

ARIGNAR ANNA GOVERNMENT ARTS COLLEGE VILLUPURAM – 605 602.



DEPARTMENT OF COMPUTER SCIENCE

MACHINE LEARNING WITH PYTHON

Project Title: Intelligent admission : The future of University decision making with machine learning

Team Id: NM2023TMID16940

Team Leader: PRADEEP C (3BA685731E9438B967C48ACCB11107B0)

Team member: PRASANTH K (88015081CD6BE0799514DDDACFBF77D8)

Team member: PUSHPARAJ A (DC1E9CF6A561692859C4786CD96BF74B)

Team member: RAMAKRISHNAN V (51294FFBE1532E1A6CC4E9852C8B32B3)

Abstract

The intelligent admissions system is a revolutionary approach to the decision-making process for university admissions. This system utilizes machine learning algorithms to analyze various data points such as student demographics, academic achievements, extracurricular activities, and other relevant information. The system then uses this information to predict the probability of a student's success in a particular program or course. The system is designed to provide universities with a more accurate and efficient method of making admissions decisions. Additionally, the system offers students the opportunity to receive personalized recommendations based on their unique profile. This project aims to create an intelligent admissions system that can improve the quality of decision-making and enhance the overall student experience.

1. INTRODUCTION

Overview:

The intelligent admissions system is a project aimed at revolutionizing the university admissions process by utilizing machine learning algorithms to make better and more accurate admissions decisions. The system analyzes various data points such as student demographics, academic achievements, extracurricular activities, and other relevant information to predict the probability of a student's success in a particular program or course.

The intelligent admissions system also offers personalized recommendations to students based on their unique profile, providing them with valuable insights into their strengths and weaknesses and helping them make informed decisions about their academic future.

The ultimate goal of this project is to create a more efficient and effective admissions process that benefits both universities and students. By leveraging the power of machine learning, the intelligent admissions system has the potential to transform the way universities make admissions decisions and provide students with a more personalized and rewarding academic experience.

Purpose:

The purpose of the intelligent admissions system is to improve the university admissions process by utilizing machine learning algorithms to make better and more accurate admissions decisions.

Traditionally, university admissions decisions are based on limited information such as academic transcripts, standardized test scores, and personal essays. However, this information does not always provide a comprehensive picture of a student's potential for success in a particular program or course.

The intelligent admissions system aims to fill this gap by analyzing a wider range of data points and utilizing machine learning algorithms to predict the probability of a student's success. By providing universities with a more accurate and efficient method of making admissions decisions, the system can help universities identify and admit the most qualified and capable students.

Additionally, the intelligent admissions system offers students personalized recommendations based on their unique profile, which can help them make informed decisions about their academic future. By leveraging the power of machine learning, the intelligent admissions system has the potential to transform the way universities make admissions decisions and provide students with a more personalized and rewarding academic experience.

2. PROBLEM DEFINITION AND DESIGN THINKING

Empathy map:



Empathy for the future of University decision making with machine learning

Empathy for the future of University decision making with machine learning is an interesting topic. As machine learning and artificial intelligence continue to advance, Machine learning algorithms can be used to analyze large amounts of data and help universities make informed decisions regarding various aspects of their operations.



Says

- "We need to be careful not to rely too much on the data and to consider individual circumstances."
- "We need to make sure the algorithms are fair and unbiased."
- "We need to be transparent about how we're using machine learning and why."

"I'm not sure which courses to take to fulfill my degree requirements."

"I'm interested in pursuing a particular field, but I'm not sure which courses or programs to choose."

By listening to the concerns and needs of students, universities can better understand how to use machine learning to support their success and well-being

"I'm having difficulty balancing my academic workload with my personal or work responsibilities."

Making data-driven decisions is an important aspect of optimizing resources in any organization, including universities

course demand, and resource utilization, which can help them make more informed decisions about how to allocate resources effectively.

By collecting and analyzing data, universities can gain insights into areas such as student performance

By combining data with human judgement and empathy, universities can make more holistic and effective decisions about resource allocation.

Future of University decision making for intelligent admission

Collaborate with colleagues across departments to share data and insights. Develop and implement policies and procedures to ensure that algorithms are used ethically and equitably.

Use machine learning algorithms to optimize resource allocation, such as scheduling courses and assigning faculty to courses.

Feel overwhelmed or uncertain about how to effectively use machine learning to support students. Experience frustration when data is incomplete or difficult to access.

It's important for universities to prioritize ongoing training and support for faculty and staff to ensure that they feel confident and capable in their use of machine learning, and to regularly evaluate and adjust algorithms and processes to minimize bias and promote equity.

Create and implement personalized learning plans for students based on their unique needs and learning styles.

Continuously evaluate and adjust machine learning algorithms and processes to ensure that they are effective and ethical.

Feel a sense of responsibility to ensure that algorithms are used in a way that is fair and equitable for all students.

By acknowledging these feelings and concerns, faculty and staff can work together to develop strategies for implementing machine learning that prioritize the needs and well-being of students while also meeting the goals of the university.

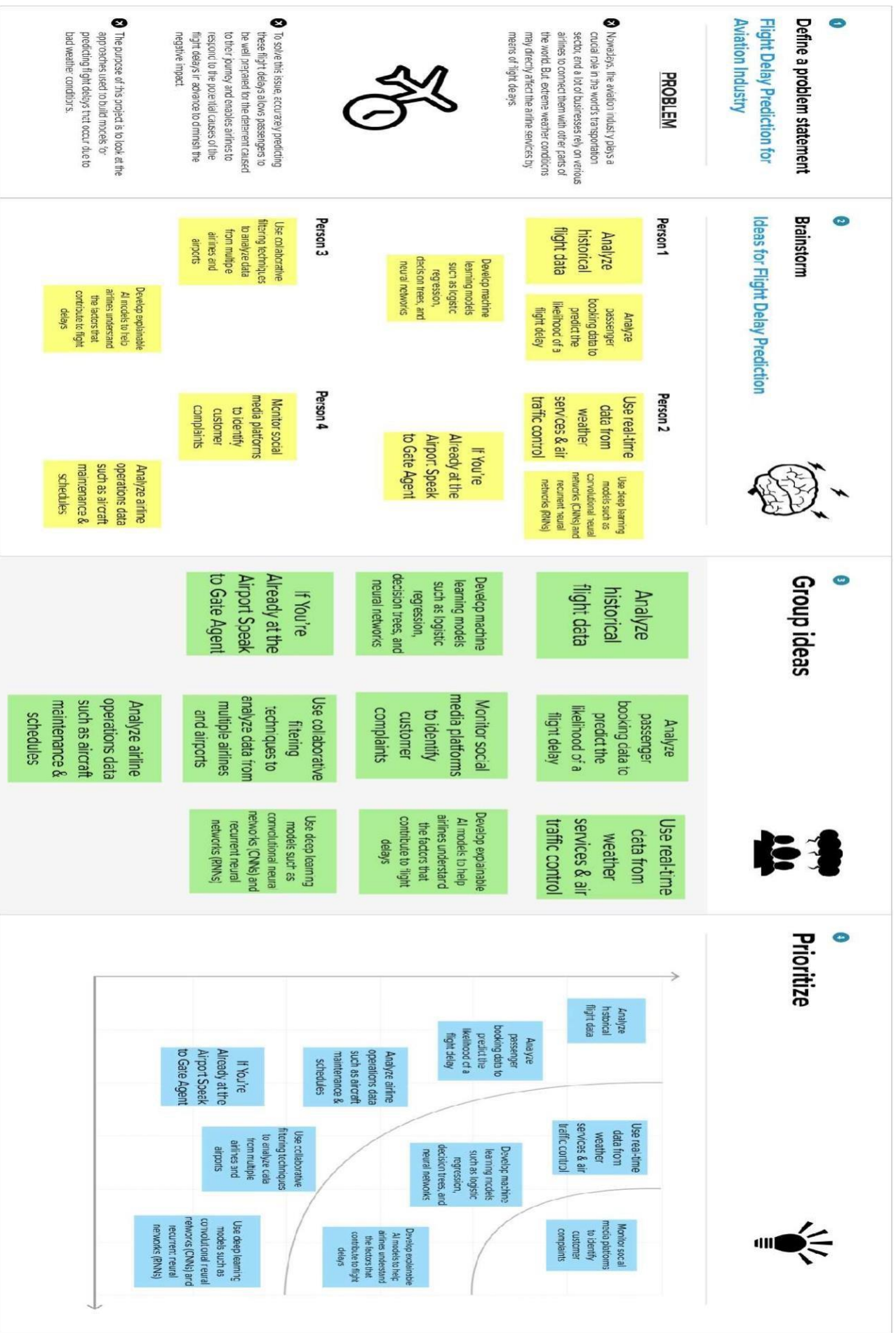
Does

- Analyze data and make decisions based on the insights provided by machine learning.
- Monitor and adjust the algorithms to ensure fairness and accuracy.
- Engage in discussions with stakeholders about the use of machine learning in decision making.

