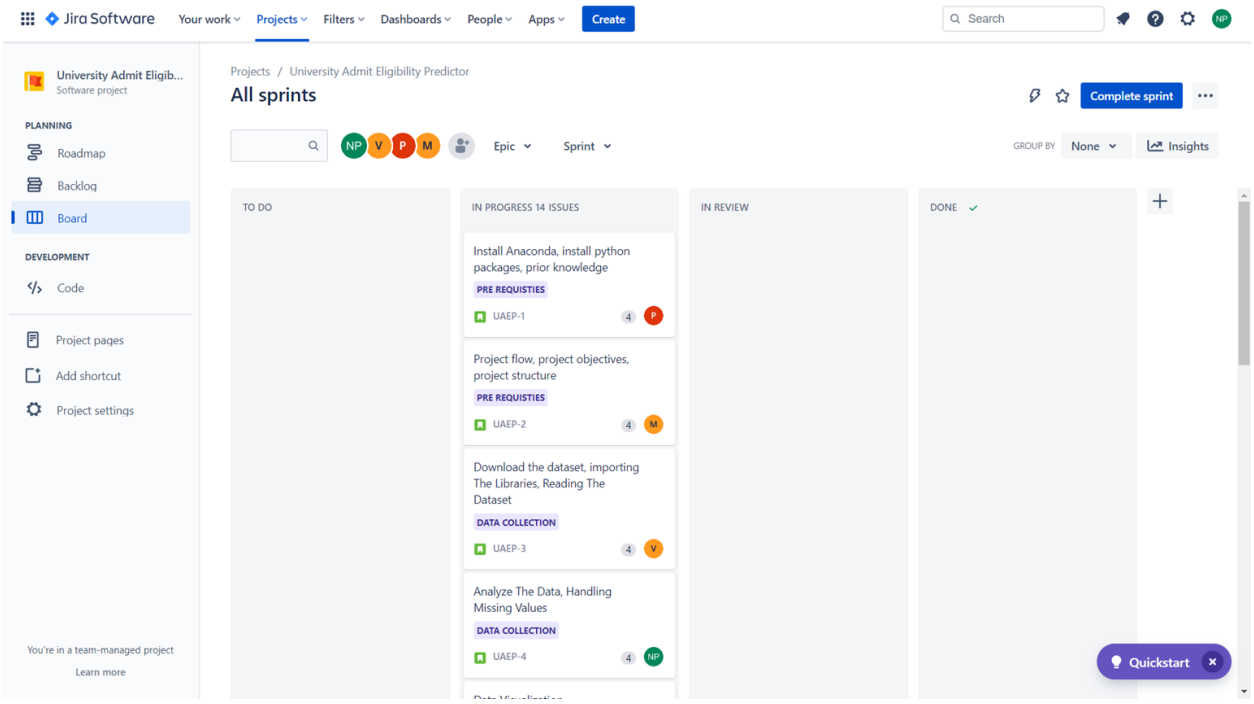

fig 8 : board

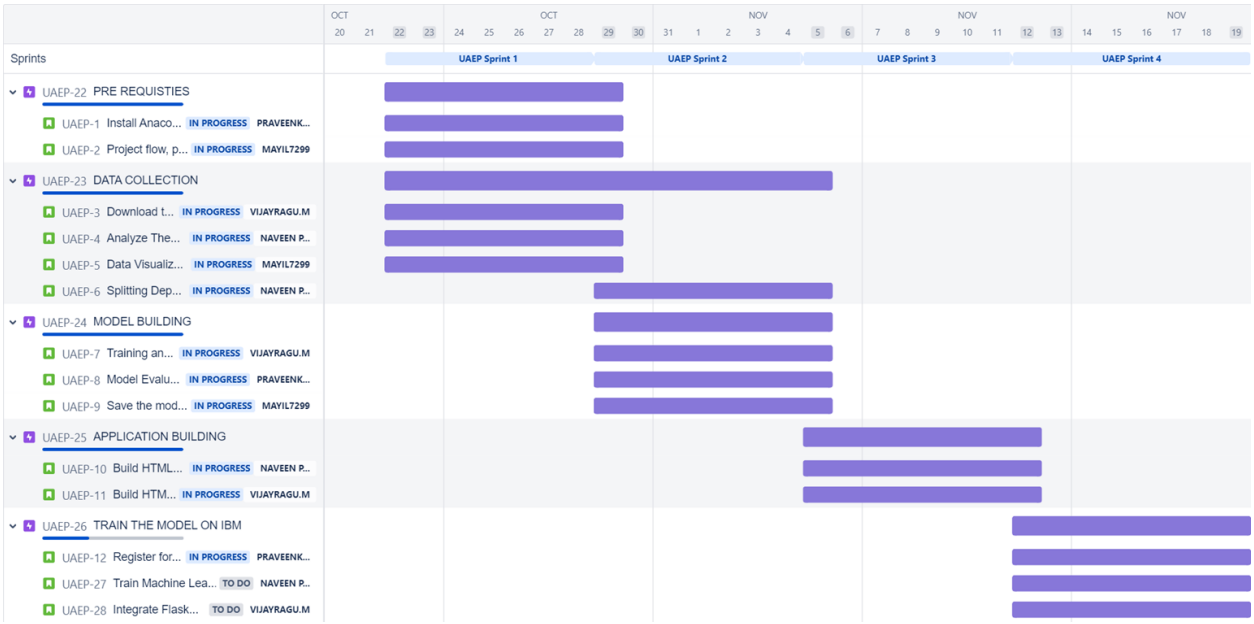

fig 9 : RoadMap

7 . CODING & SOLUTIONING

7.1 Features :

• Dataset :

	A	B	C	D	E	F	G	H	I	J
1	Serial No.	GRE Score	TOEFL Sco	University	SOP	LOR	CGPA	Research	Chance of Admit	
2	1	337	118	4	4.5	4.5	9.65	1	0.92	
3	2	324	107	4	4	4.5	8.87	1	0.76	
4	3	316	104	3	3	3.5	8	1	0.72	
5	4	322	110	3	3.5	2.5	8.67	1	0.8	
6	5	314	103	2	2	3	8.21	0	0.65	
7	6	330	115	5	4.5	3	9.34	1	0.9	
8	7	321	109	3	3	4	8.2	1	0.75	
9	8	308	101	2	3	4	7.9	0	0.68	
10	9	302	102	1	2	1.5	8	0	0.5	
11	10	323	108	3	3.5	3	8.6	0	0.45	
12	11	325	106	3	3.5	4	8.4	1	0.52	
13	12	327	111	4	4	4.5	9	1	0.84	
14	13	328	112	4	4	4.5	9.1	1	0.78	
15	14	307	109	3	4	3	8	1	0.62	
16	15	311	104	3	3.5	2	8.2	1	0.61	

fig 10 : dataset (Admission_Predict.csv file)

• Notebook deployed on IBM cloud : (python code)

```
import OS, types
import pandas as pd
from botocore.client import Config
import ibm_boto3
def __iter__(self): return 0
# @hidden_cell
```

 # The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.

 # You might want to remove those credentials before you share the notebook.

```
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='1kgosk9GTY9RrReZ4_BkW84FQf40X32_qfP1bHOUJo9o',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')


bucket = 'universityadmiteligibilitypredict-donotdelete-pr-teafgkiepxpvip'
object_key = 'Admission_Predict.csv'


body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )


data = pd.read_csv(body)
data.head()
```

• **Importing The Libraries**

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

• **Analyze The Data**

 # There is no need for Serial No. so we "Drop" the " Serial NO. " column.....

```
data.drop(["Serial No."],axis=1,inplace=True)
data.head()
data.describe()              # shows the descriptive statistics....
data.info()
```

• **Handling Missing Values**

```
data.isnull().any()     # to check any null
