University Admit Eligibility Predictor

Team Id:PNT2022TMID12677

Project Report

1. **INTRODUCTION**
   - **1.1 Project Overview**

     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
   - **1.2 Purpose**

     This system mainly helps the student to predict the university according to their academic performance,GRE score ,etc.This prevents the students from spending a lot of money on consulting firms.

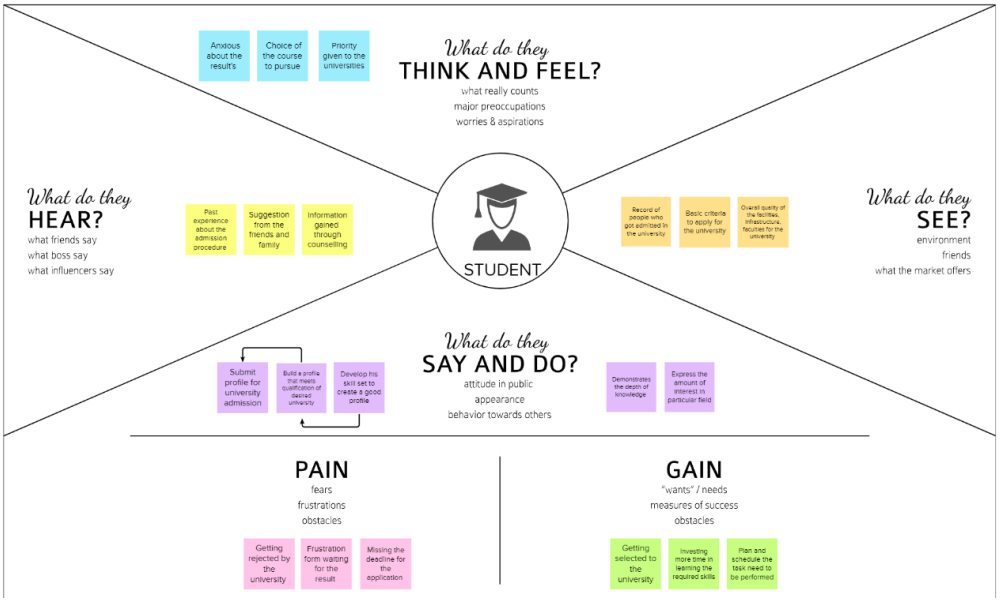
2. **LITERATURE SURVEY**
   - **2.1 Existing problem**

     Students need a way to predict the university according to their test scores and academic performance. The main problem involved in this was students spends lot of money on consulting firms.

   - **2.2 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. The model of the project is developed using advanced data sciences and python. The Model uses data of the student like his marks, rank in different exams, grade points, research etc to predict the eligibility of the student to get into a university.

3. **IDEATION & PROPOSED SOLUTION**
   - **3.1 Empathy Map Canvas**



3.2 Ideation & Brainstorming

[Drive for the Ideation & Brainstorming]

3.3 Proposed Solution

Problem Statement (Problem to be solved):

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.

Idea / Solution description:

The solution to this problem is proposed such that the eligibility of the student for getting a seat in the university will be predicted using a machine learning model . This model will be trained on the existing dataset so that it learn's the parameters to predict the eligibility of a particular student for the university admission .

Novelty / Uniqueness :

The machine learning model will automate the process of predicting the eligibility of the student for getting into a university.

Social Impact / Customer Satisfaction:

This solution will help the student to analyze their academic performance and their exam scores,so that they can confidently apply for their desired university. This solution also provides help to students who are preparing or will be preparing to get a better idea on their eligibility to the university.