Feels

- Excitement about the potential of machine learning to improve decision making.
- Concern about the potential for bias in the algorithms.
- Fear that machine learning could dehumanize decision making processes.

Ideation & Brainstorming Map:



3.RESULT

index x +

127.0.0.1:5000

New Tab Search JEE Main Syllabus 2... தமிழ்நாடு அரசு... Gmail YouTube Maps Translate associate degree -... medical coding - G... How to Become a...

UNIVERSITY ADMISSION PREDICTION SYSTEM

Enter your details and get probability of your admission

Enter GRE Score

Enter TOEFL Score

Select University No

- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5

Enter SOP Score

Enter LOR Score

Enter CGPA Score

Research

- ☐ Research
- ☐ No Research

Type here to search

08:13 PM 18-04-2023

index x +

127.0.0.1:5000

New Tab Search JEE Main Syllabus 2... தமிழ்நாடு அரசு... Gmail YouTube Maps Translate associate degree -... medical coding - G... How to Become a...

UNIVERSITY ADMISSION PREDICTION SYSTEM

Enter your details and get probability of your admission

Enter GRE Score

Enter TOEFL Score

Select University No

- ☒ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5

Enter SOP Score

Enter LOR Score

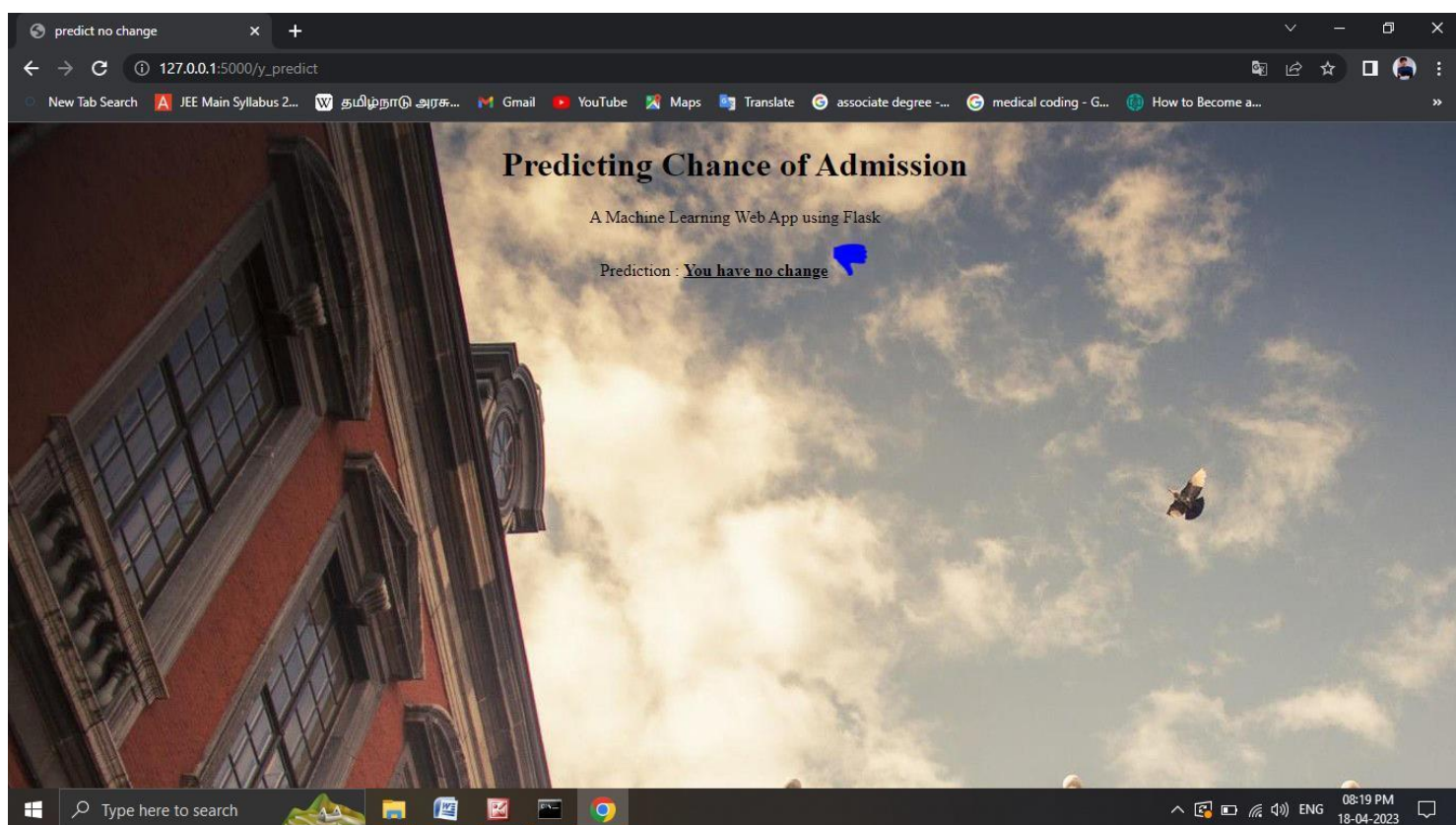
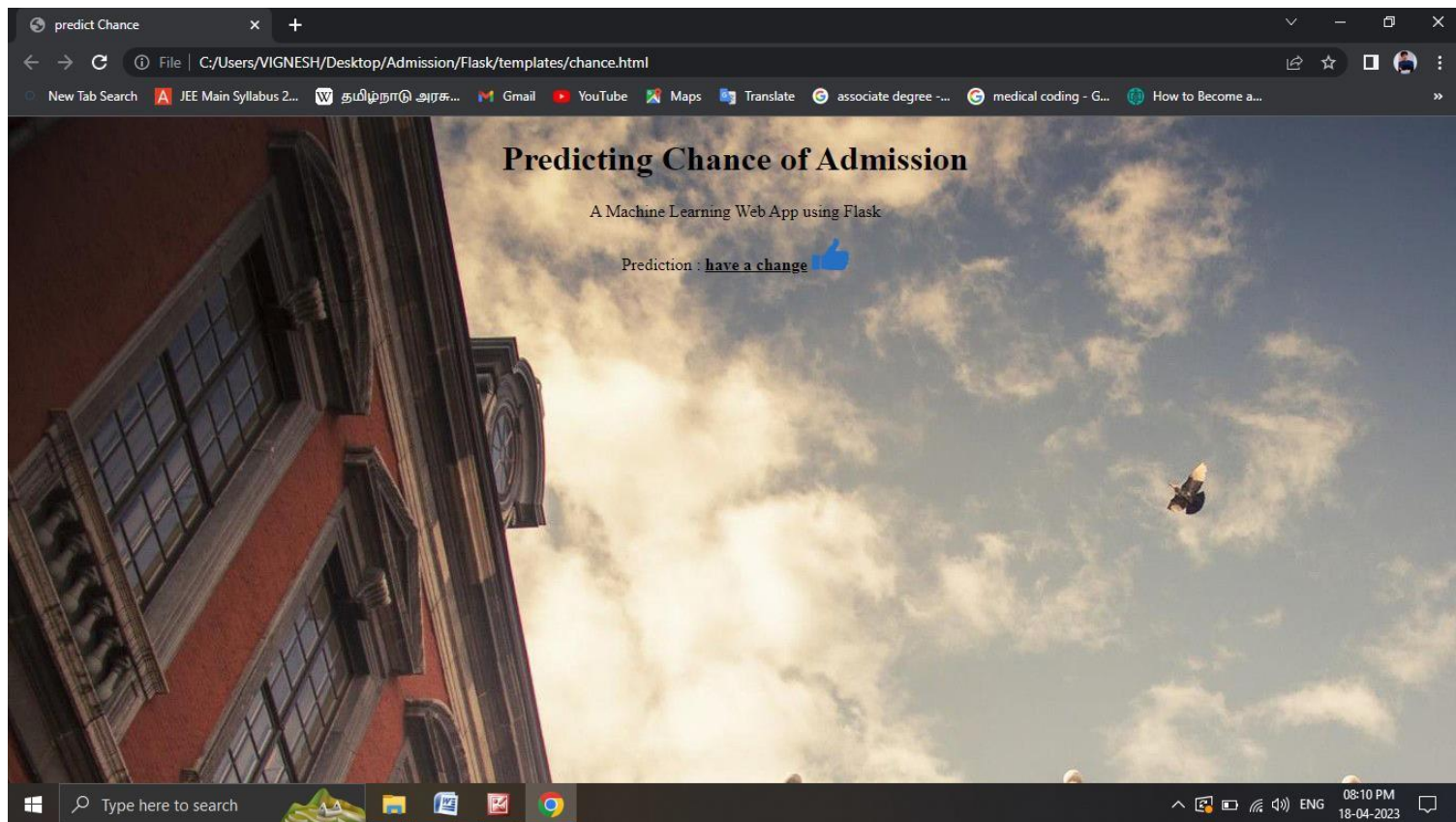
Enter CGPA Score

