Intelligent Admissions: The Future of University Decision Making with Machine Learning

ABSTRACT:

During recent years, universities have become more and more dependent on the collection, storage and processing of educational data. The dynamics and transformation of military higher education, characterized by complex processes and statuses, generate an immense volume of data, and their acquisition and storage requires the use of the innovation in the IT field. In this context, these universities have become more and more dependent on the collection, storage and processing of educational data. Decision-makers try to apply new strategies and use new tools to convert this data in useful information that would contribute to managerial problem solving. Good decisions involve using some software tools that support decision-making process to maximize the performance of universities and minimize the negative impact of faults. In this paper, we present an overview of intelligent decision support systems (iDSS), and also our own conceptual model in designing a higher education iDSS. The proposed system will be composed of three subsystems working in an integrated manner in order to provide quality services to the iDSS beneficiaries, as follows: the Data Management Subsystem (DMS) that offers the necessary data in order to develop of communication between iDSS and the beneficiary, in the sense that the beneficiary supplies the iDSS with data and extracts useful information for the educational process.

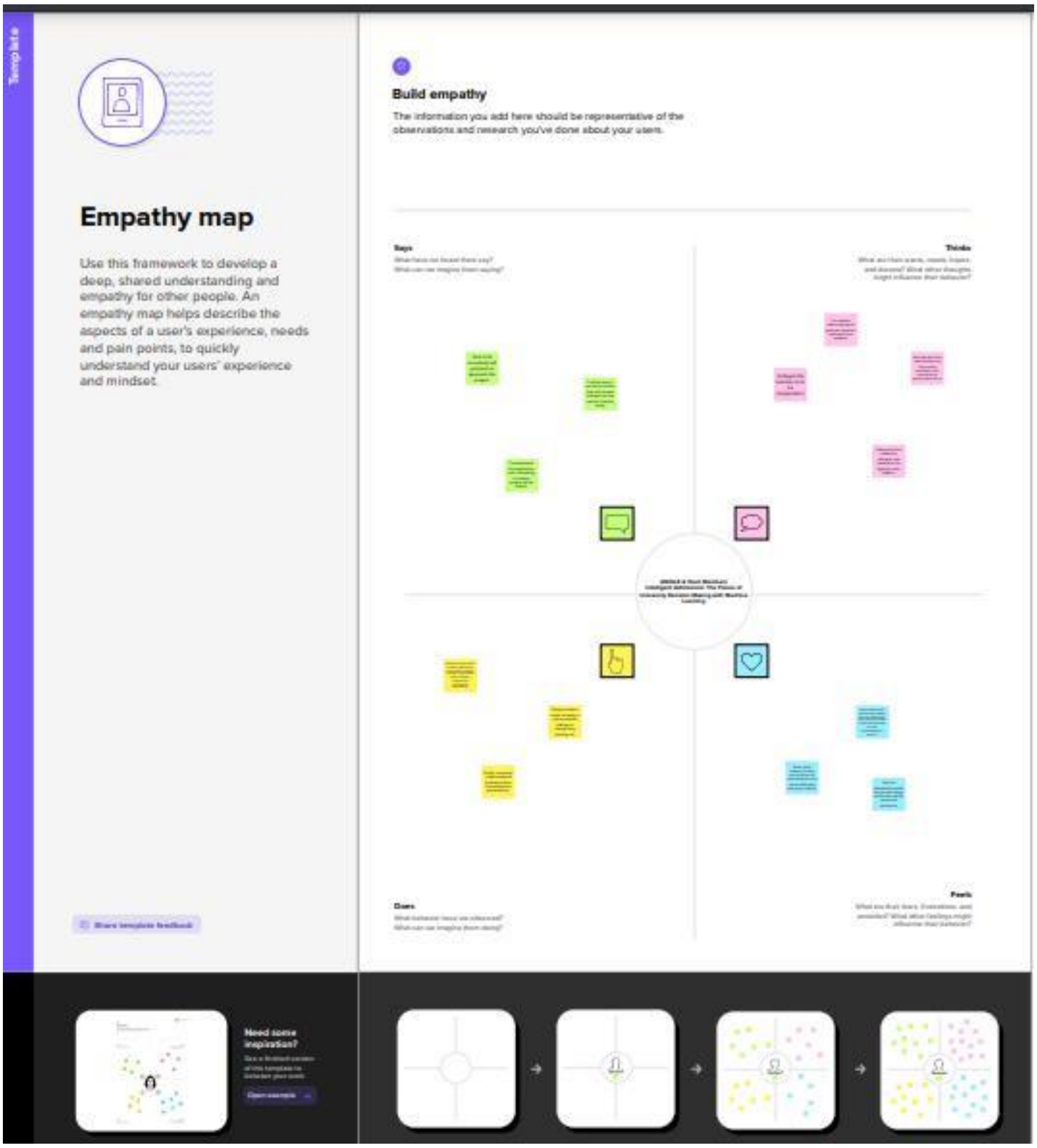
PURPOSE STATEMENT:

- The purpose of university education is to facilitate the advancement of knowledge and the development of high cognitive skills in the community. As a result, people become productive members of society who care about the well-being of others. The ability to articulate thoughts clearly is critical because it improves labor relations, which leads to improved market outcomes. It is vital to note that enlightened members of society often have a higher standard of living than individuals without the know-how required to thrive in a competitive environment.

- University education has evolved to meet the needs of an involved global community. It prepares individuals to deal with a variety of challenges by equipping them with problem-solving skills. In addition, they learn how to be influential leaders, adaptable professionals, and valuable members of society who appreciate diversity. It is important to note that graduates gain the prowess required to excel in their desired fields of practice, which allows them to make a living and lead a decent life. Institutions of higher learning prepare individuals to live in a constantly changing environment while being true to their core identities.

PROBLEM DEFINING & DESIGN THINKING:

EMPHATHY MAP:



IDEATION & BRAINSTORMING MAP:

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

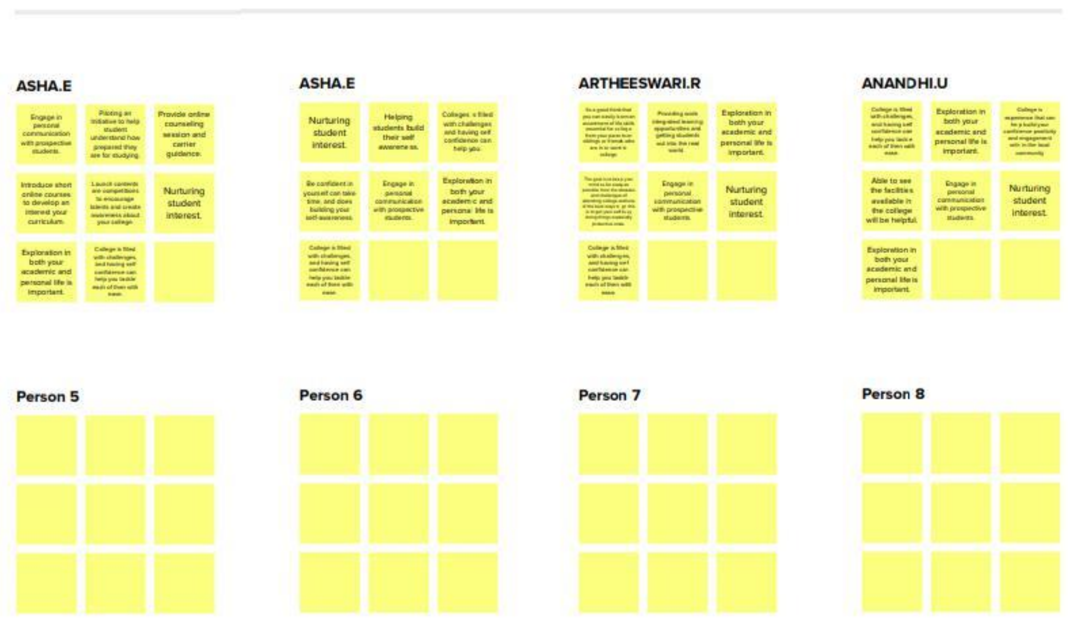
5 minutes

PROBLEM

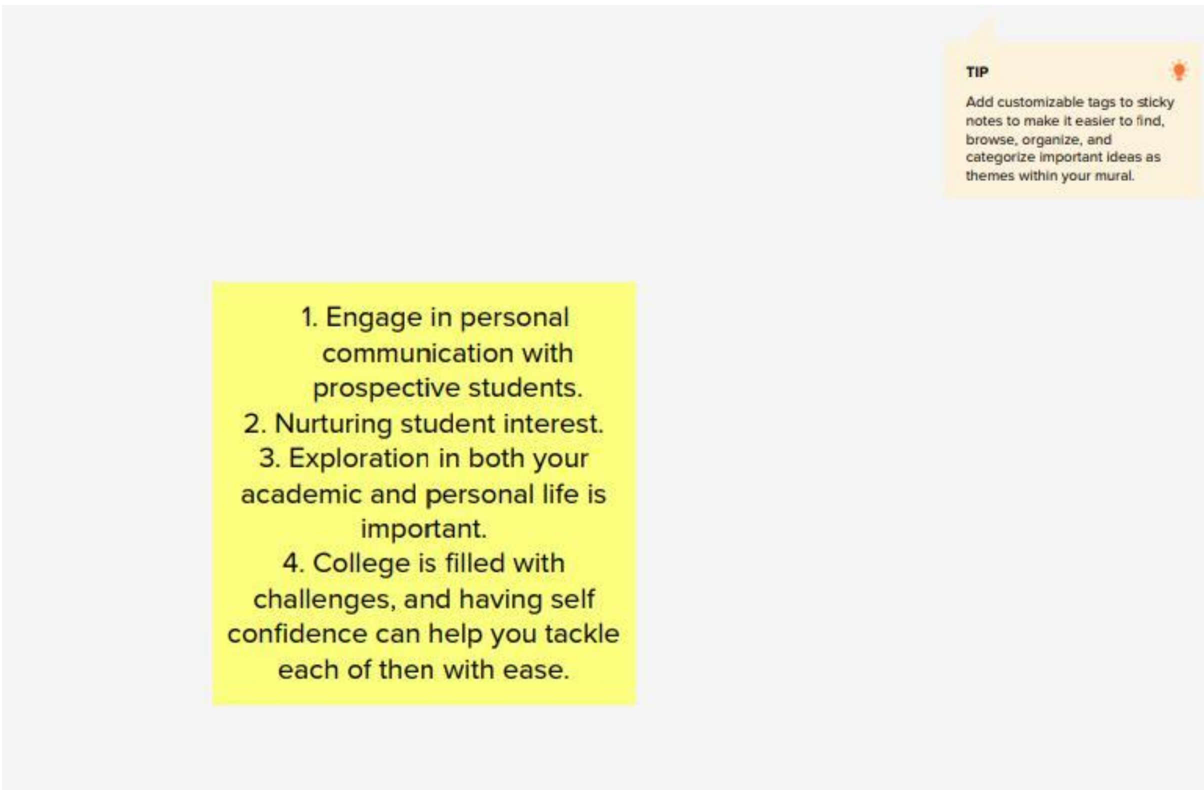
Students do not have access to a lot of quality work in making their own decisions about higher studies.

.

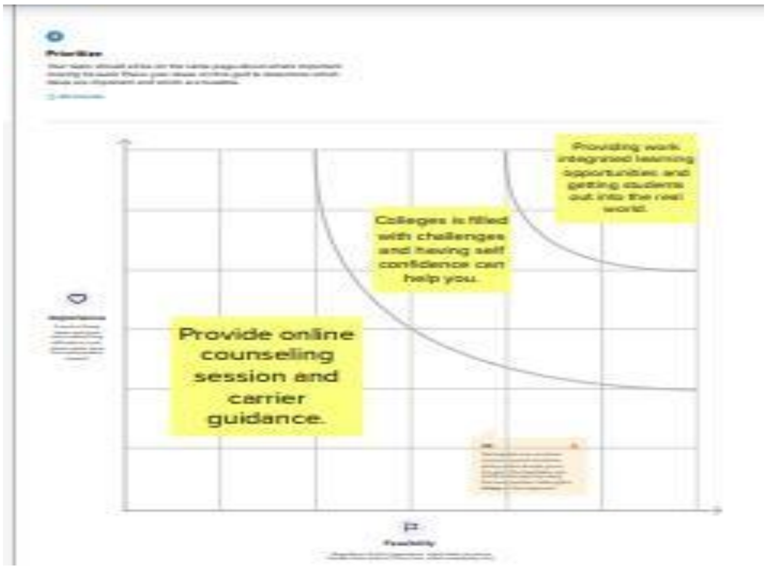
BRAINSTORM

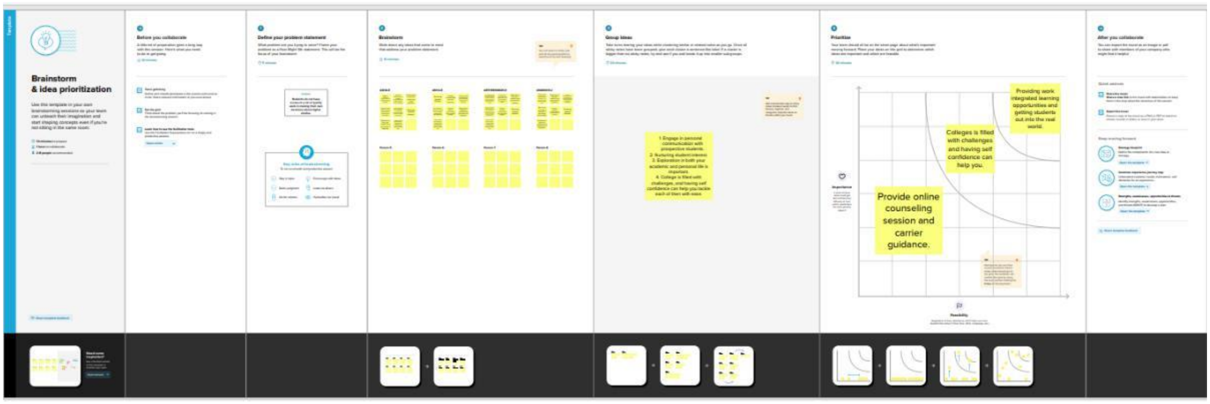


GROUP IDEAS



PRIORITIZE


Providing work integrated learning opportunities and getting students out into the real world.
Colleges is filled with challenges and having self confidence can help you.
Provide online counseling session and carrier guidance.


Brainstorm & idea prioritization
Providing work integrated learning opportunities and getting students out into the real world.
Colleges is filled with challenges and having self confidence can help you.
Provide online counseling session and carrier guidance.

RESULT:

- Universities are essential to provide learning spaces for students to plan and implement their ideas to contribute to achieve sustainable development goals and to integrate these issues into curricula and extra-curricular activities.

- Through quality education, people acquire the ability to listen, critically reflect about reality, and make informed choices about their life.

- It also provide knowledge about the latest technology used in developing the application that will be great demand in future This will provide better opportunities and guidance in future in developing projects independently.

ADVANTAGES OF ADMISSION PREDICT:

- The university education exposes students to new research and technology.
- Studying at university encourage creative and independent thought.
- Studying is the most basic knowledge that is to be acquired by every individual to learn various other things.
- A university education will help the student succeed in today's workforce and establish an enjoyable career of their choice.

DISADVANTAGES OF ADMISSION PREDICT:

- The main disadvantage of universities is the lack of individual attention. Classes may have more than a hundred students, making it difficult to stand out. For the same reason, It's difficult to find student opportunities for college.