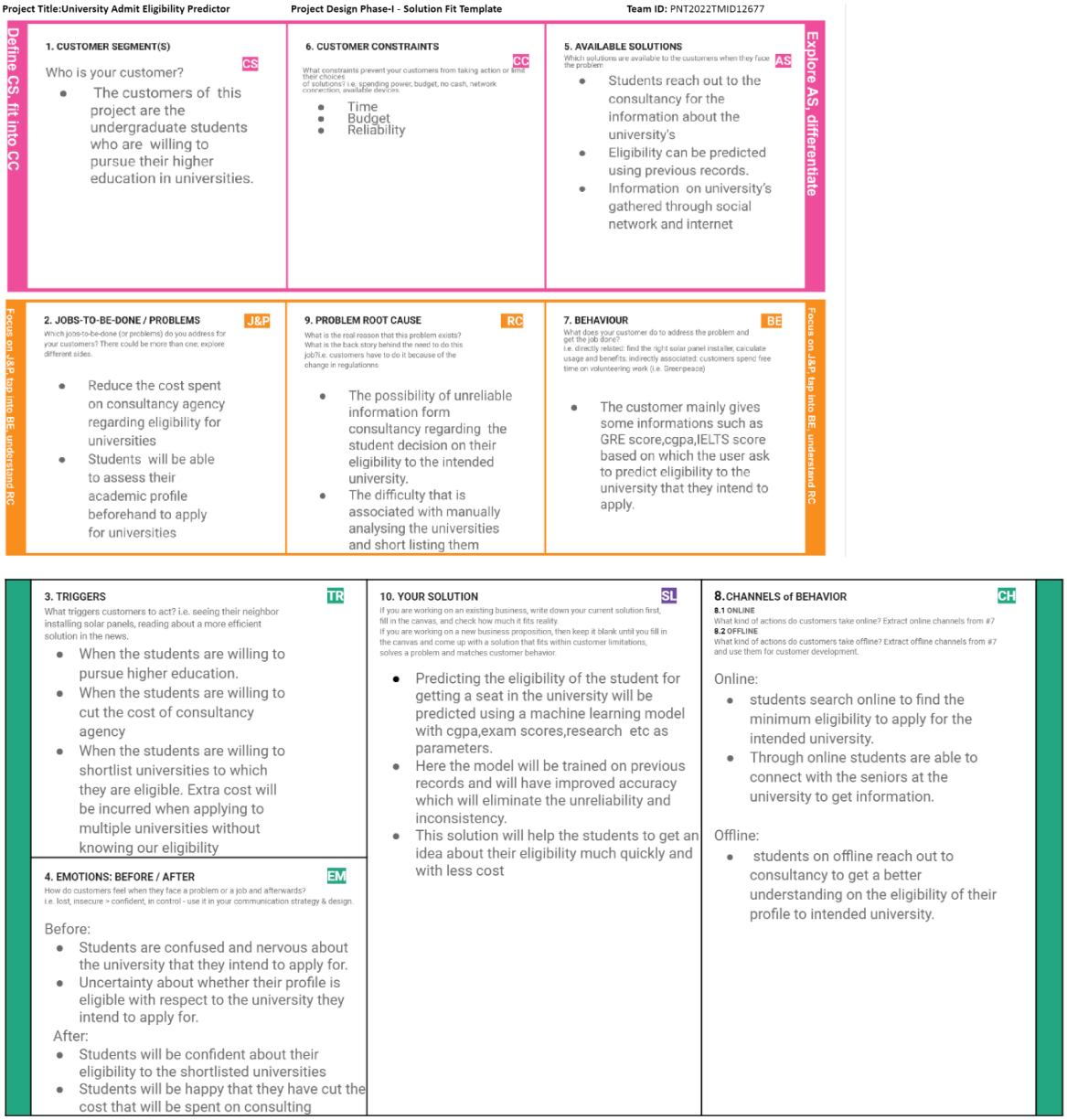
Business Model (Revenue Model):

This solution will help the students to get knowledge about the eligible university without spending on the consultancy agency.

Scalability of the Solution:

This solution can be scaled so that it is utilized by the undergraduate student's within an institution.