Research

- ☒ Research
- ☐ No Research

Type here to search

08:14 PM 18-04-2023



4.ADVANTAGES AND DISADVANTAGES

Advantages:

- **Increased efficiency:** Intelligent admission systems can process large amounts of data quickly and accurately, making the admissions process more efficient. This can save time and reduce the workload for admissions officers, who can focus on other important tasks.
- **Improved accuracy:** Machine learning algorithms can analyze vast amounts of data, including academic records, test scores, extracurricular activities, and other relevant factors, to identify patterns and make more accurate predictions about a student's likelihood of success at a particular university.
- **Fairness and transparency:** Intelligent admission systems can help reduce bias and ensure fairness in the admissions process by relying on objective data and algorithms rather than subjective judgments. This can help level the playing field for students from diverse backgrounds and reduce the impact of implicit bias on admissions decisions.
- **Better student matching:** By using machine learning algorithms to analyze data on student performance and preferences, universities can better match students with programs and courses that suit their strengths and interests. This can lead to higher retention rates and better outcomes for students.
- **Cost savings:** Intelligent admission systems can reduce the need for expensive and time-consuming manual processes, such as reviewing thousands of applications by hand, and can help universities save money on staffing and other resources.

Disadvantages:

- **Bias in the data:** Intelligent admission systems rely on historical data to make predictions about future student success, but this data may be biased due to factors such as socioeconomic status, race, gender, or other demographic variables. If the data used to train the machine learning algorithms is biased, this bias may be perpetuated in the admissions decisions, leading to unfair outcomes.
- **Lack of transparency:** Machine learning algorithms can be complex and difficult to understand, which can make it hard for stakeholders to understand how admissions decisions are being made. This lack of transparency can make it difficult for students to challenge decisions or for universities to ensure that the process is fair and equitable.
- **Inability to capture important non-academic factors:** While machine learning algorithms can analyze large amounts of academic data, they may not be able to capture important non-academic factors that are also important for success, such as grit, creativity, or social skills. Focusing too narrowly on academic metrics may miss out on students who have potential but do not fit traditional academic profiles.
- **Dependence on technology:** Intelligent admission systems are dependent on technology, which can be vulnerable to errors or hacking. If the system is not properly designed or maintained, it could lead to errors or biases in admissions decisions.
- **Reduction in human involvement:** While intelligent admission systems can streamline the admissions process, they may also lead to a reduction in human involvement. This could result in a lack of personal interaction and communication with applicants, leading to a less personalized and supportive experience for students.

5.APPLICATIONS

- **Predictive analytics:** Intelligent admission systems can use predictive analytics to identify patterns in student data and make predictions about future success. By analyzing academic records, test scores, extracurricular activities, and other relevant factors, these systems can identify students who are likely to succeed in particular programs or courses.
- **Personalized recommendations:** Intelligent admission systems can use machine learning algorithms to analyze student data and make personalized recommendations for courses, programs, and extracurricular activities that are best suited to the student's strengths and interests. This can help ensure that students are matched with programs and courses that are a good fit for their individual needs.
- **Streamlined application process:** Intelligent admission systems can automate many of the manual processes involved in the admissions process, such as data entry and document processing, leading to a more streamlined and efficient process for both students and admissions officers.
- **Improved diversity and inclusion:** Intelligent admission systems can help reduce bias in the admissions process by relying on objective data and algorithms rather than subjective judgments. This can help ensure that students from diverse backgrounds have an equal opportunity to be admitted to university programs.
- **Enhanced student support:** Intelligent admission systems can use data analytics to identify students who may be at risk of dropping out or falling behind and provide personalized support to help them succeed. This can include targeted interventions such as academic coaching, tutoring, or mentorship programs

6. CONCLUSION

- In conclusion, intelligent admission has the potential to revolutionize the university decision-making process by leveraging machine learning algorithms and data analytics to improve efficiency, fairness, and accuracy.
- By using predictive analytics to identify patterns in student data and making personalized recommendations for courses, programs, and extracurricular activities
- intelligent admission systems can help ensure that students are matched with programs and courses that are a good fit for their individual needs.
- Additionally, these systems can automate many of the manual processes involved in the admissions process, leading to a more streamlined and efficient process for both students and admissions officers.
- However, it is important to carefully consider both the advantages and disadvantages of intelligent admission, including potential biases in the data, lack of transparency, and dependence on technology.
- Overall, by implementing these systems ethically and with transparency, universities can harness the power of machine learning and data analytics to create a more efficient, equitable, and effective university decision-making process.

7. FUTURE SCOPE

- The future scope of intelligent admission is vast and promising, as machine learning and data analytics continue to advance and become more sophisticated. Here are a few potential areas where intelligent admission could be further developed and expanded in the future
- Multi-dimensional data analysis: While intelligent admission systems currently rely on academic data to make predictions about student success, future systems may incorporate additional data sources such as social media activity, personality assessments, and other non-academic factors. This could provide a more comprehensive understanding of each student's unique strengths, interests, and potential.
- Natural language processing: Natural language processing (NLP) is a subfield of machine learning that focuses on analyzing and understanding human language. In the future, NLP could be used to analyze application essays, personal statements, and other written materials to provide additional insights into each student's potential.
- Improved fairness and equity: While intelligent admission has the potential to reduce bias in the admissions process, future systems could be further developed to ensure fairness and equity. For example, algorithms could be designed to explicitly account for socioeconomic or other demographic factors that may affect a student's academic performance.

