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

PROJECT RECORD TEMPLATE

CHAPTER	TITLE	PAGE.NO.
1	INTRODUCTION 1.1OVERVIEW 1.2PURPOSE	
2	PROBLEM DEFINITION & DESIGN THINKING 2.1 EMPATHY MAP 2.2 IDEATION & BRAINSTOMING MAP	
3	RESULT	
4	ADVANTAGES & DISADVANTAGES	
5	APPLICATIONS	
6	CONCLUSION	
7	FUTURE SCOPE	
8	APPENDIX 8.1 SOURCE CODE	

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The project (Intelligent Admission: The future of university decision making with machine learning) to build a machine learning model that can predict In recent years, the use of machine learning in the admissions process for universities has become an increasingly popular topic. With the vast amounts of data that universities collect on applicants, machine learning algorithms have the potential to analyze this data and make more informed decisions about which students to admit. This approach, known as intelligent admissions, has the potential to improve the fairness, efficiency, and effectiveness of the admissions process.

By using machine learning algorithms, universities can identify patterns and trends in data that might not be immediately apparent to human admissions officers. This can help to eliminate biases that might exist in the current admissions process and allow universities to consider a wider range of factors when making decisions about which applicants to admit.

1.2 PURPOSE

The purpose of this project is to develop a machine learning model the purpose of the "Intelligent Admissions: The Future of University Decision Making with Machine Learning" project is to explore the

potential benefits of using machine learning algorithms in the admissions process for universities. The project aims to build a machine learning model that can analyze the vast amounts of data collected by universities on applicants and make more informed decisions about which students to admit.

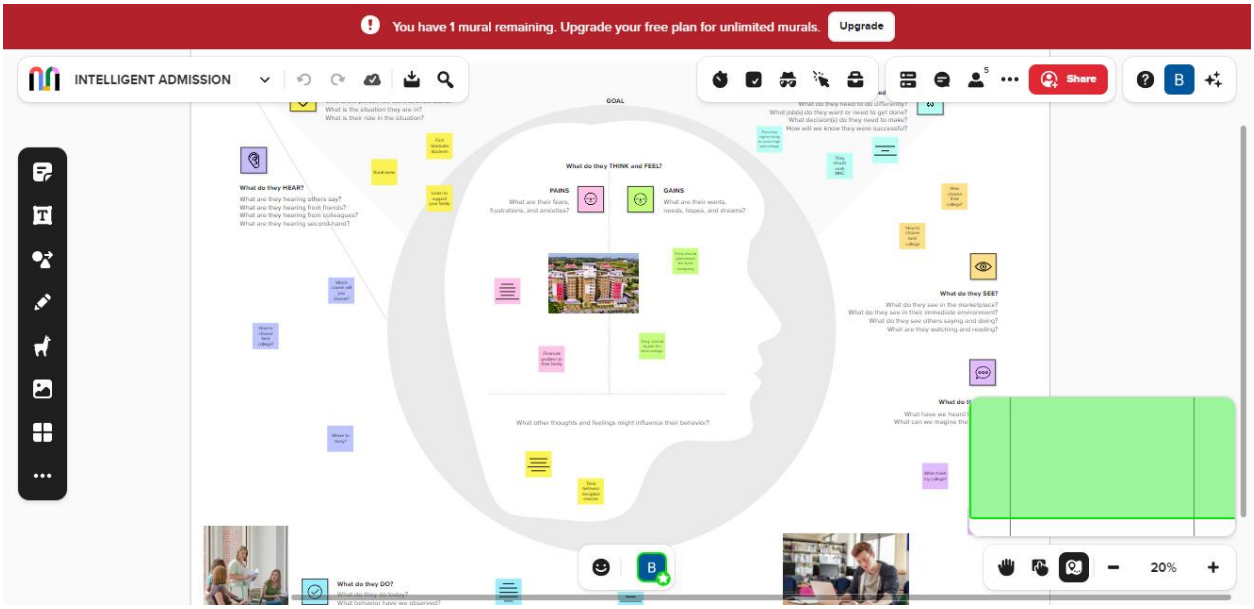
The project seeks to address some of the limitations of the current admissions process by eliminating biases that might exist and allowing universities to consider a wider range of factors when making decisions about admissions. Additionally, the project aims to improve the efficiency and effectiveness of the admissions process by automating certain aspects of the process, reducing the time and resources needed to make admissions decisions.