3.4 Problem Solution fit:



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

User Registration:

Registration through Form Registration through Gmail

User Details:

Collecting details such as CGPA, GRE score,TOEFL score , university rating, SOP , LOR

Log In:

Login with ID Login with Password

User eligibility prediction:

Analyze the given user details and predict the user's eligibility to given university rank

Change User Details:

Edit any personal information of the user even after initial registration

View previous results:

Enable the user to view the previous results and their corresponding inputs

Log Out:

Log out the user given the user is already logged in

4.2 Non-Functional requirements

Usability:

The application developed will be easier for the user to use and provides safety,effectiveness and efficiency

Security:

The user details and his results are protected using password thus enabling security

Reliability:

The user will be provided with accurate results with respect to the given inputs

Performance:

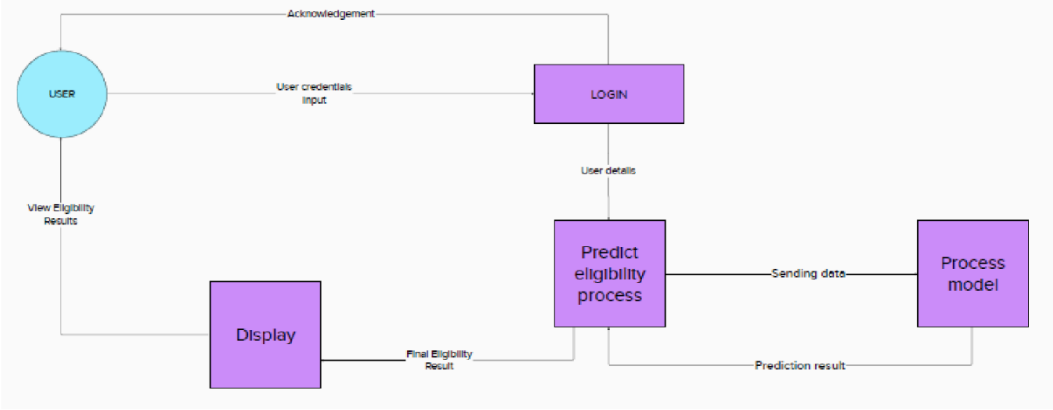
The user will be able to view the eligibility to the university instantly

Availability:

It is available to a wide range of users and can be accessed across any browsers

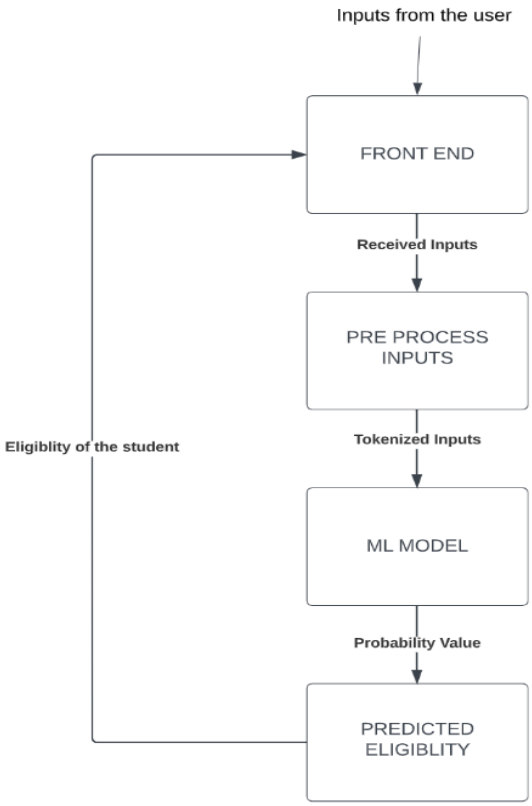
5. PROJECT DESIGN

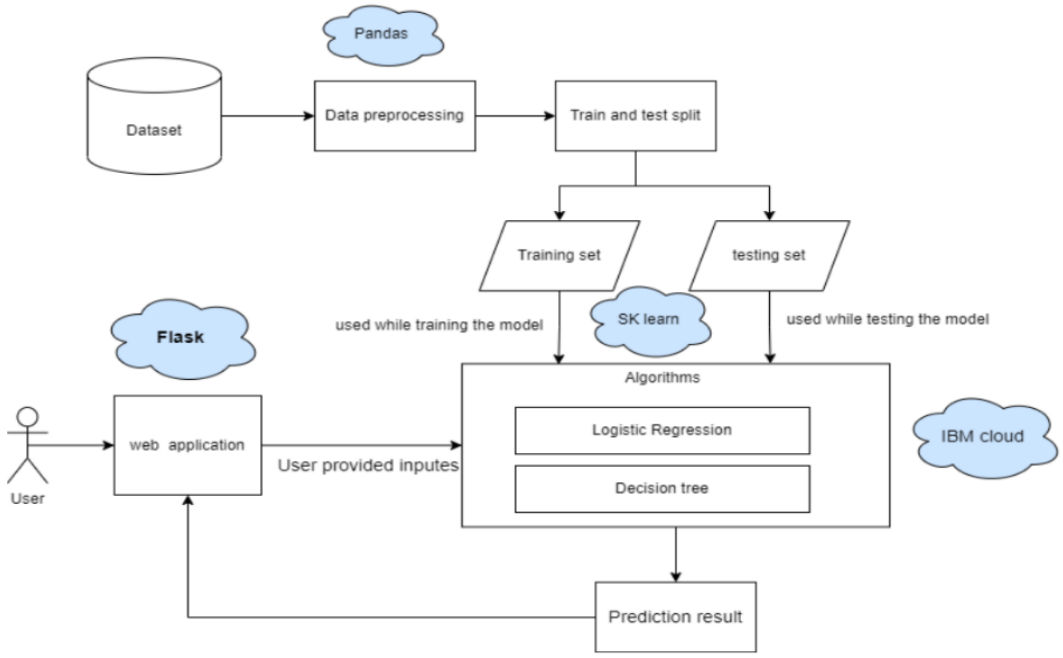
5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture

SOLUTION ARCHITECTURE





5.3 User Stories

Customer :

USN – 1:

As a user, I can register for the application by entering my username, password, age, gender, and confirming my password.

USN-2:

As a user, I can log into the application by entering email & password

USN-3:

As a user, after logging in, I will have to update my profile by providing all the required details.

USN-4:

As a user I will be able to enter the different scores like GRE, TOEFL, SOP etc

USN-5:

As a user I will be able to enter the desired university rank to find the admit eligibility

USN-6:

As a user I will be able to find my eligibility results after entering the details

USN-7:

As a user I will be able to view my previous prediction results

Administrator:

USN-8:

As an admin, the login credential of the user is authenticated by me.

USN-9:

As an admin, I will be able to verify the user entered details.

USN-10:

As an admin, I can test the trained ML model by analyzing the user details by ML algorithms like Logistic Regression.

USN-11:

As an admin, I can upload the confirmation of the user for the prediction into the Database.

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	User Registration	USN-1	During the registration, As a user you can register for the application byentering the mail id, password, and confirming my password.	2	High	2
Sprint-2		USN-3	As a user, you can verify the eligibility criteria for various universities by uploading the required details and documents	2	Low	2
Sprint-4	User Login	USN-4	As a user, you can log into the application byentering email & password. If both are matched the user are allowed to enter into system . If not matched, it shows pop up screen containing invalid credentials	1	High	2
	Dashboard		Check dashboard for results and upload the details according to the desired andeligible universities based on the eligibility criteria.			4

6.2 Sprint Delivery Schedule

Sprint	Total Story point	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (Planned)	Sprint Release Date (Actual)
Sprint 1	20	6 Days	26 Oct 2022	29 Oct 2022	20	30 Oct 2022
Sprint 2	20	6 Days	30 Oct 2022	05 Nov 2022	20	06 Nov 2022
Sprint 3	20	6 Days	5 Nov 2022	12 Nov 2022	20	13 Nov 2022
Sprint 4	20	6 Days	10 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 Reports from JIRA

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

Feature : Prediction of higher educational institutions based on the given data.

Code for deploying model in IBM:

Importing the required libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

loading dataset

```
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='cGszht1k97TyvA61S0S68Ddx_WlKxZGCynh-KELvweqd',
    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 = 'universityadmitprediction-donotdelete-pr-muquknbhnyvktk'
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 )

df = pd.read_csv(body)
df.head()
## Preparing the dataset
```

Importing the required libraries for regression analyzes

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
```

Spliting the dataset into training and testing data

```
x = df[["GRE Score","TOEFL Score","University Rating","SOP","LOR ","CGPA", "Research"]]
y = df["Chance of Admit "].values.reshape(-1,1)