8.APPENDIX

Source Code:

Milestone 2:

```
+ Code + Text

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams
from scipy import stats
%matplotlib inline
data = pd.read_csv('/content/Admission_Predict.csv')
data.info()
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):

#	Column	Non-Null Count	Dtype
0	Serial No.	400 non-null	int64
1	GRE Score	400 non-null	int64
2	TOEFL Score	400 non-null	int64
3	University Rating	400 non-null	int64
4	SOP	400 non-null	float64
5	LOR	400 non-null	float64
6	CGPA	400 non-null	float64
7	Research	400 non-null	int64
8	Chance of Admit	400 non-null	float64

dtypes: float64(4), int64(5)
memory usage: 28.2 KB

✓ 0s completed at 12:51 PM

```
+ Code + Text

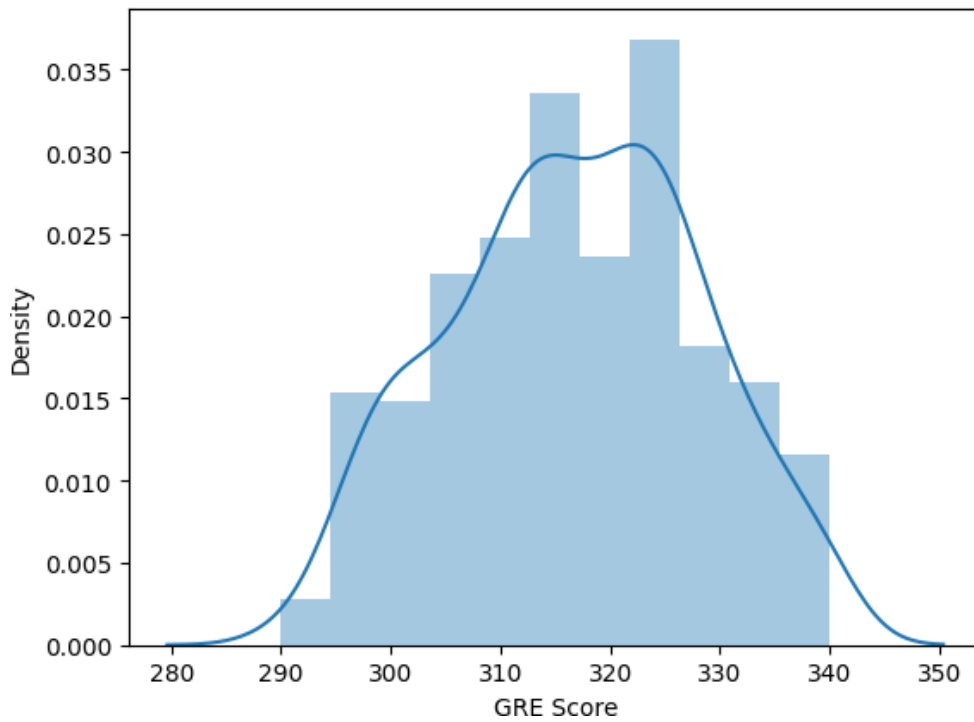
[3] data.isnull().any()

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

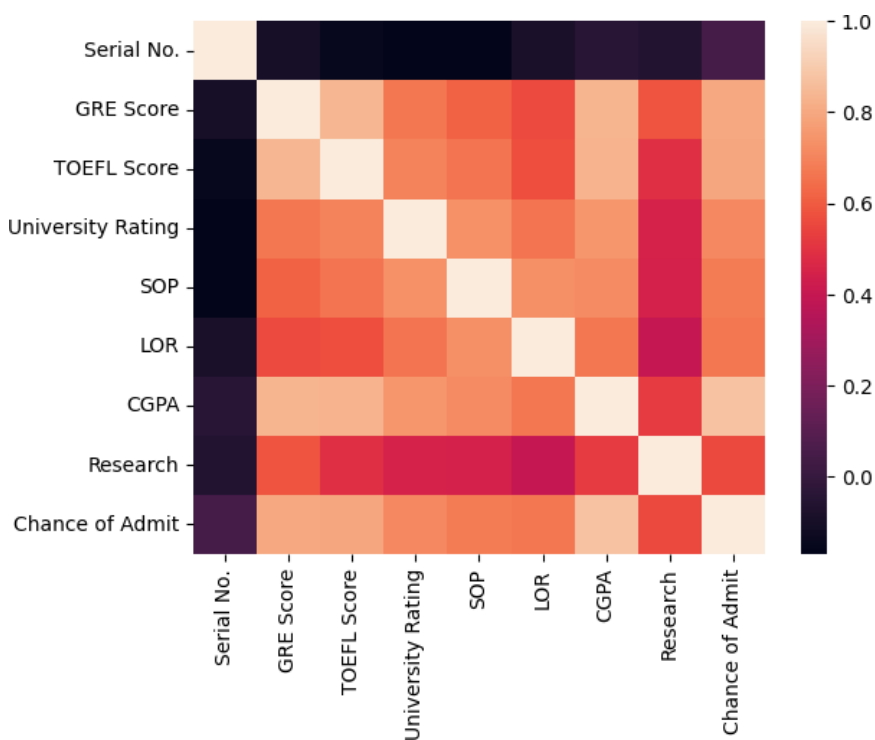
[4] data=data.rename(columns = {'Chance of Admit ':'Chance of Admit'})
data=data.rename(columns = {'LOR ':'LOR'})
```


Milestone 3:

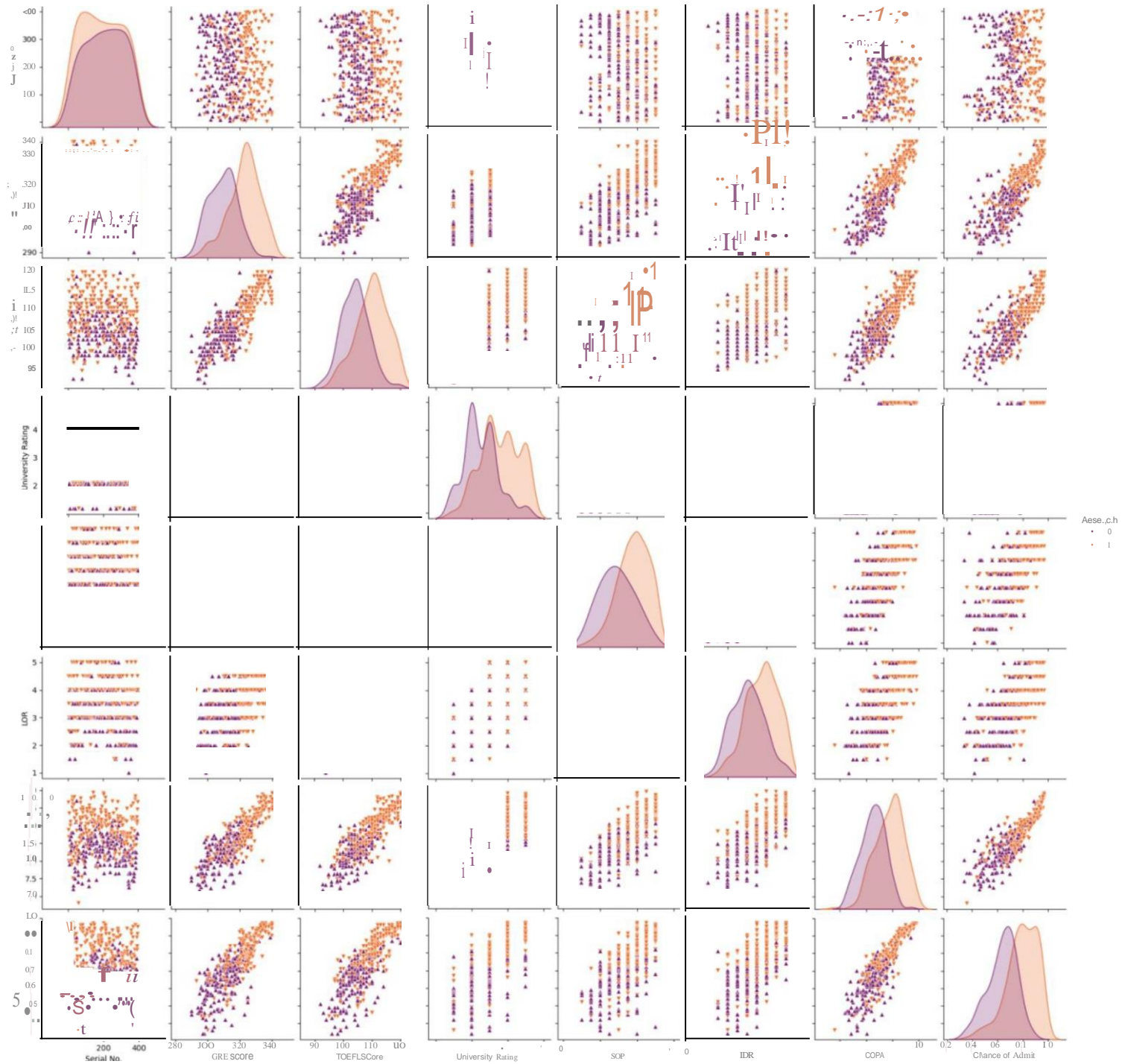
```
{x} sns.distplot(data['GRE Score'])
```



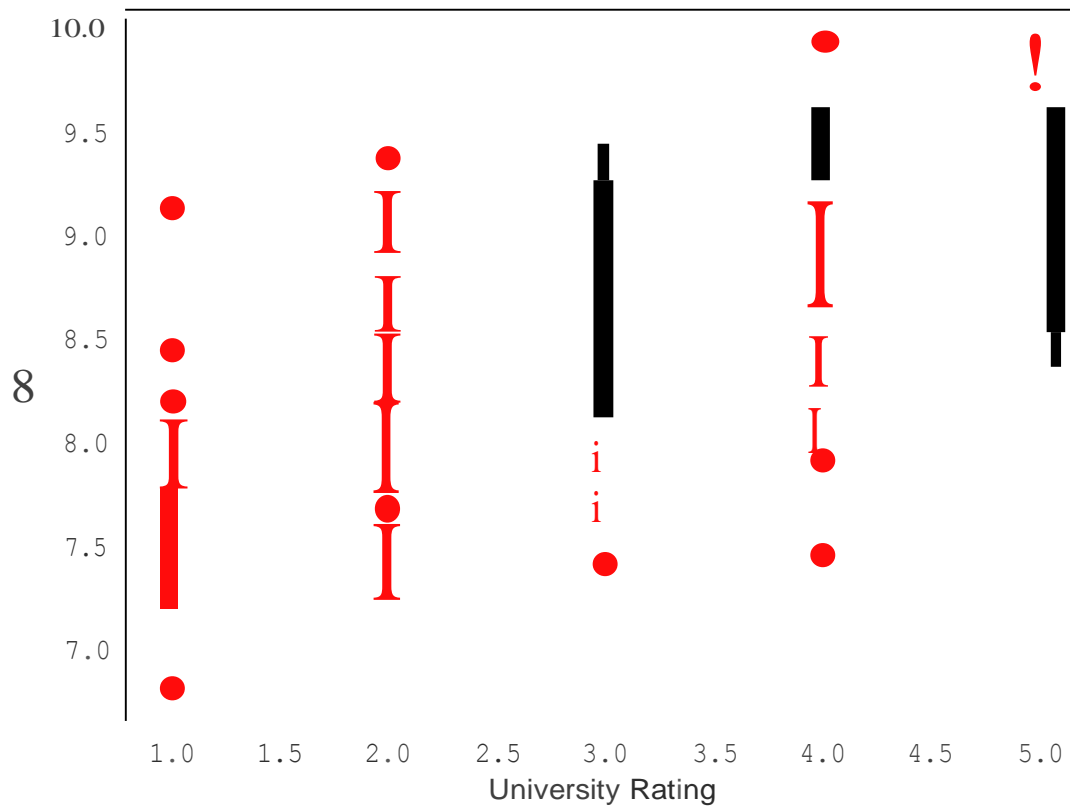
```
{x} 1s sns.heatmap(data.corr())
```