Overall, the purpose of the project is to explore how machine learning can be used to create a fairer, more efficient, and more effective admissions process for universities.

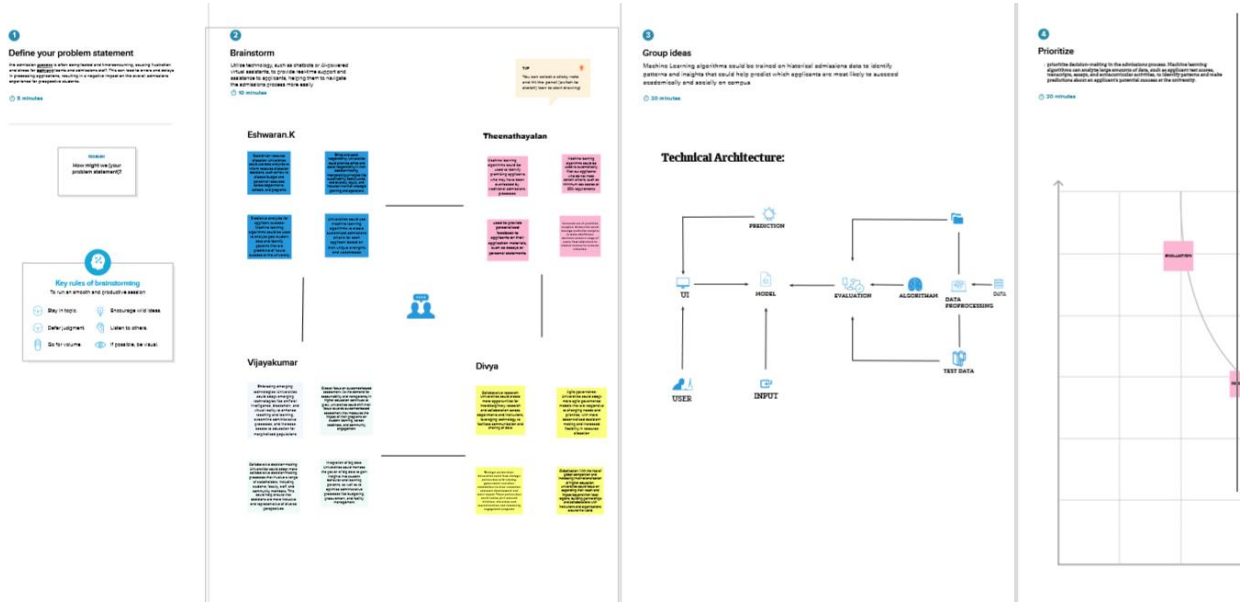
CHAPTER 2

PROBLEM DEFINITION & DESIGN THINKING

2.1 PROBLEM DEFINITION:



2.2 IDEATION & BRAINSTOMING MAP:



CHAPTER 3

RESULT

Result 1:

- Import all the tools we need.
- All needed tools import successful.