data.isnull().sum()       # to check the count of null
```

 # there is no missing values so we go for next step.....

• **Data Visualization**

(i) Univarient Analysis

```
# Univarient Analysis
sns.histplot(x=data["GRE Score"],color='Red')          #Choosing the GRE Score
```

(ii) Bi-Variate Analysis

```
#bivariate analysis
sns.barplot(x=data["SOP"],y=data["CGPA"])    # Choosing SOP and CGPA column
```

(iii) Multi-Variate Analysis

```
# Multi-Variate Analysis
sns.pairplot(data)
```

• **Find the outliers and replace them outliers**

```
sns.boxplot(data["GRE Score"])
sns.boxplot(data["TOEFL Score"])
sns.boxplot(data["University Rating"])
sns.boxplot(data.SOP)
data['LOR '].median()
q1=data['LOR '].quantile(0.25)  #(Q1)          # Replacement using median....
q3=data['LOR '].quantile(0.75)  #(Q3)
IQR=q3-q1
upper_limit= q3 + 1.5*IQR
lower_limit= q1 - 1.5*IQR
data['LOR ']= np.where(data['LOR ']<lower_limit,3.5,data['LOR '])
sns.boxplot(x=data['LOR '],showfliers=False)
sns.boxplot(data['CGPA'])
data['CGPA'].median()                          # median....
q1=data['CGPA'].quantile(0.25)  #(Q1)          # Replacement using median....
q3=data['CGPA'].quantile(0.75)  #(Q3)
IQR=q3-q1
upper_limit= q3 + 1.5*IQR
lower_limit= q1 - 1.5*IQR
data['CGPA']= np.where(data['CGPA']<lower_limit,8.61,data['CGPA'])
sns.boxplot(x=data['CGPA'],showfliers=False)
sns.boxplot(data['Research'])
sns.boxplot(data['Chance of Admit '])
data['Chance of Admit '].median()                    # median....
```

```
q1=data['Chance of Admit '].quantile(0.25)  #(Q1)          # Replacement using median....

q3=data['Chance of Admit '].quantile(0.75)  #(Q3)

IQR=q3-q1

upper_limit= q3 + 1.5*IQR

lower_limit= q1 - 1.5*IQR

data['Chance of Admit ']= np.where(data['Chance of Admit
']<lower_limit,0.73,data['Chance of Admit '])

sns.boxplot(x=data['Chance of Admit '],showfliers=False)

# Finally we Replaced all the outliers using median.....
```