```
sns.pairplot(data=data,hue='F',
             size=11,markers='.',
             palette='l_r',
             style='m')
```



0 sns.scatterplot(x='University Rating', y='CGPA', data=data, color='Rec', 5=100)



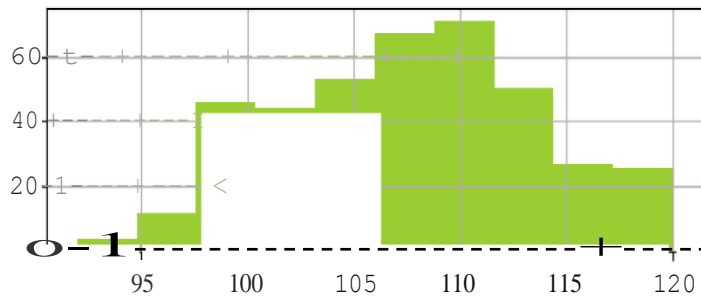
```

+ Code + Text

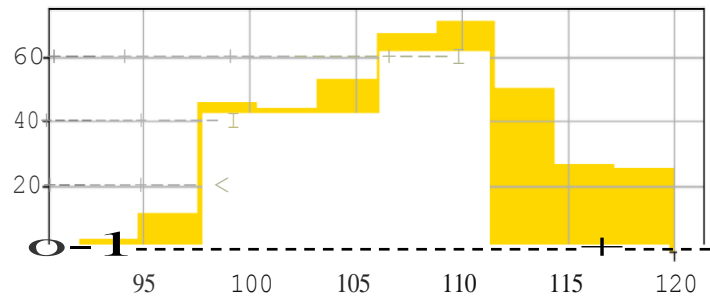
category = ['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR', 'CGPA', 'Research', 'Chance of Admit']
color = ['Yellowgreen', 'gold', 'lightskyblue', 'pink', 'red', 'purple', 'orange', 'gray']
start = True
for i in np.arange(4):
    fig = plt.figure(figsize=(14,8))
    plt.subplot2grid((4,2),(i,0))
    data[category[2*i+1]].hist(color=color[2*i],bins=10)
    plt.title(category[2*i])
    plt.subplot2grid((4,2),(i,1))
    data[category[2*i+1]].hist(color=color[2*i+1],bins=10)
    plt.title(category[2*i+1])
plt.subplots_adjust(hspace = 0.7, wspace = 0.2)
plt.show()

```

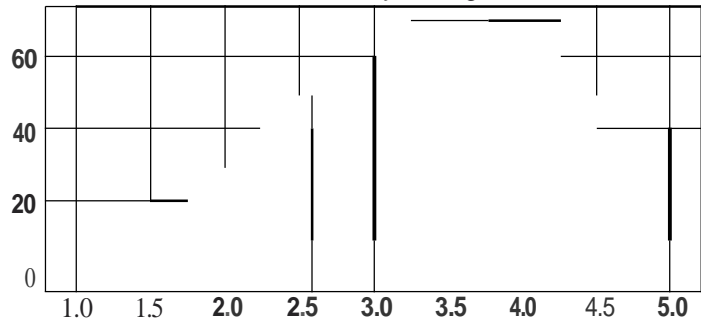

GRE Score



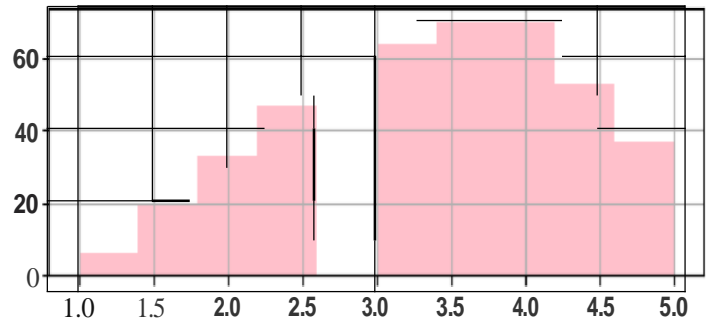
TOEFLScore



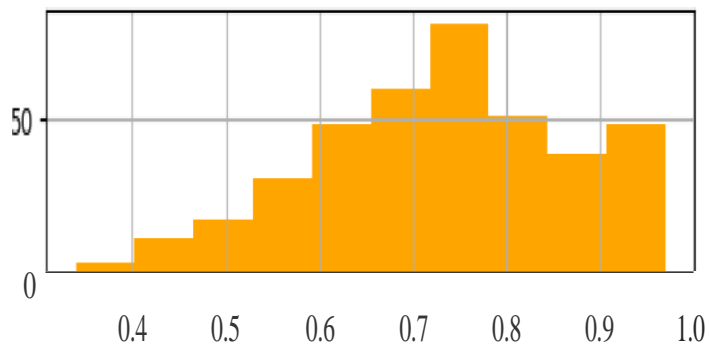
UniversityRating



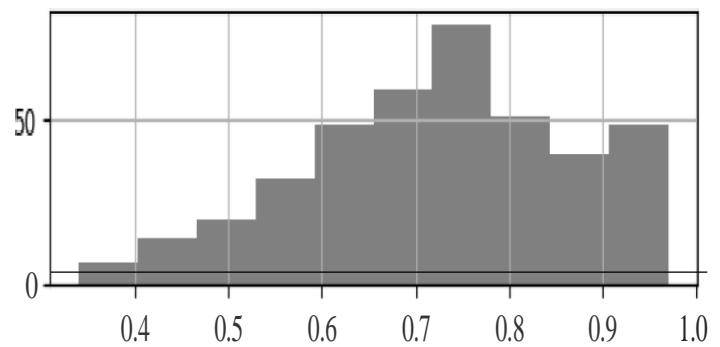
SOP



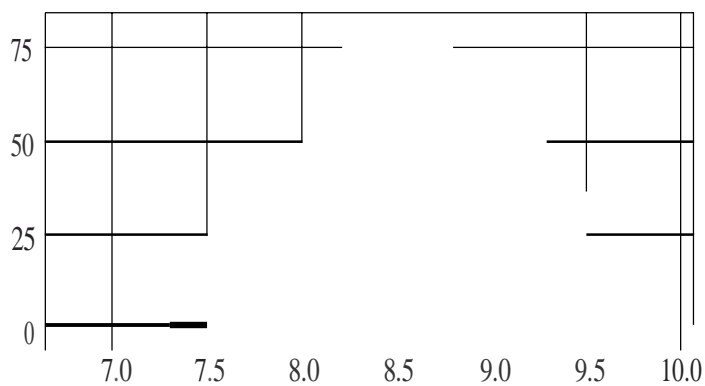
Research



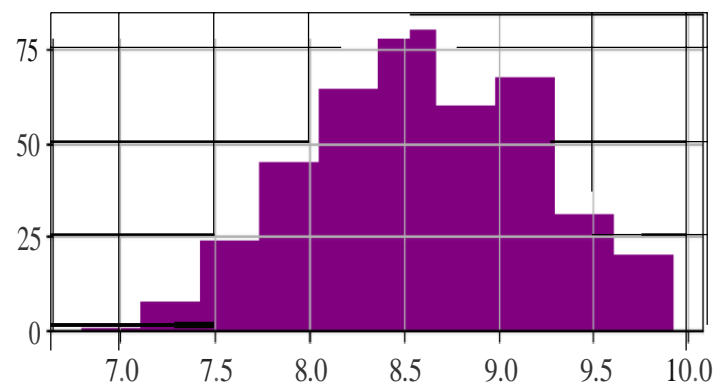
Chance of Admit



LOR



CGPA



+ Code + Text