Result 2:

```
<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
```

Result 3:

```
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
```

Result 4:

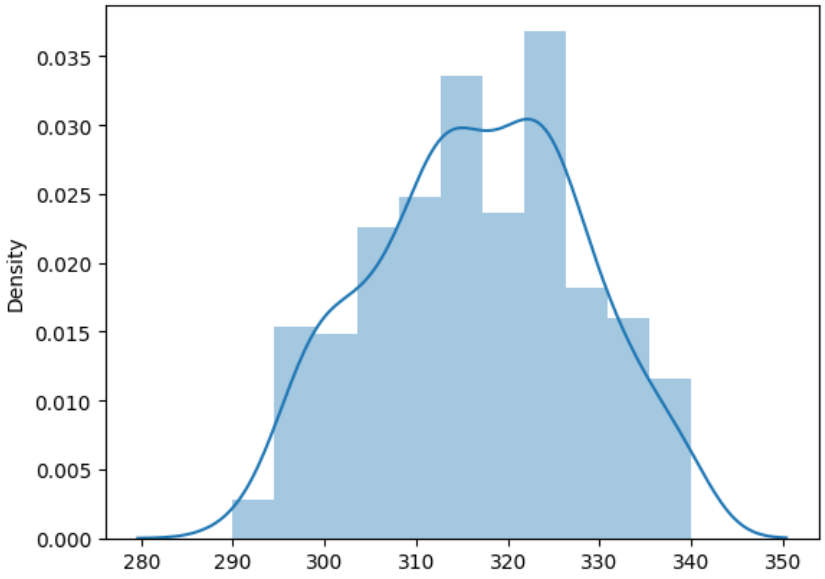
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         400 non-null    int64  
 1   GRE Score          400 non-null    int64  
 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
```

Result 5:

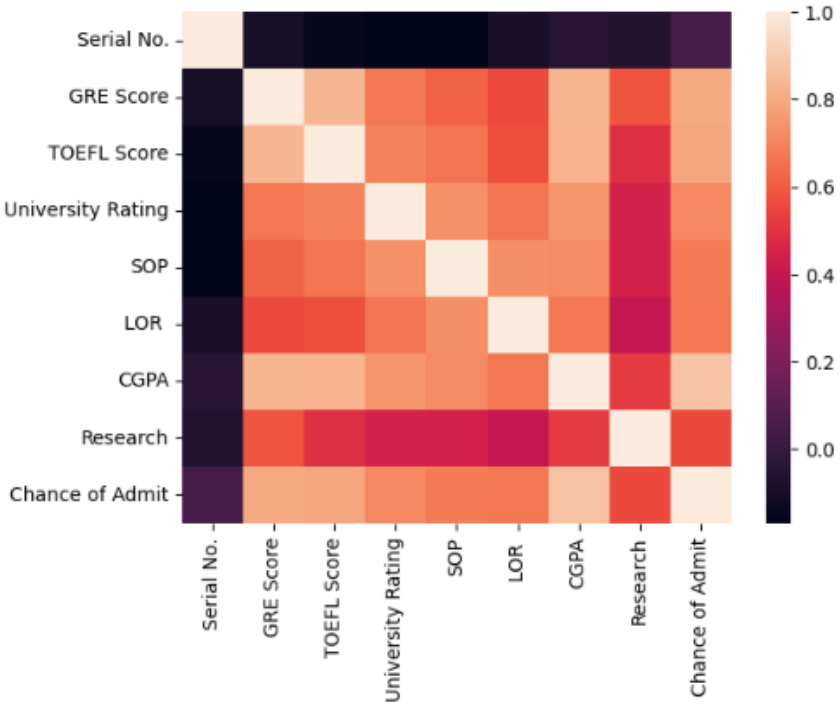
	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	200.500000	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.547500	0.724350
std	115.614301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

Result 6:

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

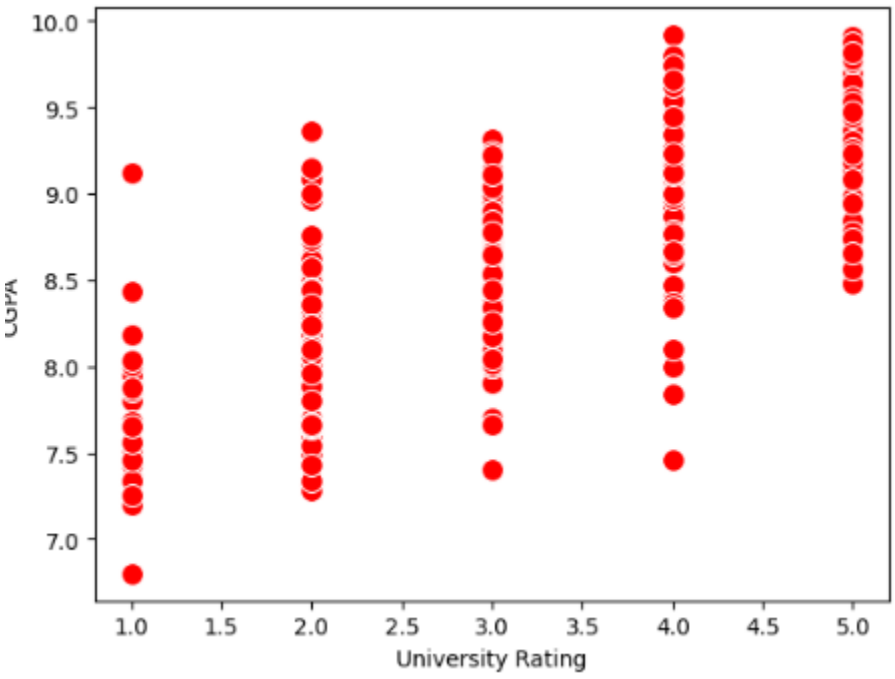


Result 7:



Result 8:

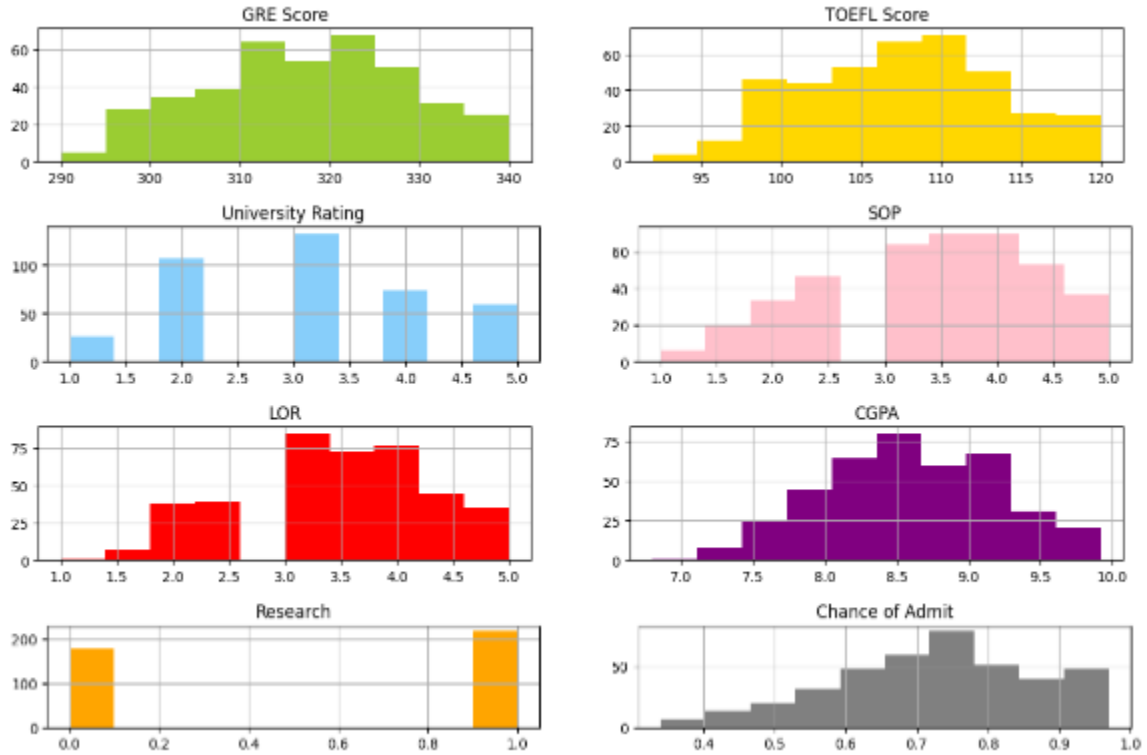
Result 9:



Result 10:

```
<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
```

Result 11:



Result 12:

```
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]])
```

Result 13:

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

Result 14:

```
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]])
```

Result 15:

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

Result 16:

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

Result 17:

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

Result 17:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 7)                 56

 dense_1 (Dense)             (None, 7)                 56

 dense_2 (Dense)             (None, 7)                 56

=================================================================
Total params: 168
Trainable params: 168
Non-trainable params: 0
_________________________________________________________________
```

Result 17:

```
Epoch 1/100
14/14 [==============================] - 1s 4ms/step - loss: 3.0168 - accuracy: 0.2893
Epoch 2/100
14/14 [==============================] - 0s 4ms/step - loss: 3.0140 - accuracy: 0.3071
Epoch 3/100
14/14 [==============================] - 0s 5ms/step - loss: 3.0116 - accuracy: 0.3179
Epoch 4/100
14/14 [==============================] - 0s 3ms/step - loss: 3.0222 - accuracy: 0.3214
Epoch 5/100
14/14 [==============================] - 0s 3ms/step - loss: 3.0202 - accuracy: 0.3321
Epoch 6/100
14/14 [==============================] - 0s 3ms/step - loss: 3.0193 - accuracy: 0.3357
Epoch 7/100
14/14 [==============================] - 0s 4ms/step - loss: 3.0193 - accuracy: 0.3393
Epoch 8/100
14/14 [==============================] - 0s 4ms/step - loss: 3.0177 - accuracy: 0.3393
Epoch 9/100
14/14 [==============================] - 0s 3ms/step - loss: 3.0176 - accuracy: 0.3357
Epoch 10/100
14/14 [==============================] - 0s 4ms/step - loss: 3.0164 - accuracy: 0.3429
Epoch 11/100
14/14 [==============================] - 0s 3ms/step - loss: 3.0158 - accuracy: 0.3500
Epoch 12/100
14/14 [==============================] - 0s 4ms/step - loss: 3.0153 - accuracy: 0.3571
Epoch 13/100
14/14 [==============================] - 0s 3ms/step - loss: 3.0147 - accuracy: 0.3571
Epoch 14/100
14/14 [==============================] - 0s 8ms/step - loss: 3.0131 - accuracy: 0.3536
Epoch 15/100
14/14 [==============================] - 0s 12ms/step - loss: 3.0133 - accuracy: 0.3536
Epoch 16/100
14/14 [==============================] - 0s 7ms/step - loss: 3.0130 - accuracy: 0.3536
Epoch 17/100
14/14 [==============================] - 0s 7ms/step - loss: 3.0121 - accuracy: 0.3571
Epoch 18/100
14/14 [==============================] - 0s 9ms/step - loss: 3.0113 - accuracy: 0.3571
Epoch 19/100
14/14 [==============================] - 0s 8ms/step - loss: 3.0111 - accuracy: 0.3571
Epoch 20/100
14/14 [==============================] - 0s 9ms/step - loss: 3.0101 - accuracy: 0.3571
Epoch 21/100
14/14 [==============================] - 0s 7ms/step - loss: 3.0113 - accuracy: 0.3571
Epoch 22/100
14/14 [==============================] - 0s 5ms/step - loss: 3.0145 - accuracy: 0.3607
Epoch 23/100
14/14 [==============================] - 0s 2ms/step - loss: 3.0164 - accuracy: 0.3536
Epoch 24/100
14/14 [==============================] - 0s 3ms/step - loss: 3.0106 - accuracy: 0.3536
Epoch 25/100
14/14 [==============================] - 0s 2ms/step - loss: 3.0097 - accuracy: 0.3536
Epoch 26/100
14/14 [==============================] - 0s 3ms/step - loss: 3.0080 - accuracy: 0.3571
Epoch 27/100
14/14 [==============================] - 0s 2ms/step - loss: 3.0076 - accuracy: 0.3571
Epoch 28/100
14/14 [==============================] - 0s 3ms/step - loss: 3.0066 - accuracy: 0.3571
Epoch 29/100
14/14 [==============================] - 0s 2ms/step - loss: 3.0063 - accuracy: 0.3571
```