• Split the data into dependent and independent columns

(i) Independent variable

```
x=data.iloc[:,0:7].values

x
```

(ii) Dependent variable

```
y=data.iloc[:,7:].values

y

x.shape

y.shape
```

• **Splitting the Data into train and test**

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=10)

y_train=(y_train>0.5)

y_train

y_test=(y_test>0.5)

y_test
```

if you observe y column it has the probability of stutent getting admitted into a university.

I would like to convert this probability as either yes or no ..... (so the logic y_train and y_test as above)

• **Training and Testing The Model**

```
from sklearn.linear_model import LogisticRegression

cls =LogisticRegression(random_state =0)

lr=cls.fit(x_train, y_train)

y_pred =lr.predict(x_test)

y_pred
```

• **Model Evaluation**

```
from sklearn.metrics import
accuracy_score,recall_score,roc_auc_score,confusion_matrix

print("\nAccuracy score: %f" %(accuracy_score(y_test,y_pred)*100))

print("Recall score : %f" %(recall_score(y_test,y_pred)*100))

print("ROC score : %f\n" %(roc_auc_score(y_test,y_pred)*100))

print(confusion_matrix(y_test,y_pred))
```

```
Accuracy score: 92.500000
Recall score : 98.666667
ROC score : 49.333333

[[ 0  5]
 [ 1 74]]
```

• **IBM Deployment**

```
!pip install -U ibm-watson-machine-learning

from ibm_watson_machine_learning import APIClient

import json
```

Authenticate and Set Space

```
wml_credentials = {

   "apikey":"ZpX-YXfeK9dPqp41dYk1ly1faNv4XM2XfYomx8pIcdZW",

   "url":"https://us-south.ml.cloud.ibm.com"

}
```

URLS Dallas: https://us-south.ml.cloud.ibm.com, London - https://eu-gb.ml.cloud.ibm.com, Frankfurt - https://eu-de.ml.cloud.ibm.com, Tokyo - https://jp-tok.ml.cloud.ibm.com

```
wml_client = APIClient(wml_credentials)
wml_client.spaces.list()
SPACE_ID= "e51e0c2e-ab07-4522-b64a-824811f7ed38"
wml_client.set.default_space(SPACE_ID)       ##   SUCCESS
wml_client.software_specifications.list(500)
```

• Save and Deploy the model

```
import sklearn
sklearn.__version__
MODEL_NAME = 'University_Admit_Eligibility_Predictor'
DEPLOYMENT_NAME = 'University_Admit_Eligibility_Predictor'
DEMO_MODEL = lr
# Set Python Version
software_spec_uid = wml_client.software_specifications.get_id_by_name('runtime-22.1-py3.9')
# Setup model meta
model_props = {
   wml_client.repository.ModelMetaNames.NAME: MODEL_NAME,
   wml_client.repository.ModelMetaNames.TYPE: 'scikit-learn_1.0',
   wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
}
```

```
#Save model
model_details = wml_client.repository.store_model (
    model=DEMO_MODEL,
    meta_props=model_props,
    training_data=x_train,
    training_target=y_train
)
model_details
model_id = wml_client.repository.get_model_id(model_details)
model_id

# Set meta
deployment_props = {
    wml_client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
    wml_client.deployments.ConfigurationMetaNames.ONLINE: {}
}
```

```
# Deploy

deployment = wml_client.deployments.create(

   artifact_uid=model_id,

   meta_props=deployment_props

)


#######################################################################################

Synchronous deployment creation for uid: 'be651858-f626-4a89-bfc5-19881938c2c1' started

#######################################################################################


initializing
Note: online_url is deprecated and will be removed in a future release. Use serving_urls instead.