- An unprofessional and non-standard education system may also cause wastage of time and money.
- Sometimes, brilliant student get bored because of the long tenure of academic sessions.
- It does not follow a proper schedule or a timespan.

APPLICATIONS FOR ADMISSION PREDICT:

- The university education function is to train people, and the university is a place to do just that. The special characteristics of university education are mainly manifested in the specificity of tasks.
- It is right to note that university education is institutionalized education and has a strict organizational structure and system.
- One of the aims of education is to have an influence on people’s purpose, organization, and planning. University education embodies all the characteristics of education.

- Nowadays studies in university are the huge improvement of the youngsters for their future and this is popular in every country.
- University graduates gain professional qualifications that are recognised and respected worldwide.
- University life exposes students to other culture and background.
- The ability to accurately predict the chances of university admission can help students make more informed decisions about which universities to apply to, increasing their chances of being admitted and ultimately gaining access to higher education.

CONCLUSION:

- The future university system which capable of storing university resources such as students and staff of the university and their relationship was implemented.
- The system supports different platforms and different languages.

- It is easy to track the relations of students and courses they have taken, courses teacher they are given by using the friendly interface of the system.

FUTURE SCOPE:

- The future scope of university for the student creating of new knowledge and presentation of past experience.
- Future university scope helps students come out of their weakness into their strength. Campus life is beyond the infrastructural and academic training programs. It enriches students with a once in a life time opportunity to live through the experiences that otherwise is earned very hard.
- Students are given the chance to travel and experience life overseas through study abroad programs.
- A college education teaches discipline to a student. They understand the importance of go through a comprehensive learning.

Attending a college paves the way for a better career.

- The future universities have stable educational places, stable educational objects, and stable educational contents, as well as stable educational order and so on. This kind of stability in universities is very conductive to personal development.

APPENDIX:

Milestone 2: Data Collection & Preparation

Importing the libraries

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

```
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

```
data.head()
```

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

```
data=data.rename(columns = {'Chance of Admit ':'Chance of Admi
t'})
```

Milestone 3: Exploratory Data Analysis

```
data.describe
```

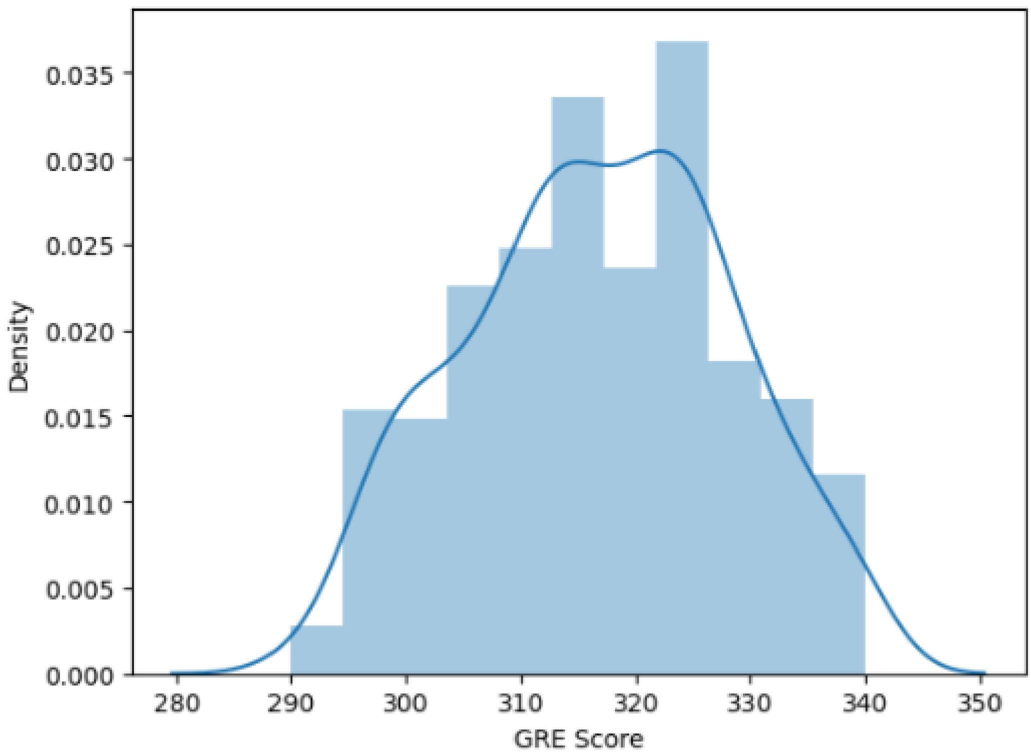
```
<bound method NDFrame.describe of      Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  \
0             1        337          118                  4  4.5   4.5  9.65
1             2        324          107                  4  4.0   4.5  8.87
2             3        316          104                  3  3.0   3.5  8.00
3             4        322          110                  3  3.5   2.5  8.67
4             5        314          103                  2  2.0   3.0  8.21
..          ...        ...          ...                ...  ...   ...   ...
395         396        324          110                  3  3.5   3.5  9.04
396         397        325          107                  3  3.0   3.5  9.11
397         398        330          116                  4  5.0   4.5  9.45
398         399        312          103                  3  3.5   4.0  8.78
399         400        333          117                  4  5.0   4.0  9.66

     Research  Chance of Admit
0           1             0.92
1           1             0.76
2           1             0.72
3           1             0.80
4           0             0.65
..        ...              ...
395         1             0.82
396         1             0.84
397         1             0.91
398         0             0.67
399         1             0.95

[400 rows x 9 columns]>
```

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

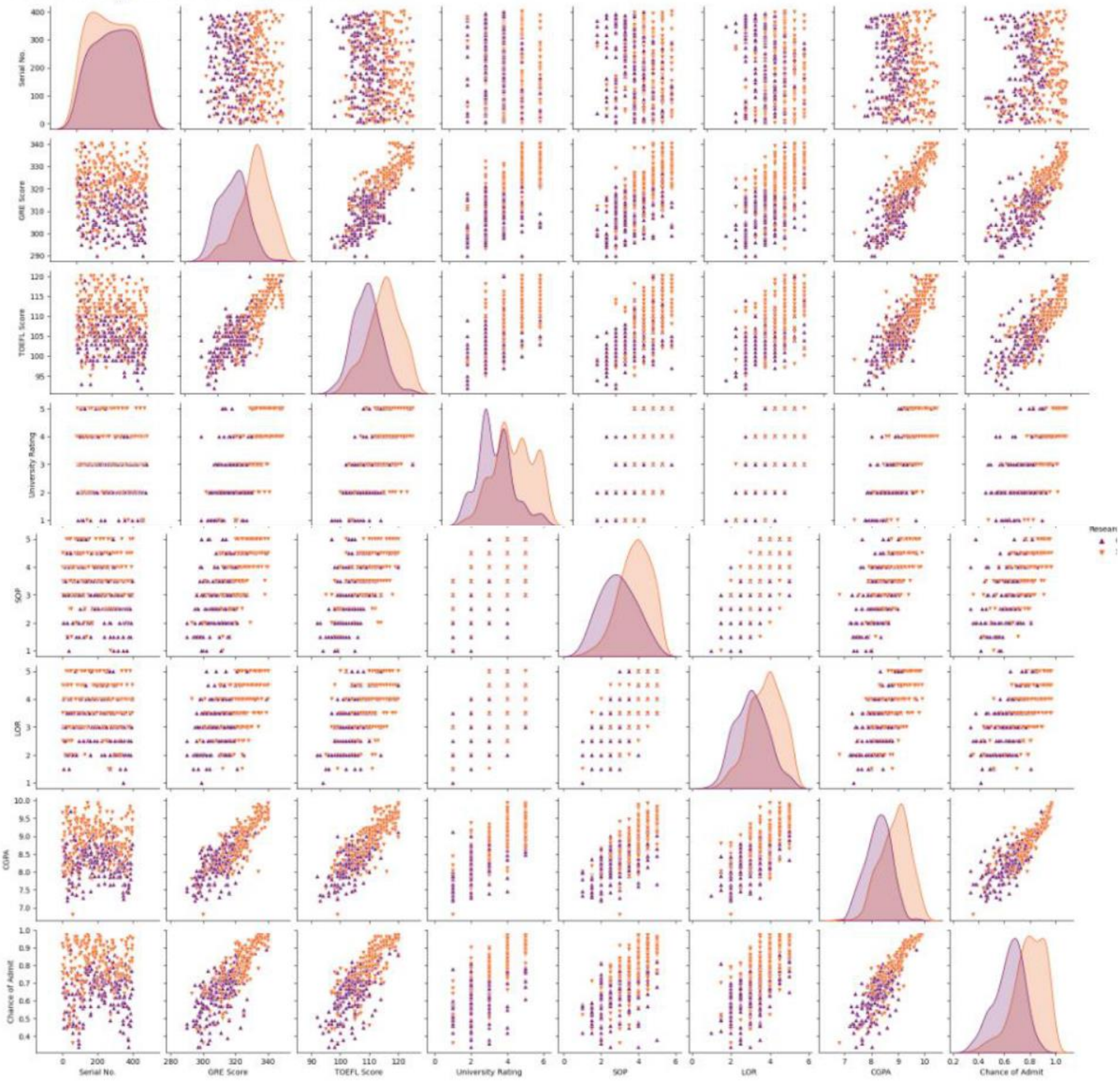
```
sns.distplot(data['GRE Score'])
<Axes: xlabel='GRE Score', ylabel='Density'>
```