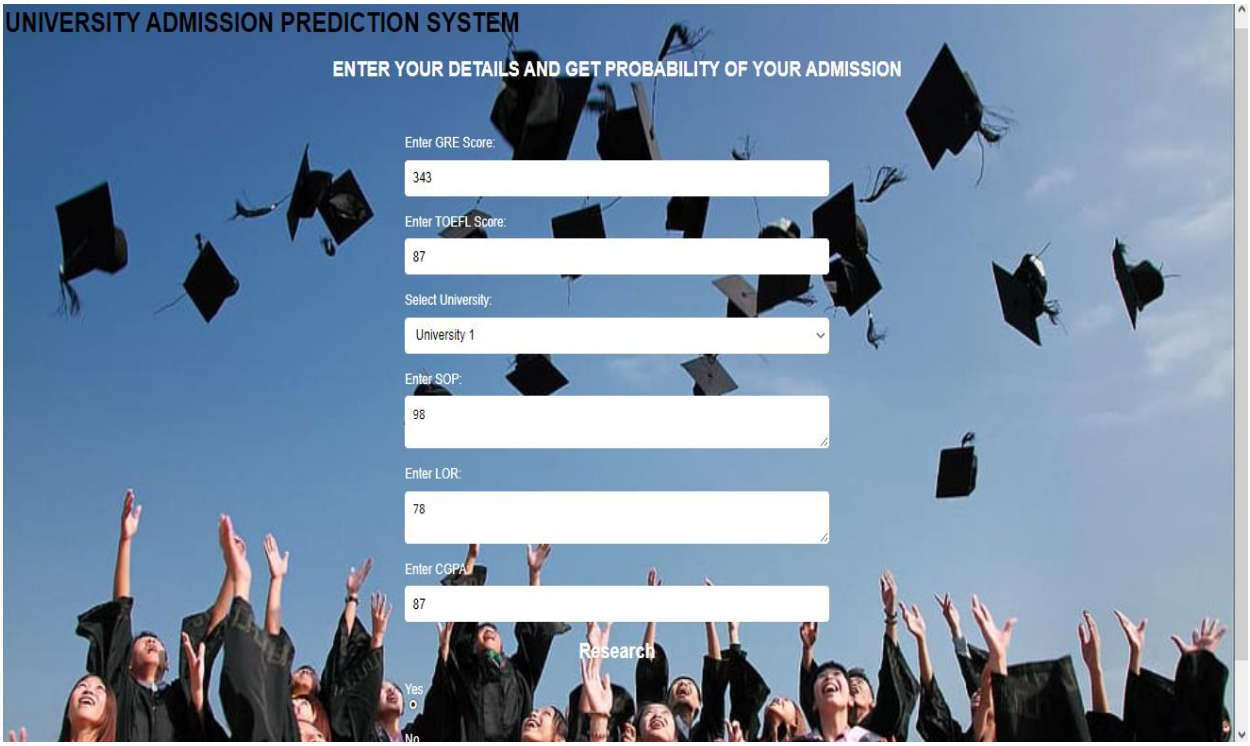
Result 18:

```
9/9 [==============================] - 0s 2ms/step
[[ 1.0459894  -0.5715048   1.0673568  ...  1.0661731  -0.41084728
   1.0874926 ]
 [ 0.91784036 -0.6004993   0.9204847  ...  0.9492643  -0.2988746
   0.9606166 ]
 [ 0.62901646 -0.38047814  0.6068774  ...  0.6049504  -0.23528156
   0.6087853 ]
 ...
 [ 1.0695648  -0.6080221   1.0918226  ...  1.0995339  -0.40376195
   1.1204951 ]
 [ 0.9804415  -0.5233968   0.99607295 ...  0.98854876 -0.39520746
   1.0080665 ]
 [ 0.80103683 -0.50781476  0.7938847  ...  0.8089706  -0.27560815
   0.81747675]]
```