ready


------------------------------------------------------------------------------------------------
Successfully finished deployment creation, deployment_uid='16fab11a-5a0c-4180-8216-d760800e43d3'
------------------------------------------------------------------------------------------------
```

• html files

(index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
   <style>
      body{
         background:url("../static/img/img1.jpg") no-repeat center fixed;
         background-size: cover;
         }
      .t{
         color:Darkblue;
         text-align:center;
      }
   </style>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>University Admit Eligibility Predictor</title>
</head>
<body>
   <h1 class="t"> UNIVERSITY ADMISSION PREDICTION SYSTEM </h1>

   <h2> Enter your details and get probability your admission </h2>
<form action='/y_predict', method='post'>
Enter GRE Score: <input type="text" name="GRE" placeholder="GRE" required="required" />
<br><br>
Enter TOEFL Score: <input type="text" name="TOEFL" placeholder="TOEFL"
required="required" />
<br><br>
University Rating :
<br>
<input type="radio" id="1" name="no" value="1"/><label>1</label><br>
<input type="radio" id="2" name="no" value="2"/><label>2</label><br>
<input type="radio" id="3" name="no" value="3"/><label>3</label><br>
<input type="radio" id="4" name="no" value="4"/><label>4</label><br>
<input type="radio" id="5" name="no" value="5"/><label>5</label><br>
<br>
Enter SOP: <input type="text" name="SOP" placeholder="SOP" required="required"  />
<br><br>
 Enter LOR: <input type="text" name="LOR" placeholder="LOR" required="required"  />
<br><br>
 Enter CGPA: <input type="text" name="CGPA" placeholder="CGPA" required="required"  />
<br><br>
```

```
 Research:
<br>
<input type="radio" id="1" name="res" value="1"/><label>Research</label><br>
<input type="radio" id="0" name="res" value="0"/><label>No Research</label><br>
<br>
<input type="submit" value="Predict"/>

</form>

</body>
</html>
```

(chance.html)

```
<html>
   <head>
      <style>
         body{
            background-image:url("../static/img/img2.jpg");
            background-repeat:no-repeat;
            background-attachment:fixed;
            background-position:center;
         }
         h1,h3{
            text-align:center;
         }
      </style>
      <title> chance </title>
   </head>




   <body>
      <h1> Predicting Chance of Admission </h1>
      <h3> Prediction : <b><u>You have a chance</u> !!...</b> <img
src="../static/img/img3.jpg" height="40px" width="40px" /></h3>
      <h3><a href="/">Go Back..</a></h3>

   </body>
</html>
```

(noChance.html)

```
<html>
   <head>
      <style>
         body{
            background-image:url("../static/img/img5.jpg");
            background-repeat:no-repeat;
            background-size:500px 500px;
            background-attachment:fixed;
            background-position:center;
         }
         h1,h3{
            text-align:center;
         }
      </style>
      <title> chance </title>
   </head>
   <body>
      <h1> Predicting Chance of Admission </h1>
      <h3> Prediction : <b><u>You don't have a chance</u> !!...</b> <img
src="../static/img/img4.jpg" height="40px" width="40px" /></h3>
      <h3><a href="/">Go Back..</a></h3>
   </body>
</html>
```

• app.py (run with the help of flask)

```
import requests
import numpy as np
from flask import Flask, redirect, url_for, render_template, request
```

```
# NOTE: you must manually set API_KEY below using information retrieved from your IBM
Cloud account.
API_KEY = "ZpX-YXfeK9dPqp41dYk1ly1faNv4XM2XfYomx8pIcdZW"
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__)

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


@app.route("/y_predict", methods=['POST'])
def y_predict():
    int_features=[float(x) for x in request.form.values()]
    final_features=[list(int_features)]

    # NOTE: manually define and pass the array(s) of values to be scored in the next line
    payload_scoring = {"input_data": [{"fields": [int_features], "values": final_features}]}
    response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/16fab11a-5a0c-4180-8216-
d760800e43d3/predictions?version=2022-11-08', json=payload_scoring,
headers={'Authorization': 'Bearer ' + mltoken})
    print(response_scoring)
    predictions=response_scoring.json()
    predict = predictions['predictions'][0]['values'][0][0]
    print("Final prediction :",predict)


    # showing the prediction results in a UI# showing the prediction results in a UI
    if predict==True:
        return render_template("chance.html")
    else:
        return render_template("noChance.html")

if __name__=='__main__':
    app.run()
```

fig 11 : home page (index.html)

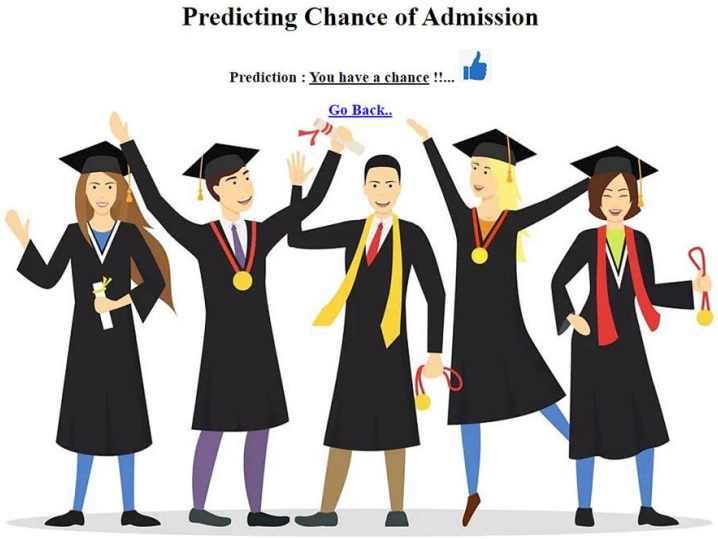

fig 12 : change page (change.html)