```
0. [11] from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
x=data.iloc[:,0:7].values
```

{x}

```
0 y=data.iloc[:,7].values
print(y)
```

```
[0. 0.58]
[0. 0.59]
[0. 0.47]
[0. 0.49]
[0. 0.47]
[0. 0.42]
[0. 0.57]
[0. 0.62]
[1. 0.74]
[1. 0.73]
[1. 0.64]
[0. 0.63]
[0. 0.59]
[0. 0.73]
[1. 0.79]
[1. 0.68]
```

<>

13

!!!

Os completed at 12:51 PM

+ Code + Text

```
[12] [0. 0.51]
[1. 0.67]
[0. 0.72]
[1. 0.89]
[1. 0.95]
[1. 0.79]
[0. 0.39]
[0. 0.38]
[0. 0.34]
[0. 0.47]
[0. 0.56]
[1. 0.71]
[1. 0.78]
[1. 0.73]
[1. 0.82]
[0. 0.62]
[1. 0.96]
[1. 0.96]
[0. 0.46]
[0. 0.53]
[0. 0.49]
[1. 0.76]
[0. 0.64]
[0. 0.71]
[1. 0.84]
```

+ Code + Text

```
[13] x=sc.fit_transform(x)
print(x)
```

```
[[0. 0.94 0.92857143 ... 0.875 0.875 0.91346154]
[0.00250627 0.68 0.53571429 ... 0.75 0.875 0.66346154]
[0.00501253 0.52 0.42857143 ... 0.5 0.625 0.38461538]
...
[0.99498747 0.8 0.85714286 ... 1. 0.875 0.84935897]
[0.99749373 0.44 0.39285714 ... 0.625 0.75 0.63461538]
[1. 0.86 0.89285714 ... 1. 0.75 0.91666667]]
```

```
[14] from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.30,random_state=101)
y_train=(y_train>0.5)
print(y_train)
y_test=(y_test>0.5)
print(y_test)
```

```
[False True]
[False False]
[False True]
[False True]
[False True]
```

RAM
Disk

Milestone 4:

```
+ Code + Text

[14] [False True]
      [ True True]
      [False False]
      [ True True]
      [ True True]
      [ True True]
      [False True]
      [False True]
      [False True]
      [False True]
      [ True True]

[15] from sklearn.linear_model import LogisticRegression
      cls =LogisticRegression(random_state =0)
      lr=cls.fit(x_train, y_train.argmax(axis=1))
      y_pred =lr.predict(x_test)
      y_pred

array([1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0])
```

```
#libraries to train neural networks
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Sequential
#initialize the model
model=keras.Sequential()
#Add input layer
model.add(Dense(7,activation = 'relu',input_dim=7))
#Add hidden layer
model.add(Dense(7,activation='relu'))
#Add output layer
model.add(Dense(1,activation='linear'))
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 7)	56

✓ 0s completed at 12:51 PM


```
Q {x} [16] =====
4s dense (Dense) (None, 7) 56

dense_1 (Dense) (None, 7) 56

dense_2 (Dense) (None, 1) 8

=====
Total params: 120
Trainable params: 120
Non-trainable params: 0

[17] model: "sequential"
0s

[18] model.compile(loss='binary_crossentropy', optimizer='adam',
6s metrics=['accuracy'])
model.fit(x_train, y_train, batch_size=20, epochs=100)

14/14 [=====] - 0s 2ms/step - loss: 0.4402 - accuracy: 0.7804
Epoch 73/100
14/14 [=====] - 0s 2ms/step - loss: 0.4402 - accuracy: 0.7804
Epoch 74/100
14/14 [=====] - 0s 2ms/step - loss: 0.4400 - accuracy: 0.7804
0s completed at 12:51 PM
```

```
+ Code + Text
14/14 [=====] - 0s 2ms/step - loss: 0.4402 - accuracy: 0.7804
6s Epoch 74/100
14/14 [=====] - 0s 2ms/step - loss: 0.4400 - accuracy: 0.7804
Epoch 75/100
14/14 [=====] - 0s 2ms/step - loss: 0.4398 - accuracy: 0.7804
Epoch 76/100
14/14 [=====] - 0s 2ms/step - loss: 0.4406 - accuracy: 0.7804
Epoch 77/100
14/14 [=====] - 0s 2ms/step - loss: 0.4402 - accuracy: 0.7804
Epoch 78/100
14/14 [=====] - 0s 2ms/step - loss: 0.4396 - accuracy: 0.7804
Epoch 79/100
14/14 [=====] - 0s 3ms/step - loss: 0.4393 - accuracy: 0.7804
Epoch 80/100
14/14 [=====] - 0s 2ms/step - loss: 0.4394 - accuracy: 0.7804
Epoch 81/100
14/14 [=====] - 0s 2ms/step - loss: 0.4389 - accuracy: 0.7804
Epoch 82/100
14/14 [=====] - 0s 2ms/step - loss: 0.4395 - accuracy: 0.7804
Epoch 83/100
14/14 [=====] - 0s 3ms/step - loss: 0.4389 - accuracy: 0.7768
Epoch 84/100
14/14 [=====] - 0s 2ms/step - loss: 0.4387 - accuracy: 0.7768
Epoch 85/100
14/14 [=====] - 0s 2ms/step - loss: 0.4386 - accuracy: 0.7804
Epoch 86/100
0s completed at 12:51 PM
```

```

+ Code + Text
Epoch 88/100
14/14 [=====] - 0s 2ms/step - loss: 0.4382 - accuracy: 0.780
Epoch 89/100
14/14 [=====] - 0s 2ms/step - loss: 0.4381 - accuracy: 0.780
Epoch 90/100
14/14 [=====] - 0s 2ms/step - loss: 0.4379 - accuracy: 0.780
Epoch 91/100
14/14 [=====] - 0s 2ms/step - loss: 0.4380 - accuracy: 0.780
Epoch 92/100
14/14 [=====] - 0s 2ms/step - loss: 0.4380 - accuracy: 0.780
Epoch 93/100
14/14 [=====] - 0s 2ms/step - loss: 0.4378 - accuracy: 0.780
Epoch 94/100
14/14 [=====] - 0s 2ms/step - loss: 0.4378 - accuracy: 0.776
Epoch 95/100
14/14 [=====] - 0s 3ms/step - loss: 0.4400 - accuracy: 0.780
Epoch 96/100
14/14 [=====] - 0s 2ms/step - loss: 0.4391 - accuracy: 0.780
Epoch 97/100
14/14 [=====] - 0s 2ms/step - loss: 0.4396 - accuracy: 0.776
Epoch 98/100
14/14 [=====] - 0s 2ms/step - loss: 0.4378 - accuracy: 0.780
Epoch 99/100
14/14 [=====] - 0s 2ms/step - loss: 0.4377 - accuracy: 0.780
Epoch 100/100
14/14 [=====] - 0s 2ms/step - loss: 0.4376 - accuracy: 0.776
Time taken to train the model: 11m 11s
OS completed at 12:51 PM