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

MULTIPLE LINEAR REGRESSION

```
#implying multiple linear regression and determining its score

multiple_lin_reg = LinearRegression()
multiple_lin_reg.fit(x_train,y_train)

y_pred_mlr = multiple_lin_reg.predict(x_test)

r2_score_mlr = r2_score(y_test,y_pred_mlr)
print("Mutiple Linear Regression's Score = {:.3f}".format(r2_score_mlr))
```

```
Mutiple Linear Regression's Score = 0.821
```

DECISION TREE REGRESSION

```
#implying decision tree regression and determining its score

tree_reg = DecisionTreeRegressor()
tree_reg.fit(x_train,y_train)

y_pred_tree = tree_reg.predict(x_test)
```

DECISION TREE REGRESSION

```
#implying decision tree regression and determining its score

tree_reg = DecisionTreeRegressor()
tree_reg.fit(x_train,y_train)

y_pred_tree = tree_reg.predict(x_test)

r2_score_tree = r2_score(y_test,y_pred_tree)
print("Decision Tree Regression's Score = {:.3f}".format(r2_score_tree))
```

```
Decision Tree Regression's Score = 0.611
```

RANDOM FOREST REGRESSION

```
#implying random forest regression and determining its score

ran_for_reg = RandomForestRegressor(n_estimators=100,random_state=42)
ran_for_reg.fit(x_train,y_train)

y_pred_rfr = ran_for_reg.predict(x_test)

r2_score_rfr = r2_score(y_test,y_pred_rfr)
print("Random Forest Regression's Score = {:.3f}".format(r2_score_rfr))
```

```
Random Forest Regression's Score = 0.807
```

ESTABLISHING CONNECTION TO IBM WATSON ML SERVICE

```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
    "url": "https://us-south.ml.cloud.ibm.com",
    "apikey": "FMnFCKYFVzZjoPfxiqMInnYK2jc14D9GpJgN0mckp1wj"
}

client = APIClient(wml_credentials)
```

CREATING DEPLOYMENT SPACE

```
def guid_from_space_name(client, space_name):
    space = client.spaces.get_details()
    return (next(item for item in space['resources'] if item['entity']['name'] == space_name)['metadata']['id'])
```

```
space_uid = guid_from_space_name(client, 'models')
print("Space UID = " + space_uid)
```

```
Space UID = 3bb76528-89be-4868-a155-29f5c81da7b4
```

```
client.set.default_space(space_uid)
```

```
'SUCCESS'
```

```
client.software_specifications.list()
```

PERSISTING THE MULTIPLE LINEAR REGRESSION MODEL AND DEPLOYING IT IN IBM CLOUD

```
#Set Python Version
software_spec_uid = client.software_specifications.get_uid_by_name("runtime-22.1-py3.9")
software_spec_uid
```

```
'12b83a17-24d8-5082-900f-0ab31fbfd3cb'
```

```
model_details = client.repository.store_model(model = multiple_lin_reg, meta_props={
    client.repository.ModelMetaNames.NAME: "UAEP_Multiple_Linear_Regression",
    client.repository.ModelMetaNames.TYPE: "scikit-learn_1.0",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
}
)

model_id = client.repository.get_model_id(model_details)
```

```
model_id
```

```
'7f328e63-d660-4e29-9256-56d34e1f940d'
```

Training

```
x_train
```

UAEP Multiple Linear Regression Model Deployment Test

```
import requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "FMnFCKYFVzZjoPfxiqMInnYK2jc14D9GpJgN0mckp1wj"
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}

# NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{"field": [["GRE Score","TOEFL Score","University Rating","SOP","LOR ","CGPA", "Research"]], "values": [[326, 110, 2

response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/172d4b7a-ef21-46c8-8936-2962f34ee250/predictions?version=2022-11
 headers={'Authorization': 'Bearer ' + mltoken})
print("Scoring response")
print(response_scoring.json())
```

```
Scoring response
{'predictions': [{'fields': ['prediction'], 'values': [[[0.8339846446531018]]]}]}
```

```
probability = response_scoring.json()['predictions'][0]['values'][0][0][0]
probability
```

```
0.8339846446531018
```