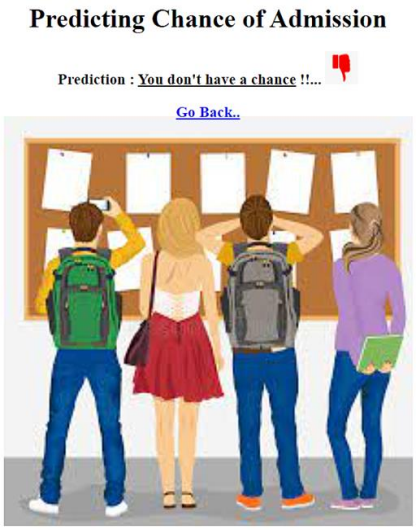

fig 13 : noChance page (noChance.html)

8.TESTING

8.1 Test Cases :

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
UAEP_TC_001	Functional	Home Page	Verify user is able to see the submitted data when user click on predict button	Flask , vscode	1. Click on the input text box 2. Fill the box with the required data 3. Click the predict button	home.html	home page get display	Working as expected	Pass	Steps are clear to follow	y	null	Naveen Pandian
UAEP_TC_002	Functional	Home Page	Verify the UI elements in home page	vscode	1. Click on the input text box 2. Fill the box with the required data 3. Click the predict button	home.html	retrieve to prediction result	Working as expected	Pass	Steps are clear to follow	y	null	Vijayragu
UAEP_TC_003	Functional	chance Page	Verify the UI elements in chance page	Flask , vscode	1. See your UAEP prediction result 2. Also you can see the back to home link 3. Click the link to back to home	chance.html	predict the chances	Working as expected	pass	Steps are clear to follow	y	null	Praveen Kumar
UAEP_TC_004	Functional	chance Page	Verify user is able to go back to home	Flask , vscode	1. See your UAEP prediction result 2. Also you can see the back to home link 3. Click the link to back to home	chance.html	retrieve to home page	Working as expected	pass	Steps are clear to follow	y	null	Mayil Samy
UAEP_TC_005	Functional	noChance Page	Verify the UI elements in noChance page	Flask , vscode	1. See your UAEP prediction result 2. Also you can see the back to home link 3. Click the link to back to home	noChance.html	predict the chances	Working as expected	pass	Steps are clear to follow	y	null	Naveen Pandian
UAEP_TC_006	Functional	noChance Page	Verify user is able to go back to home	Flask , vscode	1. See your UAEP prediction result 2. Also you can see the back to home link 3. Click the link to back to home	nochance.html	retrieve to home page	Working as expected	pass	Steps are clear to follow	y	null	Vijayragu

fig 14 : test case report

8.2 User Acceptance Testing :

1. Purpose of Document

The purpose of document is how to designed the model of test-case and running clear set of document in correct format through the user acceptance testing(UAT)

2. Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	8	2	1	2	13r
Duplicate	1	0	3	0	4
External	2	2	0	2	6
Fixed	8	1	4	16	35
Not Reproduced	0	0	1	0	1

Skipped	0	1	0	1	3
Won't Fix	0	4	1	1	6
Totals	19	10	10	22	68

3. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Handling missing values	6	0	0	6
Client Application	49	0	0	49
testing the model	3	0	0	3
Tunning the model	4	0	0	4
Exception Reporting	8	0	0	8
Final Report Output	2	0	0	2
Control version	2	0	0	2

9. RESULTS

9.1 Performance Metrics :

S.No.	Parameter	Values
1.	Metrics	**Regression Model:** Logistic Regression **Classification Model:** 1. Confusion Matrix - [[0 5] [1 74]] 2. Accuracy Score - 92.50000 3. Recall Score – 98.666667 4. ROC Score – 49.333333 **Classification Report :** precision – 0.88 recall - 0.93 f1-score - 0.90 support - 80
2.	Tune The Model	Hyperparameter Tuning: (GridSearchCV) clf.best_score_ - 0.921875 Validation Method – GridSearchCV(e stimator=SVC())

• Testing the Model : Logistic Regression

In [143]:
```
from sklearn.linear_model import LogisticRegression
cls =LogisticRegression(random_state =0)
```

In [144]:
```
lr=cls.fit(x_train, y_train)
```

```
D:\Anaconda\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d ar
ray was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
D:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [145]:
```
y_pred =lr.predict(x_test)
```

Out[145]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
       False,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True])
```

fig 15 : Regression Model

```
from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matrix
print("\nAccuracy score: %f" %(accuracy_score(y_test,y_pred)*100))
print("Recall score : %f" %(recall_score(y_test,y_pred)*100))
print("ROC score : %f\n" %(roc_auc_score(y_test,y_pred)*100))
print(confusion_matrix(y_test,y_pred))
```

```
Accuracy score: 92.500000
Recall score : 98.666667
ROC score : 49.333333

[[ 0  5]
 [ 1 74]]
```

fig 16 : Confusion matrix

```
from sklearn.svm import SVC
model = SVC()
```

```
model.fit(x_train,y_train)
```

```
D:\Anaconda\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d ar
ray was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
SVC()
```

```
y_pred = model.predict(x_test)
y_pred1=model.predict(x_train)
```