Result 19:

```
Accuracy Score : 71.666667
Recall Score : 78.431373
ROC Score : 72.549020

Confusion Matrix
[[46 23]
 [11 40]]

Classification Report
              precision    recall  f1-score   support

       False       0.81      0.67      0.73        69
        True       0.63      0.78      0.70        51

    accuracy                           0.72       120
   macro avg       0.72      0.73      0.72       120
weighted avg       0.73      0.72      0.72       120
```

Result 20:



Result 21:

CHAPTER 4

ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- Efficient and Accurate: Intelligent admission can process large volumes of data quickly and accurately, helping universities to make more informed decisions and reducing of risk of human error.
- Machine learning algorithms can help eliminates biases and discrimination in the admission process, ensuring that candidates are based on their merit rather than their demographic characteristics.
- By admitting the most qualified candidates, University can improve students overcomes, such as graduation employment prospects.

By automating the admission process, University can save time and money on administrative tasks, freeing up resources to invest in other areas

DISADVANTAGES:

- Intelligent admission relies solely on data and algorithms to make admission decision, which can lead to a lack of personal touch and a failure to consider intangible factors that may be important process.
- Machine learning algorithms require large amounts of data to train and make accurate predictions, which may not always be available or accessible for some university.
- Although machine learning algorithms can help eliminate bias and discrimination, they can also perpetuate it if the data used o train them is biased or incomplete.
- Machine learning algorithms can be complex and difficult to understand, which may make it challenging for university to explain admission decisions to students and other stakeholders.

CHAPTER 5

APPLICATIONS

Machine learning is a subfield of artificial intelligence that involves the use of algorithms and statistical models to analyze data and make predictions. The application of machine learning techniques in university admissions can streamline the process, reduce bias, and improve the accuracy of decision-making.

Here are some potential applications of machine learning in university admissions:

- Predictive modeling: Machine learning algorithms can be used to develop predictive models that analyze various factors such as grades, test scores, extracurricular activities, and personal statements. These models can predict a student's likelihood of success at the university and help admissions officers make informed decisions.
- Bias reduction: Machine learning algorithms can reduce the impact of unconscious bias by removing identifying information such as gender, race, and ethnicity from applications. This can help ensure that admissions decisions are based solely on merit.
- Data analysis: Machine learning algorithms can analyze large volumes of data to identify patterns and trends. This can help admissions officers identify the characteristics of successful students and make informed decisions based on data-driven insights.
- Personalization: Machine learning algorithms can be used to personalize the admissions process by tailoring questions and feedback to individual applicants. This can help applicants feel more engaged with the process and improve their overall experience.

- Fraud detection: Machine learning algorithms can be used to detect fraudulent applications by analyzing patterns and anomalies in the data. This can help prevent admissions fraud and ensure the integrity of the admissions process.

Overall, the application of machine learning in university admissions has the potential to revolutionize the way universities make admissions decisions. By streamlining the process, reducing bias, and improving the accuracy of decision-making, machine learning can help universities select the best candidates and build a more diverse and successful student body

CHAPTER 6

CONCLUSION

Every year millions of students apply to universities to begin their educational life. Most of them don’t have proper resources, prior knowledge and are not cautions, which in turn created a lot problem as applying to the wrong university/college, which further wastes their time, money and energy.

With the help of our project, we have tried to help out such students who are finding difficulty in finding the right university for them. It is very important that a candidate should apply to colleges that he has a good chance of getting into, instead of applying to colleges that they may never get into. This will help in reduction of cost as students will be applying to only those university that they are highly likely to get into this will help in reduction to get into.