```

```

+ Code + Text
14/14 [=====] - 0s 2ms/step - loss: 0.4377 - accuracy: 0.7804
[18] Epoch 100/100
14/14 [=====] - 0s 2ms/step - loss: 0.4376 - accuracy: 0.7768
<keras.callbacks.History at 0x7fe7b96f32b0>

from sklearn.metrics import accuracy_score
#make predictions on the training data
train_predictions = model.predict(x_train)
print(train_predictions)

[0.59748834]
[0.80043024]
[1.0052499]
[0.8724538]
[0.74945575]
[0.7959505]
[0.9801957]
[0.7956752]
[1.041937]
[0.7650639]
[0.7194241]
[0.19744772]
[0.46010217]
[0.5307102]
[0.76528466]

```

The screenshot shows a Jupyter Notebook interface with a dark theme. On the left sidebar, there are icons for file operations (three vertical bars), search (magnifying glass), code execution (play button), and other functions. The main area displays a list of 20 numerical values, each enclosed in square brackets. The values are: [0.7180634], [0.6540337], [1.0573319], [0.65322906], [0.58767647], [0.7636115], [0.6211141], [0.8392916], [0.6468839], [0.7143608], [0.8932875], [0.72693706], [0.37652457], [0.82695556], [0.81987995], [0.9492601], [0.25083324], [0.6802557], [0.9061159], [0.6418109], [0.9394229], [1.0109066], [0.76821476], [0.6695751], and [0.51159]. At the bottom right, a status bar indicates a successful completion with a green checkmark, '0s' time, and 'completed at 12:51 PM'.

✓ 2s	[0.7180634]
	[0.6540337]
	[1.0573319]
	[0.65322906]
	[0.58767647]
	[0.7636115]
	[0.6211141]
	[0.8392916]
	[0.6468839]
	[0.7143608]
	[0.8932875]
	[0.72693706]
	[0.37652457]
	[0.82695556]
	[0.81987995]
	[0.9492601]
	[0.25083324]
	[0.6802557]
	[0.9061159]
	[0.6418109]
	[0.9394229]
	[1.0109066]
	[0.76821476]
	[0.6695751]
	[0.51159]

✓ 0s completed at 12:51 PM

2s

▶

📄

[0.7180634]

[0.6540337]

[1.0573319]

[0.65322906]

[0.58767647]

[0.7636115]

[0.6211141]

[0.8392916]

[0.6468839]

[0.7143608]

[0.8932875]

[0.72693706]

[0.37652457]

[0.82695556]

[0.81987995]

[0.9492601]

[0.25083324]

[0.6802557]

[0.9061159]

[0.6418109]

[0.9394229]

[1.0109066]

[0.76821476]

[0.6695751]

[0.51159]

✓ 0s completed at 12:51 PM

```
+ Code + Text
✓ [19] [0.6000971 ]
2s [0.6385046 ]
[0.57613033]
[0.73316383]
[0.34014824]
[1.0108042 ]
[0.70238143]
[1.0440484 ]
[0.4761784 ]
[0.87881666]
[0.6528384 ]
[0.9031703 ]
[0.72381365]
[0.6248472 ]
[0.60756105]
[0.81453335]
[1.0267439 ]
[0.8783551 ]
[0.9724293 ]
[0.8994015 ]
[0.6880869 ]
[0.92019916]
[0.5022987 ]
[0.86423206]
[0.6721024 ]
[0.8472414 ]
```

✓ 2s [19]

[0.6000971]
[0.6385046]
[0.57613033]
[0.73316383]
[0.34014824]
[1.0108042]
[0.70238143]
[1.0440484]
[0.4761784]
[0.87881666]
[0.6528384]
[0.9031703]
[0.72381365]
[0.6248472]
[0.60756105]
[0.81453335]
[1.0267439]
[0.8783551]
[0.9724293]
[0.8994015]
[0.6880869]
[0.92019916]
[0.5022987]
[0.86423206]
[0.6721024]
[0.8472414]

✓ 0s completed at 12:51 PM

+ Code + Text

```
[20] #get the training accuracy
train_acc = model.evaluate(x_train, y_train, verbose=0)[1]
train_acc
```

$\{x\}$

0.7767857313156128

```
[21] #get the test accuracy
test_acc = rmodel.evaluate(x_test, y_test, verbose=0)[1]
print(test_acc)
```

,0.6958333253860474

```
0 pred=rmodel.predict(x_test)
pred = (pred>0.5)
pred
```

Os completed at 12:51 PM

A screenshot of a Jupyter Notebook interface. The top bar shows '+ Code' and '+ Text' tabs. The left sidebar contains icons for a menu, search, a variable '{x}', a file folder, and a code editor icon. The main area displays a code cell with a green checkmark and '0s' indicating successful execution. The cell contains a list of 20 boolean values: [True], [True], [True], [True], [True], [True], [True], [True], [True], [True], [True], [True], [True], [True], [True], [True], [True], [True], [True], [True]. The bottom status bar shows a green checkmark, '0s', and 'completed at 12:51 PM'.