Hyper parameter Tuning

GridsearchCV

```
from sklearn.model_selection import GridSearchCV
```

```
parameters = {'kernel':['linear','rbf'],
              'C':[0.1,0.5,1.0],
              'gamma':['scale', 'auto']

}
```

```
clf=GridSearchCV(SVC(),param_grid=parameters,verbose=2)
```

```
clf.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
[CV] END ..................C=0.1, gamma=scale, kernel=linear; total time=   0.0s
[CV] END ..................C=0.1, gamma=scale, kernel=linear; total time=   0.0s
[CV] END ..................C=0.1, gamma=scale, kernel=linear; total time=   0.0s
```

fig 17 : Hyperparameter Tunning

10. CONCLUSION

This system , being the first we have created in PHP , has proven more difficult than originally imagined. While it may sound simple to fill out a few forms and process the information, much more is involved in the selection of applicants than this. Every time progress was made and features were added, ideas for additional features or methods to improve the usability of the system made themselves apparent. Further more, adding one feature meant that another required feature was now possible, and balancing completing these required features with the ideas for improvement as well as remembering everything that had to be done was a project in itself. Debugging can sometimes be a relatively straight forward process, or rather finding out what you must debug can be . Since so many parts of the admissions system are integrated into one another , if an error occurs on one page, it may be a display error , for example; it may be the information is not correctly read from the database; or even that the information is not correctly stored in the database initially, and all three must be checked on each occasion. Thisslows down the process and can be frustrating if the apparent cause of aproblemis not obvious at first . Language used must be simple and easy to understand and compatibility is paramount. If this system were not designed as anentirely web based application, it would not have been possible to recreate its current state of portability .Overall, the system performs well, and while it does not include all of the features that may have been desired, it lives up to initial expectations. The majority of features that are included work flawlessly and the errors that do exist are minor or graphical.

11. FUTURE SCOPE

The future scope of this project is very broad. Few of them are:

- This can be implemented in less time for proper admission process.

- This can be accessed anytime anywhere, since it is a web application provided only an internet connection.

- The user had not need to travel a long distance for the admission and his/her time is also saved as a result of this automated system.

12. APPENDIX

Source Code :

- **Notebook deployed on IBM cloud : (python code)**

```
import OS, types

import pandas as pd

from botocore.client import Config

import ibm_boto3

def __iter__(self): return 0

# @hidden_cell

# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.

# You might want to remove those credentials before you share the notebook.

cos_client = ibm_boto3.client(service_name='s3',

    ibm_api_key_id='1kgosk9GTY9RrReZ4_BkW84FQf40X32_qfP1bHOUJo9o',

    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",

    config=Config(signature_version='oauth'),

    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')


bucket = 'universityadmiteligibilitypredict-donotdelete-pr-teafgkiepxpvip'

object_key = 'Admission_Predict.csv'


body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']

# add missing __iter__ method, so pandas accepts body as file-like object

if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )
```

```
data = pd.read_csv(body)
data.head()
```

• Importing The Libraries

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

• Analyze The Data

```
# There is no need for Serial No. so we "Drop" the " Serial NO. " column.....
data.drop(["Serial No."],axis=1,inplace=True)
data.head()
data.describe()              # shows the descriptive statistics....
data.info()
```

• Handling Missing Values

```
data.isnull().any()     # to check any null
data.isnull().sum()      # to check the count of null
# there is no missing values so we go for next step.....
```

• Data Visualization

(i) Univarient Analysis

```
# Univarient Analysis
sns.histplot(x=data["GRE Score"],color='Red')          #Choosing the GRE Score
```

(ii) Bi-Variate Analysis

```
#bivariate analysis
sns.barplot(x=data["SOP"],y=data["CGPA"])    # Choosing SOP and CGPA column
```

(iii) Multi-Variate Analysis

```
# Multi-Variate Analysis
sns.pairplot(data)
```