```
sns.heatmap(data.corr())
```

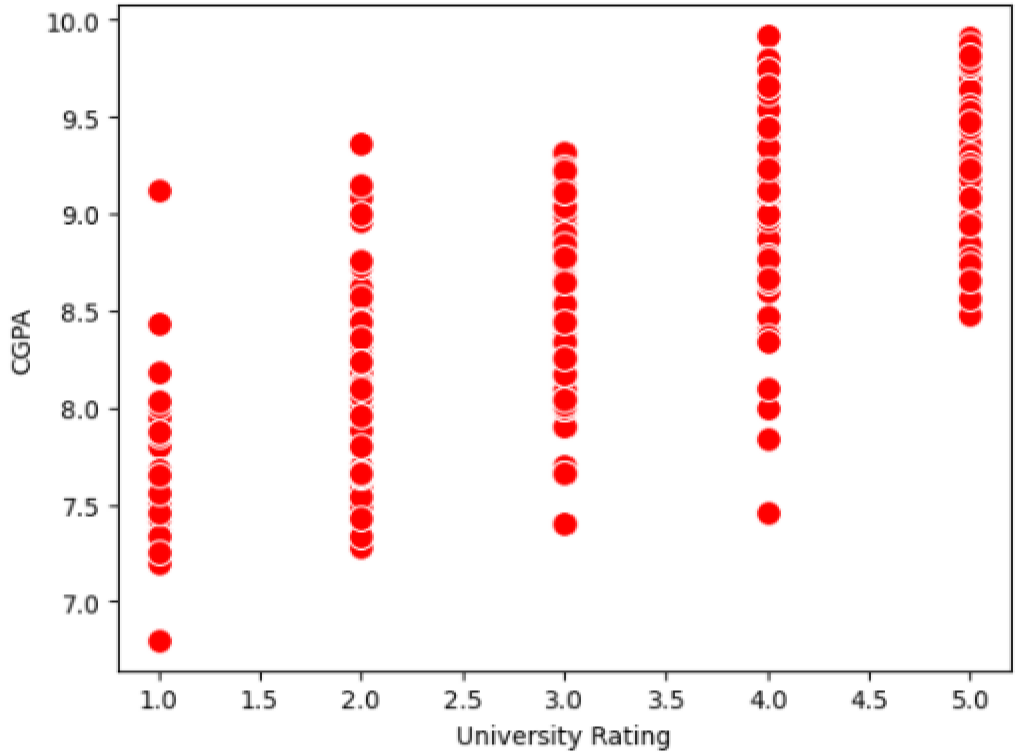
```
sns.pairplot(data=data,hue='Research',markers=["^","v"],palette='inferno')
```

```
<seaborn.axisgrid.PairGrid at 0x7fc066116580>
```

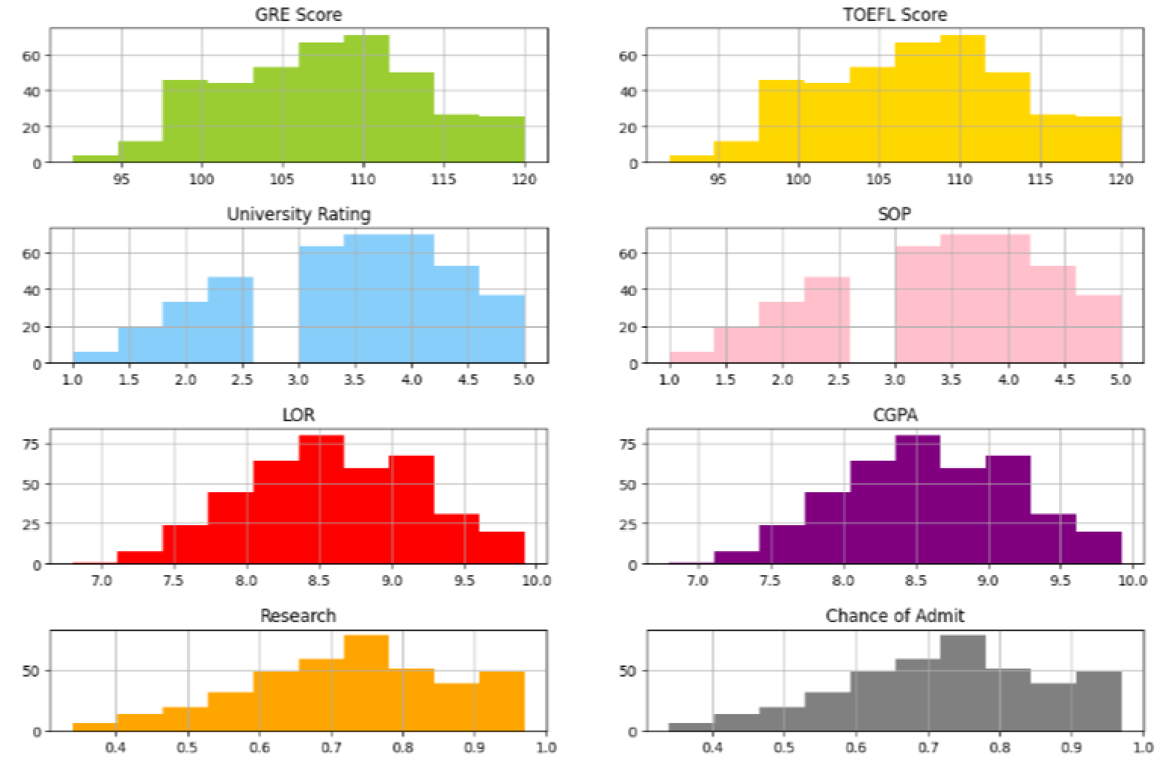


```
sns.scatterplot(x='University
Rating',y='CGPA',data=data,color ='Red', s=100)
```

```
<Axes: xlabel='University Rating', ylabel='CGPA'>
```




```
category = ['GRE Score','TOEFL Score','University Rating','SOP
','LOR','CGPA','Research','Chance of Admit']
color = ['Yellowgreen','gold','lightskyblue','pink','red','pur
ple','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()
```