Integration using flask:

```python
from flask import Flask, render_template, request, jsonify
import numpy as np
import requests
import json

app=Flask(__name__)

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

@app.route('/predictpage',methods=['POST'])
def predictpage():
    if request.method == 'POST':
        if request.form.get('action1') == 'GET STARTED':
            return render_template('predict.html')
    else:
        return render_template("homepage.html")
    # return render_template('predict.html')

@app.route('/predictpage/y_predict',methods=['POST'])
def y_predict():
        g_score= request.form["gscore"]
        t_score=request.form["tscore"]
        urs =request.form["urating"]
        sops=request.form["sop"]
        lors=request.form["lor"]
        gpa=request.form["cgpa"]
        rscore=request.form["research"]
        # deepcode ignore HardcodedNonCryptoSecret: <please specify a reason of ignoring this>
        API_KEY = "FMnFCKYFVzZjoPfxiqMInnYK2jc14D9GpJgN0mckp1wj"
        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}

        payload_scoring = {"input_data": [
            {"field": [["GRE Score", "TOEFL Score", "University Rating", "SOP", "LOR ", "CGPA", "Research"]],
             "values": [[g_score,t_score , urs, sops, lors, gpa, rscore]]}]}

        response_scoring = requests.post(
            'https://us-south.ml.cloud.ibm.com/ml/v4/deployments/172d4b7a-ef21-46c8-8936-2962f34ee250/predictions?version=2022-11-15',
            json=payload_scoring,
            headers={'Authorization': 'Bearer ' + mltoken})
        res = response_scoring.json()
        print(res)
        result=res['predictions'][0]['values'][0][0][0]
        coa=str(result*100)[:5]
        if result < 0.5:
            output = "Chance of admit is " + coa +" %."+ "Low chance"
        elif result >= 0.5 and result < 0.65:
            output = "Chance of admit is " + coa +" %."+  "Moderate chance"
        elif result >= 0.65 and result < 0.8:
            output = "Chance of admit is " + coa +" %."+  "Good chance"
        elif result >= 0.8 and result <= 1.0:
            output = "Chance of admit is " + coa +" %."+  "High chance"
        return render_template('Result.html',prediction_text=output)

@app.route('/predictpage/y_predict/redirectpage',methods=['POST'])
def redirectpage():
    if request.method == 'POST':
        if request.form.get('action1') == 'Retry':
            return render_template('predict.html')
    else:
        return render_template("homepage.html")

if __name__ == "__main__":
    app.debug=True
    app.run()
```

8. TESTING

8.1 Test Cases

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
View Home Page	10	0	1	9
Enter the scores	20	0	1	19
Click Submit button	2	0	0	2
Image displayed	10	0	3	7
Selecting from Drop down	5	0	0	5
Final Report Output	30	0	10	20
Version Control	5	0	2	3

8.2 User Acceptance Testing

Acceptance Testing
UAT Execution & Report Submission

Date	17 November 2022
Team ID	PNT2022TMID12677
Project Name	Project – University Admit Eligibility Predictor
Maximum Marks	4 Marks

1. Purpose of Document

The goal of this document is to discuss briefly the test coverage and open issues of the University Admit Eligibility Predictor project at the time of the User Acceptance Testing release (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	0	1	0	1	2
Duplicate	1	0	1	0	2
External	1	2	0	0	3
Fixed	0	1	1	0	2
Not Reproduced	0	1	0	0	1
Skipped	0	0	0	0	0
Won't Fix	0	0	1	0	1
Totals	2	5	3	1	11

9. RESULTS

Performance Metrics:

Accuracy: The model used was multiple linear regression and accuracy obtained was 89%

Responsive time:450ms.

10. ADVANTAGES & DISADVANTAGES:

Advantages: This prevents the students from spending a lot of money on consulting firms.This application also gives the idea about the documents that are required for continuing the higher education.

Disadvantages: This system takes only fixed number document details and predicts the result.The requirements for different universities are different.

11. CONCLUSION:

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.

12. FUTURE SCOPE

The dataset which we use will be changed in future by altering the number of features.The model which will be trained with updated dataset gives the better result.

13. APPENDIX

SourceCode:

https://github.com/IBM-EPBL/IBM-Project-13169-1659512755/tree/main/Final%20Deliverables/Final%20Code

GitHub & Project Demo Link: https://youtu.be/lFhrHRVqSxc

GitHub link