• Find the outliers and replace them outliers

```
sns.boxplot(data["GRE Score"])
sns.boxplot(data["TOEFL Score"])
sns.boxplot(data["University Rating"])
sns.boxplot(data.SOP)
data['LOR '].median()
q1=data['LOR '].quantile(0.25)  #(Q1)          # Replacement using median....
q3=data['LOR '].quantile(0.75)  #(Q3)
IQR=q3-q1
upper_limit= q3 + 1.5*IQR
lower_limit= q1 - 1.5*IQR
data['LOR ']= np.where(data['LOR ']<lower_limit,3.5,data['LOR '])
sns.boxplot(x=data['LOR '],showfliers=False)
sns.boxplot(data['CGPA'])
data['CGPA'].median()                          # median....
q1=data['CGPA'].quantile(0.25)  #(Q1)          # Replacement using median....
q3=data['CGPA'].quantile(0.75)  #(Q3)
IQR=q3-q1
upper_limit= q3 + 1.5*IQR
lower_limit= q1 - 1.5*IQR
data['CGPA']= np.where(data['CGPA']<lower_limit,8.61,data['CGPA'])
sns.boxplot(x=data['CGPA'],showfliers=False)
```

```
sns.boxplot(data['Research'])

sns.boxplot(data['Chance of Admit '])

data['Chance of Admit '].median()                    # median....

q1=data['Chance of Admit '].quantile(0.25)  #(Q1)        # Replacement using median....

q3=data['Chance of Admit '].quantile(0.75)  #(Q3)

IQR=q3-q1

upper_limit= q3 + 1.5*IQR

lower_limit= q1 - 1.5*IQR

data['Chance of Admit ']= np.where(data['Chance of Admit
']<lower_limit,0.73,data['Chance of Admit '])

sns.boxplot(x=data['Chance of Admit '],showfliers=False)

# Finally we Replaced all the outliers using median.....
```

• Split the data into dependent and independent columns

(i) Independent variable

```
x=data.iloc[:,0:7].values

x
```

(ii) Dependent variable

```
y=data.iloc[:,7:].values

y

x.shape

y.shape
```

• Splitting the Data into train and test

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=10)

y_train=(y_train>0.5)

y_train

y_test=(y_test>0.5)

y_test

# if you observe y column it has the probability of stutent getting admitted into a university.

# I would like to convert this probability as either yes or no .....  ( so the logic y_train and y_test as above )
```

• Training and Testing The Model

```
from sklearn.linear_model import LogisticRegression

cls =LogisticRegression(random_state =0)

lr=cls.fit(x_train, y_train)

y_pred =lr.predict(x_test)

y_pred
```

• Model Evaluation

```
from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matrix

print("\nAccuracy score: %f" %(accuracy_score(y_test,y_pred)*100))

print("Recall score : %f" %(recall_score(y_test,y_pred)*100))

print("ROC score : %f\n" %(roc_auc_score(y_test,y_pred)*100))
```

print(confusion_matrix(y_test,y_pred))

Accuracy score: 92.500000
Recall score : 98.666667
ROC score : 49.333333

[[0 5]
 [1 74]]

• IBM Deployment

!pip install -U ibm-watson-machine-learning

from ibm_watson_machine_learning import APIClient

import json

Authenticate and Set Space

wml_credentials = {

"apikey":"ZpX-YXfeK9dPqp41dYk1ly1faNv4XM2XfYomx8pIcdZW",

"url":"https://us-south.ml.cloud.ibm.com"

}

URLS Dallas: https://us-south.ml.cloud.ibm.com, London - https://eu-gb.ml.cloud.ibm.com, Frankfurt - https://eu-de.ml.cloud.ibm.com, Tokyo - https://jp-tok.ml.cloud.ibm.com

wml_client = APIClient(wml_credentials)

wml_client.spaces.list()

SPACE_ID= "e51e0c2e-ab07-4522-b64a-824811f7ed38"

wml_client.set.default_space(SPACE_ID) ## SUCCESS

wml_client.software_specifications.list(500)

• Save and Deploy the model

import sklearn

sklearn.__version__

MODEL_NAME = 'University_Admit_Eligibility_Predictor'

DEPLOYMENT_NAME = 'University_Admit_Eligibility_Predictor'

```
DEMO_MODEL = lr
# Set Python Version
software_spec_uid = wml_client.software_specifications.get_id_by_name('runtime-22.1-py3.9')
# Setup model meta
model_props = {
    wml_client.repository.ModelMetaNames.NAME: MODEL_NAME,
    wml_client.repository.ModelMetaNames.TYPE: 'scikit-learn_1.0',
    wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
}



#Save model
model_details = wml_client.repository.store_model (
    model=DEMO_MODEL,
    meta_props=model_props,
    training_data=x_train,
    training_target=y_train
)
model_details
model_id = wml_client.repository.get_model_id(model_details)
model_id

# Set meta
deployment_props = {
```

```
    wml_client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
    wml_client.deployments.ConfigurationMetaNames.ONLINE: {}
}

# Deploy
deployment = wml_client.deployments.create(
    artifact_uid=model_id,
    meta_props=deployment_props
)

#######################################################################################

Synchronous deployment creation for uid: 'be651858-f626-4a89-bfc5-19881938c2c1' started

#######################################################################################

initializing
Note: online_url is deprecated and will be removed in a future release. Use serving_urls instead.