```
from sklearn.preprocessing import
MinMaxScaler sc = MinMaxScaler()


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

```
array([[  1.  , 337.  , 118.  , ...,   4.5 ,   4.5 ,   9.65],
       [  2.  , 324.  , 107.  , ...,   4.  ,   4.5 ,   8.87],
       [  3.  , 316.  , 104.  , ...,   3.  ,   3.5 ,   8.  ],
       ...,
       [398.  , 330.  , 116.  , ...,   5.  ,   4.5 ,   9.45],
       [399.  , 312.  , 103.  , ...,   3.5 ,   4.  ,   8.78],
       [400.  , 333.  , 117.  , ...,   5.  ,   4.  ,   9.66]])
```

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

```
array([[1. , 0.92],
       [1. , 0.76],
       [1. , 0.72],
       [1. , 0.8 ],
       [0. , 0.65],
       [1. , 0.9 ],
       [1. , 0.75],
       [0. , 0.68],
       [0. , 0.5 ],
```

```
[0. , 0.45],
[1. , 0.52],
[1. , 0.84],
[1. , 0.78],
[1. , 0.62],
[1. , 0.61],
[0. , 0.54],
[0. , 0.66],
[1. , 0.65],
[0. , 0.63],
[0. , 0.62],
[1. , 0.64],
[0. , 0.7 ],
[1. , 0.94],
[1. , 0.95],
[1. , 0.97],
[1. , 0.94],
[0. , 0.76],
[1. , 0.44],
[0. , 0.46],
[0. , 0.54],
[1. , 0.65],
[1. , 0.74],
[1. , 0.91],
[1. , 0.9 ],
[1. , 0.94],
[1. , 0.88],
[0. , 0.64],
[0. , 0.58],
[0. , 0.52],
[0. , 0.48],
[1. , 0.46],
[1. , 0.49],
[1. , 0.53],
[0. , 0.87],
[1. , 0.91],
[1. , 0.88],
[1. , 0.86],
[0. , 0.89],
[1. , 0.82],
[1. , 0.78],
[1. , 0.76],
[1. , 0.56],
[1. , 0.78],
[1. , 0.72],
[0. , 0.7 ],
[0. , 0.64],
[0. , 0.64],
[0. , 0.46],
[1. , 0.36],
```

[0. , 0.42],
[0. , 0.48],
[0. , 0.47],
[1. , 0.54],
[1. , 0.56],
[0. , 0.52],
[0. , 0.55],
[0. , 0.61],
[1. , 0.57],
[1. , 0.68],
[1. , 0.78],
[1. , 0.94],
[1. , 0.96],
[1. , 0.93],
[1. , 0.84],
[0. , 0.74],
[1. , 0.72],
[1. , 0.74],
[0. , 0.64],
[1. , 0.44],
[0. , 0.46],
[1. , 0.5],
[1. , 0.96],
[1. , 0.92],
[1. , 0.92],
[1. , 0.94],
[0. , 0.76],
[0. , 0.72],
[0. , 0.66],
[0. , 0.64],
[1. , 0.74],
[1. , 0.64],
[0. , 0.38],
[0. , 0.34],
[1. , 0.44],
[0. , 0.36],
[0. , 0.42],
[0. , 0.48],
[1. , 0.86],
[1. , 0.9],
[1. , 0.79],
[1. , 0.71],
[0. , 0.64],
[0. , 0.62],
[0. , 0.57],
[1. , 0.74],
[1. , 0.69],
[1. , 0.87],
[1. , 0.91],
[1. , 0.93],

[0. , 0.68],
[0. , 0.61],
[1. , 0.69],
[1. , 0.62],
[0. , 0.72],
[1. , 0.59],
[1. , 0.66],
[0. , 0.56],
[0. , 0.45],
[0. , 0.47],
[1. , 0.71],
[1. , 0.94],
[1. , 0.94],
[0. , 0.57],
[0. , 0.61],
[0. , 0.57],
[1. , 0.64],
[1. , 0.85],
[1. , 0.78],
[1. , 0.84],
[1. , 0.92],
[1. , 0.96],
[0. , 0.77],
[0. , 0.71],
[0. , 0.79],
[1. , 0.89],
[1. , 0.82],
[0. , 0.76],
[1. , 0.71],
[1. , 0.8],
[0. , 0.78],
[1. , 0.84],
[1. , 0.9],
[1. , 0.92],
[1. , 0.97],
[1. , 0.8],
[1. , 0.81],
[0. , 0.75],
[1. , 0.83],
[1. , 0.96],
[1. , 0.79],
[1. , 0.93],
[1. , 0.94],
[1. , 0.86],
[0. , 0.79],
[0. , 0.8],
[0. , 0.77],
[0. , 0.7],
[0. , 0.65],
[0. , 0.61],

[0. , 0.52],
[0. , 0.57],
[0. , 0.53],
[0. , 0.67],
[0. , 0.68],
[1. , 0.81],
[0. , 0.78],
[0. , 0.65],
[0. , 0.64],
[1. , 0.64],
[0. , 0.65],
[1. , 0.68],
[1. , 0.89],
[1. , 0.86],
[1. , 0.89],
[1. , 0.87],
[1. , 0.85],
[1. , 0.9],
[0. , 0.82],
[0. , 0.72],
[0. , 0.73],
[0. , 0.71],
[0. , 0.71],
[0. , 0.68],
[0. , 0.75],
[0. , 0.72],
[1. , 0.89],
[1. , 0.84],
[1. , 0.93],
[1. , 0.93],
[1. , 0.88],
[1. , 0.9],
[1. , 0.87],
[1. , 0.86],
[1. , 0.94],
[0. , 0.77],
[1. , 0.78],
[0. , 0.73],
[0. , 0.73],
[0. , 0.7],
[0. , 0.72],
[1. , 0.73],
[1. , 0.72],
[1. , 0.97],
[1. , 0.97],
[0. , 0.69],
[0. , 0.57],
[0. , 0.63],
[1. , 0.66],
[0. , 0.64],

[1. , 0.68],
[1. , 0.79],
[1. , 0.82],
[1. , 0.95],
[1. , 0.96],
[1. , 0.94],
[1. , 0.93],
[1. , 0.91],
[1. , 0.85],
[1. , 0.84],
[0. , 0.74],
[0. , 0.76],
[0. , 0.75],
[0. , 0.76],
[0. , 0.71],
[0. , 0.67],
[0. , 0.61],
[0. , 0.63],
[0. , 0.64],
[0. , 0.71],
[1. , 0.82],
[0. , 0.73],
[1. , 0.74],
[0. , 0.69],
[0. , 0.64],
[1. , 0.91],
[1. , 0.88],
[1. , 0.85],
[1. , 0.86],
[0. , 0.7],
[0. , 0.59],
[0. , 0.6],
[0. , 0.65],
[1. , 0.7],
[1. , 0.76],
[0. , 0.63],
[1. , 0.81],
[0. , 0.72],
[0. , 0.71],
[1. , 0.8],
[1. , 0.77],
[1. , 0.74],
[0. , 0.7],
[1. , 0.71],
[1. , 0.93],
[0. , 0.85],
[0. , 0.79],
[0. , 0.76],
[1. , 0.78],
[1. , 0.77],

[1. , 0.9],
[1. , 0.87],
[0. , 0.71],
[1. , 0.7],
[1. , 0.7],
[1. , 0.75],
[0. , 0.71],
[0. , 0.72],
[1. , 0.73],
[0. , 0.83],
[0. , 0.77],
[1. , 0.72],
[0. , 0.54],
[0. , 0.49],
[1. , 0.52],
[0. , 0.58],
[1. , 0.78],
[1. , 0.89],
[0. , 0.7],
[0. , 0.66],
[0. , 0.67],
[1. , 0.68],
[1. , 0.8],
[1. , 0.81],
[1. , 0.8],
[1. , 0.94],
[1. , 0.93],
[1. , 0.92],
[1. , 0.89],
[0. , 0.82],
[0. , 0.79],
[0. , 0.58],
[0. , 0.56],
[0. , 0.56],
[1. , 0.64],
[1. , 0.61],
[0. , 0.68],
[0. , 0.76],
[0. , 0.86],
[1. , 0.9],
[0. , 0.71],
[0. , 0.62],
[0. , 0.66],
[1. , 0.65],
[1. , 0.73],
[0. , 0.62],
[1. , 0.74],
[1. , 0.79],
[1. , 0.8],
[0. , 0.69],