Milestone 5:

```
+ Code + Text

[23] y_pred = y_pred.astype(int)
y_pred

array([1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0])

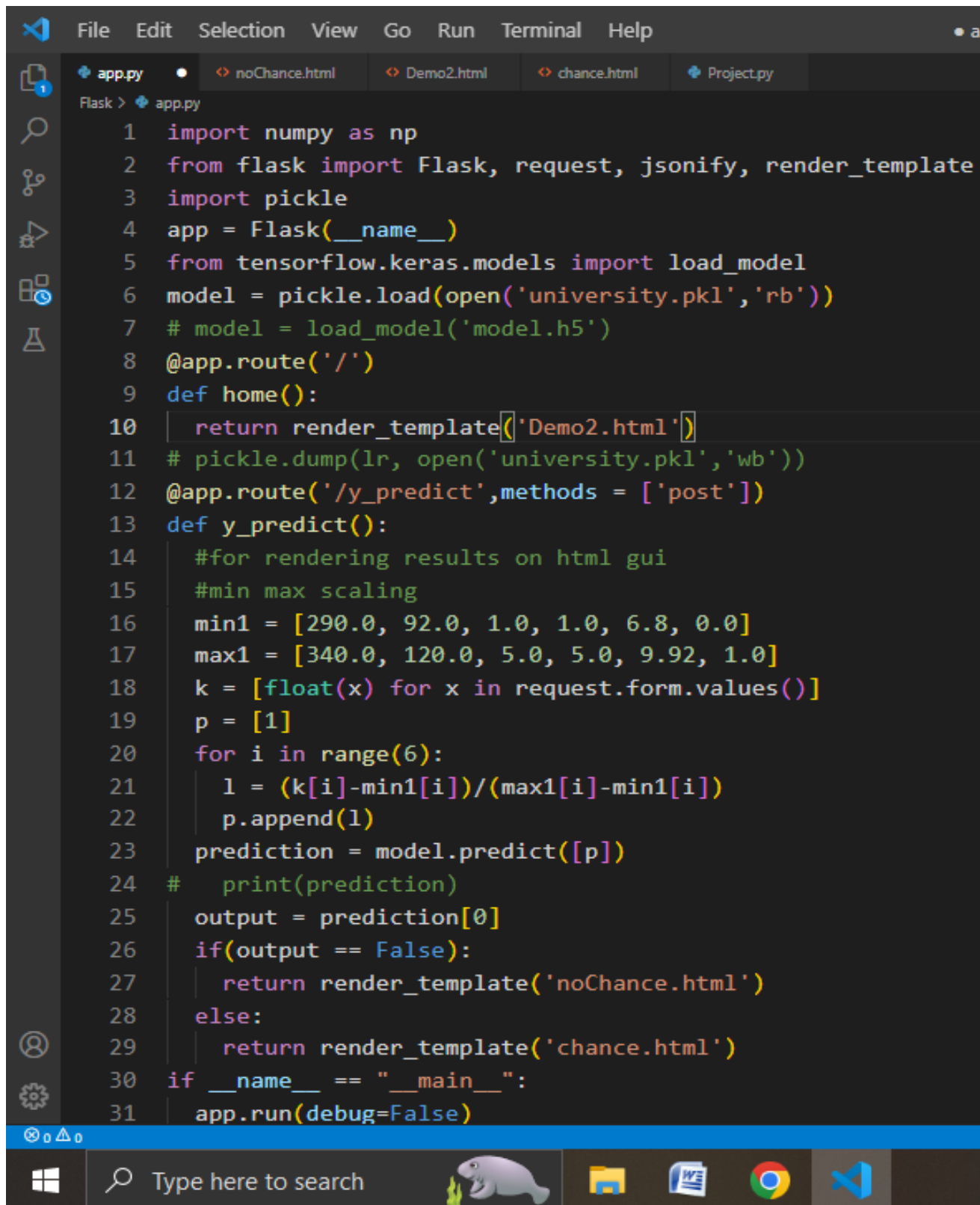
y_test = y_test.astype(int)
y_test

array([[0, 1],
       [0, 1],
       [1, 1],
       [0, 1],
       [1, 1],
       [1, 1],
       [0, 1],
       [0, 0],
       [1, 1],
       [0, 1],
```

```
+ Code + Text
✓ 0s [0, 0],
[0, 1],
[1, 1],
[0, 1],
[0, 1],
[0, 1],
[0, 1],
[0, 1],
[1, 1],
[1, 1],
[0, 1],
[1, 1],
[0, 1],
[1, 1],
[0, 1],
[1, 1],
[0, 1],
[1, 1],
[1, 1],
[1, 1],
[0, 0],
[1, 1],
[0, 1],
[0, 1],
[0, 1],
[0, 1],
[0, 1],
[0, 1],
[0, 1]
```


Milestone 6:

Flask file (app.py):



```
File Edit Selection View Go Run Terminal Help
app.py noChance.html Demo2.html chance.html Project.py
Flask > app.py
1 import numpy as np
2 from flask import Flask, request, jsonify, render_template
3 import pickle
4 app = Flask(__name__)
5 from tensorflow.keras.models import load_model
6 model = pickle.load(open('university.pkl','rb'))
7 # model = load_model('model.h5')
8 @app.route('/')
9 def home():
10     return render_template('Demo2.html')
11 # pickle.dump(lr, open('university.pkl','wb'))
12 @app.route('/y_predict',methods = ['post'])
13 def y_predict():
14     #for rendering results on html gui
15     #min max scaling
16     min1 = [290.0, 92.0, 1.0, 1.0, 6.8, 0.0]
17     max1 = [340.0, 120.0, 5.0, 5.0, 9.92, 1.0]
18     k = [float(x) for x in request.form.values()]
19     p = [1]
20     for i in range(6):
21         l = (k[i]-min1[i])/(max1[i]-min1[i])
22         p.append(l)
23     prediction = model.predict([p])
24     # print(prediction)
25     output = prediction[0]
26     if(output == False):
27         return render_template('noChance.html')
28     else:
29         return render_template('chance.html')
30 if __name__ == "__main__":
31     app.run(debug=False)
```

User Interface (index.html):



```
File Edit Selection View Go Run Terminal Help Demo2.h
app.py noChance.html Demo2.html x chance.html Project.py
Flask > templates > Demo2.html > html > body > div.Login > form
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6
7     <title>index</title>
8     <style>
9     div h1{
10         color: blue;
11         font-size: 2.7em;
12     }
13     div h2{
14         color: black;
15         font-size: 2em;
16     }
17     div button{
18         color: skyblue;
19         border: 1px solid skyblue;
20         padding: 10px;
21         border-radius: 5px ;
22     }
23     body{
24         background-image: url('../static/pic.jpg');
25         /* background-repeat: no-repeat; */
26     }
27     </style>
28 </head>
29 <body>
30     <div class="Login">
31         <h1>UNIVERSITY ADMISSION PREDICTION SYSTEM</h1>
```

```
File Edit Selection View Go Run Terminal Help Demo2.html - Admission - Visual Studio Code
app.py 1 noChance.html Demo2.html X chance.html Project.py
Rask > templates > Demo2.html > html > body > div.Login > form
24 background-image: url(../static/pic.jpg);
25 /* background-repeat: no-repeat; */
26 }
27 </style>
28 </head>
29 <body>
30 <div class="Login">
31 <h1>UNIVERSITY ADMISSION PREDICTION SYSTEM</h1>
32 <h2>Enter your details and get probability of your admission</h2>
33 <form action="{{ url_for('y_predict')}}" method="post">
34 Enter GRE Score <input type="text" name="gre" placeholder="GRE Score (out of 340)" required="required"><br><br>
35 Enter TOEFL Score <input type="text" name="toefl" placeholder="TOEFL Score (out of 120)" required="required"><br><br>
36 Select University No<br> <input type="radio" name="rating" value="1">1<br>
37 <input type="radio" name="rating" value="2">2<br>
38 <input type="radio" name="rating" value="3">3<br>
39 <input type="radio" name="rating" value="4">4<br>
40 <input type="radio" name="rating" value="5">5<br><br>
41 Enter SOP Score <input type="text" name="sop" placeholder="SOP (out of 5)" required="required"><br><br>
42 Enter LOR Score <input type="text" name="lor" placeholder="LOR (out of 5)" required="required"><br><br>
43 Enter CGPA Score <input type="text" name="cgpa" placeholder="CGPA (out of 10)" required="required"><br><br>
44 Research<br> <input type="radio" name="research" value="1">Research <br>
45 <input type="radio" name="research" value="0">No Research <br>
46
47 <button type="submit" class="btn btn-default">Predict</button>
48
49 </form>
50 {{prediction_text}}
51 </div>
52 </body>
53 </html>
```

```
File Edit Selection View Go Run Terminal Help chance.html - Admission - Visual Studio Code
app.py 1 noChance.html Demo2.html chance.html X Project.py
Rask > templates > chance.html > html > body > center > header > p > b
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <title>predict Chance</title>
7 <style>
8 body{
9 background-image: url('../static/pic2.jpg');
10 }
11 </style>
12 </head>
13 <body>
14 <center>
15 <header>
16 <h1><b>Predicting Chance of Admission</b></h1>
17 <p>A Machine Learning Web App using Flask</p>
18 <p>Prediction : <b><u>have a chance</u></b> </p>
19 </header>
20 </center>
21 </body>
22 </html>
```