ready

------------------------------------------------------------------------------------------------
Successfully finished deployment creation, deployment_uid='16fab11a-5a0c-4180-8216-d760800e43d3'
------------------------------------------------------------------------------------------------
```

• html files

(index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <style>
        body{
            background:url("../static/img/img1.jpg") no-repeat center fixed;
            background-size: cover;
            }
```

```
        .t{
            color:Darkblue;
            text-align:center;
        }
    </style>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>University Admit Eligibility Predictor</title>
</head>
<body>
    <h1 class="t"> UNIVERSITY ADMISSION PREDICTION SYSTEM </h1>

    <h2> Enter your details and get probability your admission </h2>
<form action='/y_predict', method='post'>
Enter GRE Score: <input type="text" name="GRE" placeholder="GRE" required="required" />
<br><br>
Enter TOEFL Score: <input type="text" name="TOEFL" placeholder="TOEFL"
required="required" />
<br><br>
University Rating :
<br>
<input type="radio" id="1" name="no" value="1"/><label>1</label><br>
<input type="radio" id="2" name="no" value="2"/><label>2</label><br>
<input type="radio" id="3" name="no" value="3"/><label>3</label><br>
<input type="radio" id="4" name="no" value="4"/><label>4</label><br>
<input type="radio" id="5" name="no" value="5"/><label>5</label><br>
<br>
Enter SOP: <input type="text" name="SOP" placeholder="SOP" required="required"  />
<br><br>
 Enter LOR: <input type="text" name="LOR" placeholder="LOR" required="required"  />
<br><br>
 Enter CGPA: <input type="text" name="CGPA" placeholder="CGPA" required="required"  />
<br><br>
 Research:
<br>
<input type="radio" id="1" name="res" value="1"/><label>Research</label><br>
<input type="radio" id="0" name="res" value="0"/><label>No Research</label><br>
<br>
<input type="submit" value="Predict"/>

</form>

</body>
</html>
```

(chance.html)

```
<html>
   <head>
      <style>
         body{
            background-image:url("../static/img/img2.jpg");
            background-repeat:no-repeat;
            background-attachment:fixed;
            background-position:center;
         }
         h1,h3{
            text-align:center;
         }
      </style>
      <title> chance </title>
   </head>




   <body>
      <h1> Predicting Chance of Admission </h1>
      <h3> Prediction : <b><u>You have a chance</u> !!...</b> <img
src="../static/img/img3.jpg" height="40px" width="40px" /></h3>
      <h3><a href="/">Go Back..</a></h3>

   </body>
</html>
```

(noChance.html)

```
<html>
   <head>
      <style>
         body{
            background-image:url("../static/img/img5.jpg");
            background-repeat:no-repeat;
            background-size:500px 500px;
            background-attachment:fixed;
            background-position:center;
         }
         h1,h3{
```

```
            text-align:center;
        }
    </style>
    <title> chance </title>
  </head>
  <body>
    <h1> Predicting Chance of Admission </h1>
    <h3> Prediction : <b><u>You don't have a chance</u> !!...</b> <img
src="../static/img/img4.jpg" height="40px" width="40px" /></h3>
    <h3><a href="/">Go Back..</a></h3>
  </body>
</html>
```

• app.py (run with the help of flask)

```
import requests
import numpy as np
from flask import Flask, redirect, url_for, render_template, request

# NOTE: you must manually set API_KEY below using information retrieved from your IBM
Cloud account.
API_KEY = "ZpX-YXfeK9dPqp41dYk1ly1faNv4XM2XfYomx8pIcdZW"
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__)

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


@app.route("/y_predict", methods=['POST'])
def y_predict():
  int_features=[float(x) for x in request.form.values()]
  final_features=[list(int_features)]

  # NOTE: manually define and pass the array(s) of values to be scored in the next line
  payload_scoring = {"input_data": [{"fields": [int_features], "values": final_features}]}
  response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/16fab11a-5a0c-4180-8216-
```

```
d760800e43d3/predictions?version=2022-11-08', json=payload_scoring,
headers={'Authorization': 'Bearer ' + mltoken})
    print(response_scoring)
    predictions=response_scoring.json()
    predict = predictions['predictions'][0]['values'][0][0]
    print("Final prediction :",predict)


    # showing the prediction results in a UI# showing the prediction results in a UI
    if predict==True:
        return render_template("chance.html")
    else:
        return render_template("noChance.html")

if __name__=='__main__':
    app.run()
```

GitHub Link :
https://github.com/IBM-EPBL/IBM-Project-12307-1659447149

Project Demo Link :
https://universityadmiteligibilitypredict-donotdelete-pr-teafgkiepxpvip.s3.us.cloud-object-storage.appdomain.cloud/PNT2022TMID32168%20(demo%20video).mkv

UAEP Notebook link :
https://universityadmiteligibilitypredict-donotdelete-pr-teafgkiepxpvip.s3.us.cloud-object-storage.appdomain.cloud/notebook/University_Admit_Eligibility_Predictor_Yg0DEF8A6.ipynb

dataset link :
https://universityadmiteligibilitypredict-donotdelete-pr-teafgkiepxpvip.s3.us.cloud-object-storage.appdomain.cloud/Admission_Predict.csv