[0. , 0.7],
[1. , 0.76],
[1. , 0.84],
[1. , 0.78],
[0. , 0.67],
[0. , 0.66],
[0. , 0.65],
[0. , 0.54],
[0. , 0.58],
[1. , 0.79],
[1. , 0.8],
[1. , 0.75],
[1. , 0.73],
[0. , 0.72],
[0. , 0.62],
[0. , 0.67],
[1. , 0.81],
[0. , 0.63],
[0. , 0.69],
[1. , 0.8],
[0. , 0.43],
[1. , 0.8],
[1. , 0.73],
[1. , 0.75],
[1. , 0.71],
[1. , 0.73],
[1. , 0.83],
[0. , 0.72],
[1. , 0.94],
[1. , 0.81],
[1. , 0.81],
[1. , 0.75],
[1. , 0.79],
[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],
[0. , 0.7],

```
       [0. , 0.81],
       [1. , 0.85],
       [1. , 0.93],
       [1. , 0.91],
       [0. , 0.69],
       [1. , 0.77],
       [1. , 0.86],
       [1. , 0.74],
       [0. , 0.57],
       [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],
       [0. , 0.77],
       [1. , 0.89],
       [1. , 0.82],
       [1. , 0.84],
       [1. , 0.91],
       [0. , 0.67],
       [1. , 0.95]])
```

```
x=sc.fit_transform(x)
x
```

```
array([[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]])
```

```
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)
y_train
y_test=(y_test>0.5)
y_test
```

```
array([[False, True], [False, True], [ True, True], [False, True], [
True, True], [ True, True], [False, True], [False, False], [ True,
True], [False, True], [False, True], [ True, True], [ True, True], [
True, True], [False, False], [False, True], [ True, True], [False,
True], [False, True], [False, True], [False, True], [False, True], [
True, True], [ True, True], [False, True], [ True, True], [False,
True], [ True, True], [False, True], [ True, True], [ True, True], [
True, True], [False, False], [ True, True], [False, True], [False,
True], [False, True], [False, True], [False, True], [False, True], [
True, True], [False, True], [ True, True], [ True, True], [ True,
True], [ True, True], [False, True], [False, True], [False, True], [
True, True], [False, True], [ True, True], [ True, True], [False,
True], [ True, True], [ True, True], [False, True], [ True, True],
[False, True], [False, False], [False, False], [ True, True], [False,
True], [False, False], [False, True], [False, True], [False, True], [
True, True], [False, False], [False, True], [False, True], [False,
True], [ True, True], [False, True], [False, True], [ True, True], [
True, True], [ True, True], [False, True], [False, True], [ True,
True], [ True, True], [False, False], [ True, True], [ True, True], [
True, True], [False, True], [ True, True], [ True, True], [False,
```

```
True], [False, True], [False, True], [ True, True], [ True, True],
[False, True], [False, True], [ True, True], [ True, True], [False,
True], [False, True], [False, False], [False, False], [False, True],
[False, False], [False, True], [ True, True], [False, False], [ True,
True], [ True, True], [ True, True], [False, True], [False, True],
[False, True], [False, True], [ True, True], [False, True], [False,
True], [ True, True], [ True, True], [False, True]])
```

Milestone 4:Model Building

```
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])
```

ANN Model

```
#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"
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 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
```

Testing the Model

```
model.compile(loss ='binary_crossentropy', optimizer = 'adam',
metrics = ['accuracy'])
```

```
model.fit(x_train, y_train, batch_size = 20, epochs = 100)
```

Epoch 1/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4530 - accuracy: 0.7661
Epoch 2/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4519 - accuracy: 0.7696
Epoch 3/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4519 - accuracy: 0.7696
Epoch 4/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4516 - accuracy: 0.7661
Epoch 5/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4511 - accuracy: 0.7661
Epoch 6/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4506 - accuracy: 0.7696
Epoch 7/100

14/14 [==============================] - 0s 2ms/step - loss: 0.4512 - accuracy:
0.7696
Epoch 8/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4507 - accuracy:
0.7732
Epoch 9/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4510 - accuracy:
0.7732
Epoch 10/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4497 - accuracy:
0.7696
Epoch 11/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4499 - accuracy:
0.7732
Epoch 12/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4490 - accuracy:
0.7696
Epoch 13/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4488 - accuracy:
0.7696
Epoch 14/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4489 - accuracy:
0.7732
Epoch 15/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4487 - accuracy:
0.7696
Epoch 16/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4486 - accuracy:
0.7696
Epoch 17/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4496 - accuracy:
0.7732
Epoch 18/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4478 - accuracy:
0.7732
Epoch 19/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4486 - accuracy:
0.7768
Epoch 20/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4523 - accuracy:
0.7696
Epoch 21/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4805 - accuracy:
0.7768
Epoch 22/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4893 - accuracy:
0.7696
Epoch 23/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4520 - accuracy:
0.7804

Epoch 24/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4490 - accuracy: 0.7732
Epoch 25/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4468 - accuracy: 0.7768
Epoch 26/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4489 - accuracy: 0.7768
Epoch 27/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4466 - accuracy: 0.7768
Epoch 28/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4467 - accuracy: 0.7732
Epoch 29/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4465 - accuracy: 0.7804
Epoch 30/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4458 - accuracy: 0.7804
Epoch 31/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4465 - accuracy: 0.7768
Epoch 32/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4453 - accuracy: 0.7768
Epoch 33/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4456 - accuracy: 0.7804
Epoch 34/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4453 - accuracy: 0.7768
Epoch 35/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4452 - accuracy: 0.7804
Epoch 36/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4449 - accuracy: 0.7804
Epoch 37/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4457 - accuracy: 0.7768
Epoch 38/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4454 - accuracy: 0.7804
Epoch 39/100
14/14 [==============================] - 0s 4ms/step - loss: 0.4447 - accuracy: 0.7804
Epoch 40/100

14/14 [==============================] - 0s 3ms/step - loss: 0.4442 - accuracy: 0.7768
Epoch 41/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4453 - accuracy: 0.7804
Epoch 42/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4448 - accuracy: 0.7804
Epoch 43/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4446 - accuracy: 0.7804
Epoch 44/100
14/14 [==============================] - 0s 4ms/step - loss: 0.4439 - accuracy: 0.7804
Epoch 45/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4437 - accuracy: 0.7768
Epoch 46/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4437 - accuracy: 0.7804
Epoch 47/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4441 - accuracy: 0.7804
Epoch 48/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4431 - accuracy: 0.7804
Epoch 49/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4429 - accuracy: 0.7804
Epoch 50/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4442 - accuracy: 0.7804
Epoch 51/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4450 - accuracy: 0.7768
Epoch 52/100
14/14 [==============================] - 0s 4ms/step - loss: 0.4448 - accuracy: 0.7804
Epoch 53/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4436 - accuracy: 0.7804
Epoch 54/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4426 - accuracy: 0.7804
Epoch 55/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4434 - accuracy: 0.7804
Epoch 56/100
14/14 [==============================] - 0s 4ms/step - loss: 0.4450 - accuracy: 0.7839