Our prepared models work to a satisfactory level of accuracy and may be of great assistance to such people. This is a project with good future, especially of our age group who want to pursue their education in their dream college.

CHAPTER 7

FUTURE SCOPE

Here are some potential enhancements that can be made in the future for the intelligent admission classification model:

- Feature selection and engineering: While the current model uses all the features available in the dataset, further analysis can be done to identify the most important features for accurate classification. Additionally, new features can be engineered from the existing ones to improve the model's performance.
- Hyper parameter tuning: The current model uses default hyper parameters for the decision tree classifier, but more advanced techniques such as grid search or random search can be used to optimize the hyper parameters for better performance.
- Ensemble learning: Ensemble learning techniques such as random forests or gradient boosting can be used to combine multiple decision tree classifiers to improve the overall accuracy and generalization of the model.
- Improved dataset: The current dataset only contains 200 samples, and collecting more data can help in building a more robust and accurate model.

Overall, there are many potential enhancements that can be made to the classification model, and future research can focus on exploring these areas to improve the accuracy and applicability of the model in settings.

.

CHAPTER 8

APPENDIX

8.1 SOURCE CODE

```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
import os # accessing directory structure
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns

%matplotlib inline

from  google.colab import files
import io

In [3]:
data=files.upload()
data=pd.read_csv('/content/Admission_Predict.csv')

In [6]:
data.info()
data.isnull().any()
data=data.rename(columns ={'Chance of Admit ': 'Chance of Admit','LO
R ':'LOR'})

In [10]:
data.info()
data.describe()
sns.distplot(data["GRE Score"])
```

```
<Axes: xlabel='GRE Score', ylabel='Density'>
ax = sns.heatmap(data.corr(), annot=False)
sns.pairplot(data=data, hue="Research", markers=["^", "v"],palette="inferno")
sns.scatterplot(x="University Rating",y="CGPA",data=data,color="Red",s=100)
data.info()
category=["GRE Score",'TOEFL Score','University Rating','SOP','LOR','CGPA','Research','Chance of Admit']
color=["yellowgreen","gold","lightskyblue",'pink','red','purple','orange','grey']

start=True
for i in np.arange(4):
fig=plt.figure(figsize=(14,8))
plt.subplot2grid((4,2),(i,0))
data[category[2*i]].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()

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

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

from sklearn.preprocessing import MinMaxScaler
sc=MinMaxScaler()
x_sc=sc.fit_transform(x)
x_sc
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_s
tate=101)
```

In [44]:

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

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

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(random_state=0)
l=lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)
l
y_pred
```

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dense, Activation,Dropout

classifier=keras.Sequential()
classifier.add(Dense(7,activation='relu',input_dim=7))

classifier.add(Dense(7,activation='relu'))
classifier.add(Dense(7,activation='linear'))

classifier.summary()
loss_1=tf.keras.losses.BinaryCrossentropy()
classifier.compile(loss=loss_1,optimizer="Adam",metrics=['accuracy'])
classifier.fit(x_train,y_train,batch_size=20,epochs=100)

from sklearn.metrics import accuracy_score
t_p=classifier.predict(x_train)
```

```
print(t_p)
tr_acc=classifier.evaluate(x_train,y_train,verbose=0)[1]

print(tr_acc)


test_acc=classifier.evaluate(x_test,y_test,verbose=0)[1]

print(test_acc)

from sklearn.metrics import accuracy_score,recall_score, roc_auc_sco
re, confusion_matrix,classification_report

print("Accuracy Score : %f" %(accuracy_score(y_test,y_pred)*100))

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

print("ROC Score : %f" %(roc_auc_score(y_test,y_pred)*100))
print("\nConfusion Matrix")

print(confusion_matrix(y_test,y_pred))

print("\nClassification Report")

print(classification_report(y_test,y_pred))
```