Epoch 57/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4429 - accuracy: 0.7804
Epoch 58/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4420 - accuracy: 0.7768
Epoch 59/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4436 - accuracy: 0.7768
Epoch 60/100
14/14 [==============================] - 0s 4ms/step - loss: 0.4444 - accuracy: 0.7839
Epoch 61/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4421 - accuracy: 0.7804
Epoch 62/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4424 - accuracy: 0.7804
Epoch 63/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4416 - accuracy: 0.7804
Epoch 64/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4412 - accuracy: 0.7804
Epoch 65/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4421 - accuracy: 0.7804
Epoch 66/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4414 - accuracy: 0.7804
Epoch 67/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4411 - accuracy: 0.7804
Epoch 68/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4406 - accuracy: 0.7804
Epoch 69/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4405 - accuracy: 0.7804
Epoch 70/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4415 - accuracy: 0.7804
Epoch 71/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4415 - accuracy: 0.7804
Epoch 72/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4423 - accuracy: 0.7804
Epoch 73/100

14/14 [==============================] - 0s 3ms/step - loss: 0.4397 - accuracy: 0.7804
Epoch 74/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4405 - accuracy: 0.7804
Epoch 75/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4406 - accuracy: 0.7804
Epoch 76/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4406 - accuracy: 0.7804
Epoch 77/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4394 - accuracy: 0.7804
Epoch 78/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4401 - accuracy: 0.7804
Epoch 79/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4396 - accuracy: 0.7804
Epoch 80/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4418 - accuracy: 0.7804
Epoch 81/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4413 - accuracy: 0.7839
Epoch 82/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4399 - accuracy: 0.7804
Epoch 83/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4431 - accuracy: 0.7804
Epoch 84/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4576 - accuracy: 0.7696
Epoch 85/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4458 - accuracy: 0.7804
Epoch 86/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4393 - accuracy: 0.7804
Epoch 87/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4406 - accuracy: 0.7804
Epoch 88/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4411 - accuracy: 0.7804
Epoch 89/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4400 - accuracy: 0.7804

Epoch 90/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4382 - accuracy: 0.7804
Epoch 91/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4387 - accuracy: 0.7839
Epoch 92/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4388 - accuracy: 0.7804
Epoch 93/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4386 - accuracy: 0.7804
Epoch 94/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4378 - accuracy: 0.7804
Epoch 95/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4376 - accuracy: 0.7804
Epoch 96/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4407 - accuracy: 0.7839
Epoch 97/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4413 - accuracy: 0.7839
Epoch 98/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4378 - accuracy: 0.7839
Epoch 99/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4473 - accuracy: 0.7732
Epoch 100/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4447 - accuracy: 0.7875
<keras.callbacks.History at 0x7fbff7eb8910>

```
from sklearn.metrics import accuracy_score

#make predictions on the training data
train_predictions = model.predict(x_train)

print(train_predictions)
```

```
9/9 [==============================] - 0s
2ms/step [[ 3.35384756e-02]
 [ 1.13783265e-02]
 [ 3.74864936e-02]
```

```
[-3.13801542e-02]
[ 4.53513861e-03]
[-2.53757183e-02]
[ 1.96127314e-02]
[-5.19028157e-02]
[ 3.66824679e-02]
[ 5.65473177e-02]
[-8.17151964e-02]
[-1.45236030e-04]
[ 9.31834430e-03]
[-5.15447371e-03]
[ 1.99613839e-01]
[ 3.39177251e-01]
[-2.48153880e-03]
[ 2.03071386e-02]
[ 3.19009684e-02]
[ 1.00787818e-01]
[-1.71384402e-02]
[ 7.57895261e-02]
[ 4.21526693e-02]
[ 7.29008764e-02]
[-1.13474680e-02]
[ 5.05380332e-02]
[ 3.35797742e-02]
[ 1.68121532e-02]
[ 1.48677118e-02]
[ 6.04356416e-02]
[ 8.96014497e-02]
[ 5.50781656e-03]
[ 5.27895093e-02]
[ 3.20668310e-01]
[ 9.07141566e-02]
[-6.92866147e-02]
[-4.70442325e-03]
[-2.75324900e-02]
[ 3.75954434e-03]
[ 2.12761492e-01]
[ 9.15854238e-03]
[ 1.14521116e-01]
[-7.41040334e-03]
[ 1.59121260e-01]
[-3.66416350e-02]
[ 3.58796343e-02]
[ 4.87731807e-02]
[ 6.02347404e-02]
[ 3.27577703e-02]
[ 1.83190480e-02]
[ 8.13199133e-02]
[ 3.20306420e-03]
[ 1.35713741e-01]
[-5.79888299e-02]
```

```
[ 4.19950783e-02]
[ 2.45544296e-02]
[ 1.91510215e-01]
[ 3.89609151e-02]
[ 1.33620882e-02]
[ 5.50619029e-02]
[ 6.81232065e-02]
[ 1.05126366e-01]
[-2.89431009e-02]
[-6.98045176e-03]
[ 1.64235801e-01]
[ 1.49639919e-02]
[ 3.64482701e-02]
[ 5.46331629e-02]
[ 7.80566037e-03]
[ 2.73623973e-01]
[ 3.48821841e-03]
[ 2.81939477e-01]
[ 1.33604892e-02]
[ 2.40258455e-01]
[-5.10098115e-02]
[ 1.25848055e-01]
[ 1.03069581e-02]
[-7.04625994e-02]
[-5.33122905e-02]
[ 6.24305829e-02]
[-4.34131175e-03]
[ 8.46153423e-02]
[ 3.27638090e-01]
[ 2.37920880e-03]
[-9.91694070e-03]
[ 1.67833984e-01]
[-1.89571828e-02]
[ 3.59194875e-02]
[ 9.41228122e-02]
[ 3.85422595e-02]
[ 1.94444314e-01]
[ 2.56926026e-02]
[ 1.62806332e-01]
[-8.69288817e-02]
[ 8.44136178e-02]
[ 2.02835679e-01]
[ 2.42933661e-01]
[ 7.42138550e-02]
[-4.04651463e-02]
[ 8.41594636e-02]
[ 1.08881488e-01]
[ 1.90772027e-01]
[ 2.03301653e-01]
[-5.06841838e-02]
[ 3.33104908e-01]
```

```
[-4.40696254e-04]
[ 9.00766253e-03]
[ 6.35541901e-02]
[ 1.08447820e-01]
[ 2.08852254e-02]
[ 8.65181983e-02]
[ 7.80767202e-02]
[ 1.00009944e-02]
[ 5.87079208e-03]
[ 1.07929949e-02]
[-1.09392554e-02]
[-5.86666055e-02]
[-8.73204768e-02]
[-6.04210235e-03]
[ 1.09846495e-01]
[-3.64387184e-02]
[ 6.75831735e-03]
[ 2.63651103e-01]
[-1.82572547e-02]
[ 9.60448086e-02]
[-3.93426931e-03]
[-1.21686589e-02]
[-3.78906801e-02]
[ 8.79697204e-02]
[ 5.67349568e-02]
[-1.17456615e-02]
[ 3.43428776e-02]
[-1.27173942e-02]
[ 4.42201123e-02]
[-1.72681957e-02]
[ 1.92847073e-01]
[-4.35950831e-02]
[-2.66535338e-02]
[-3.40775400e-03]
[ 5.79172447e-02]
[ 2.13473998e-02]
[-1.41676906e-02]
[ 3.35926004e-02]
[ 1.59407198e-01]
[-5.44588119e-02]
[ 2.39864830e-02]
[ 1.22865252e-02]
[ 5.96231297e-02]
[ 1.07612416e-01]
[ 3.62150446e-02]
[ 4.64832410e-03]
[ 2.62559280e-02]
[-3.82508561e-02]
[ 3.71838883e-02]
[-5.26814498e-02]
[-8.87332670e-03]
```

[6.82600737e-02]
[5.26095107e-02]
[8.59551430e-02]
[2.28213314e-02]
[2.28596106e-03]
[3.69029120e-02]
[2.48648271e-01]
[3.99437882e-02]
[5.49837276e-02]
[2.52578948e-02]
[-3.78139988e-02]
[1.04140572e-01]
[-1.16151124e-02]
[1.70800537e-02]
[6.75157532e-02]
[1.44945048e-02]
[-2.53426619e-02]
[1.13360167e-01]
[4.50515337e-02]
[6.30716458e-02]
[-7.36417063e-03]
[7.43151680e-02]
[-9.41486843e-03]
[3.32701653e-02]
[1.23541458e-02]
[6.53972402e-02]
[-9.53802764e-02]
[-5.13146296e-02]
[8.17883015e-03]
[7.15094507e-02]
[8.08577836e-02]
[4.66674156e-02]
[-7.27615505e-02]
[-1.84245408e-03]
[2.83840988e-02]
[3.35444361e-02]
[5.02580330e-02]
[-5.41930497e-02]
[9.49242711e-02]
[1.23189770e-01]
[1.13428667e-01]
[-5.17687760e-02]
[1.00491136e-01]
[2.01509073e-02]
[7.36704022e-02]
[-5.70447743e-02]
[6.82439730e-02]
[-3.62615176e-02]
[-5.23721613e-03]
[1.14745703e-02]
[-1.31747872e-03]

[6.94487020e-02]
[-2.44280044e-02]
[-1.41864084e-03]
[2.30216458e-02]
[-1.74401272e-02]
[6.15912527e-02]
[-8.16440806e-02]
[1.26375575e-02]
[5.70113026e-02]
[1.36384651e-01]
[7.20096081e-02]
[2.72023261e-01]
[-2.86854859e-02]
[-1.46190338e-02]
[7.95872957e-02]
[4.11990359e-02]
[7.81851783e-02]
[4.71645966e-03]
[-1.29803028e-02]
[1.47271678e-02]
[1.36752874e-01]
[2.88334228e-02]
[2.23067030e-02]
[-6.64119869e-02]
[1.22308627e-01]
[2.13306352e-01]
[2.09014937e-01]
[1.75749019e-01]
[8.43736529e-02]
[7.69469440e-02]
[-1.46185011e-02]
[-2.55990271e-02]
[1.03192627e-01]
[4.76190858e-02]
[-1.31208431e-02]
[1.51805803e-02]
[-1.97999775e-02]
[6.44813478e-02]
[1.37409166e-01]
[-3.52910981e-02]
[2.45893478e-01]
[-2.91608404e-02]
[4.55157273e-03]
[-7.36767054e-03]
[-5.93652204e-03]
[6.70890138e-02]
[1.09804533e-02]
[9.24665853e-02]
[-6.82122856e-02]
[6.50386978e-03]
[3.16592634e-01]

```
 [ 1.10128120e-01]
 [-4.66692448e-03]
 [-1.42946597e-02]
 [ 1.39721856e-01]
 [-5.00374138e-02]
 [ 1.27313718e-01]
 [-9.12095606e-02]
 [ 7.47198611e-03]
 [ 1.95730962e-02]
 [ 3.63375470e-02]
 [ 1.87365770e-01]
 [ 3.79888080e-02]
 [ 1.09042190e-01]
 [ 3.96643952e-02]
 [-4.31355610e-02]
 [ 2.16610748e-02]
 [ 1.06593966e-02]
 [ 5.62556088e-04]
 [ 1.82541981e-01]
 [ 5.96568435e-02]
 [-6.31931871e-02]
 [ 1.00867912e-01]
```

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

print(train_acc)
```

0.7839285731315613

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

print(test_acc)

0.7124999761581421
```

```
pred=model.predict(x_test)
pred = (pred>0.5)
pred
```

4/4 [==============================] - 0s 3ms/step
array([[True],
       [True],
       [True],
       [True],

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

[True],
[True],
[True],
[True],
[True],
[True],
[False],
[True],
[True],
[False],
[True],
[False],
[True],
[True],
[False],
[True],
[False],
[True],
[True],
[True],
[True],
[True],
[True],
[True],
[True],
[True],
[True],
[True],
[False],
[True],
[True],
[True],
[True],
[True],
[True],
[True],
[True],
[True],
[True],
[True],
[False],
[True],
[False],
[True],
[True],
[True],
[True],
[True],
[True],
[True],

```
       [ True],
       [ True],
       [False],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [False],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True]])
```

```
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, 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],
       [0, 1],
       [1, 1],
       [1, 1],
       [1, 1],
       [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],
[1, 1],
[1, 1],
[0, 0],
[1, 1],
[0, 1],
[0, 1],
[0, 1],
[0, 1],
[0, 1],
[0, 1],
[1, 1],
[0, 1],
[1, 1],
[1, 1],
[1, 1],
[1, 1],
[0, 1],
[0, 1],
[0, 1],
[1, 1],
[0, 1],
[1, 1],
[1, 1],
[0, 1],
[1, 1],
[1, 1],
[0, 1],
[1, 1],
[0, 1],
[0, 0],
[0, 0],
[1, 1],
[0, 1],
[0, 0],
[0, 1],
[0, 1],
[0, 1],
[1, 1],
[0, 0],
[0, 1],

```
[0, 1],
[0, 1],
[1, 1],
[0, 1],
[0, 1],
[1, 1],
[1, 1],
[1, 1],
[0, 1],
[0, 1],
[1, 1],
[1, 1],
[0, 0],
[1, 1],
[1, 1],
[1, 1],
[0, 1],
[1, 1],
[1, 1],
[0, 1],
[0, 1],
[0, 1],
[1, 1],
[1, 1],
[0, 1],
[0, 1],
[1, 1],
[1, 1],
[0, 1],
[0, 1],
[0, 0],
[0, 0],
[0, 1],
[0, 0],
[0, 1],
[1, 1],
[0, 0],
[1, 1],
[1, 1],
[1, 1],
[0, 1],
[0, 1],
[0, 1],
[0, 1],
[1, 1],
[0, 1],
[0, 1],
[1, 1],
[1, 1],
[0, 1]])
```

Milestone 5:

Performance Testing & Hyperparameter Tuning

```
def logreg(x_train,x_test,y_train,y_test):
  lr = LogisticRegression(random_state=0)
  lr.fit(x_train,y_train)
  y_lr_tr = lr.predict(x_train)
  print(accuracy_score(y_lr_tr,y_train))
  ypred_lr = lr.predict(x_test)
  print(accuracy_score(y_lr_tr,y_train))
  print("***Logistic Regression***")
  print("Confusion_Matrix")
  print("Classification Report")
  print(classification_report(y_test,ypred_lr))
```

```
from sklearn.metrics import
accuracy_score,recall_score,roc_au c_score,confusion_matrix
print("Accuracy score: %f" %(accuracy_score(y_test,y_pred) * 1
00))
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: 90.000000
Recall score : 99.074074
ROC score : 53.703704

[[   1  11]
 [   1 107]]
```

```
from sklearn.metrics import accuracy_score,recall_score,roc_au
c_score,confusion_matrix
print(classification_report(y_train,y_pred))
```

	precision	recall	f1-score	support
False	1.00	0.16	0.28	25
True	0.93	1.00	0.97	295
accuracy			0.93	320
macro avg	0.97	0.58	0.62	320
weighted avg	0.94	0.93	0.91	320

```
from sklearn.metrics import accuracy_score,recall_score,roc_au
c_score,confusion_matrix
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
False	0.00	0.00	0.00	10
True	0.88	1.00	0.93	70
accuracy			0.88	80
macro avg	0.44	0.50	0.47	80
weighted avg	0.77	0.88	0.82	80

Milestone 6: Model Deployment

```
# save the model in HDF5
format model.save('model.h5')


import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
app = Flask(__name__)
from tensorflow.keras.models import load_model


model = load_model('model.h5')
```

```python
@app.route('/')
def home():
  return render_template('Demo2.html')
```