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

Submitted by

V.CHANDRAMOHAN

CHAPTER 1

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 analyse 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 thepotential 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.

Result 1

- Import needed tools
- All needed tools added successfully

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

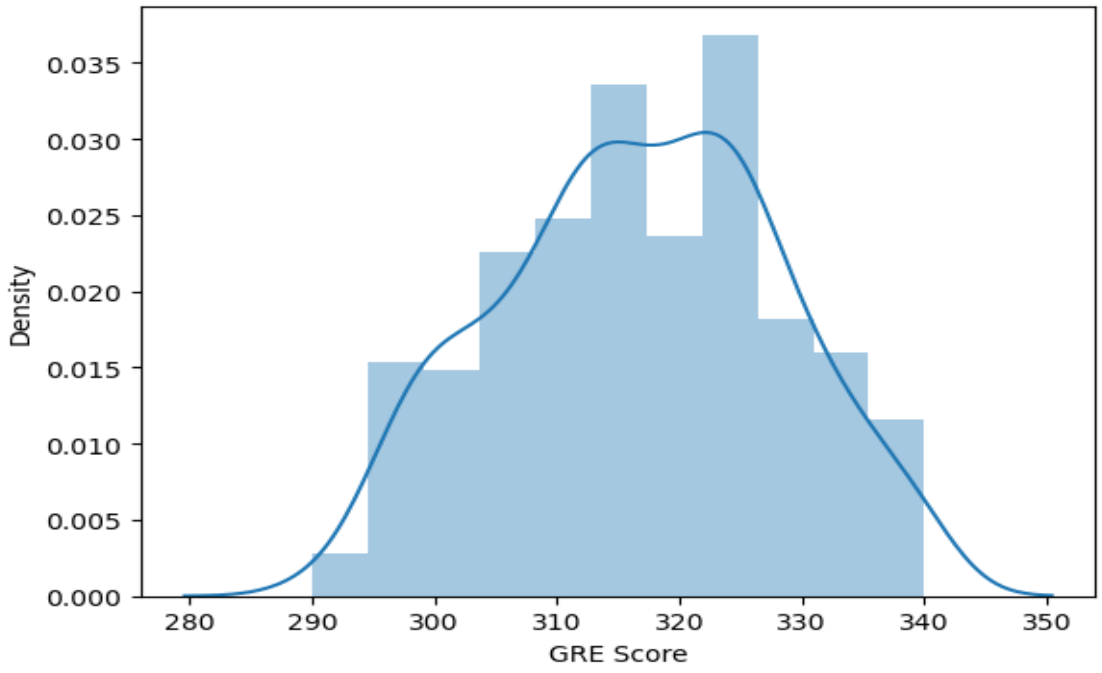
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Serial No.         400 non-null    int64
 1   GRE Score          400 non-null    int64
 2   TOEFL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

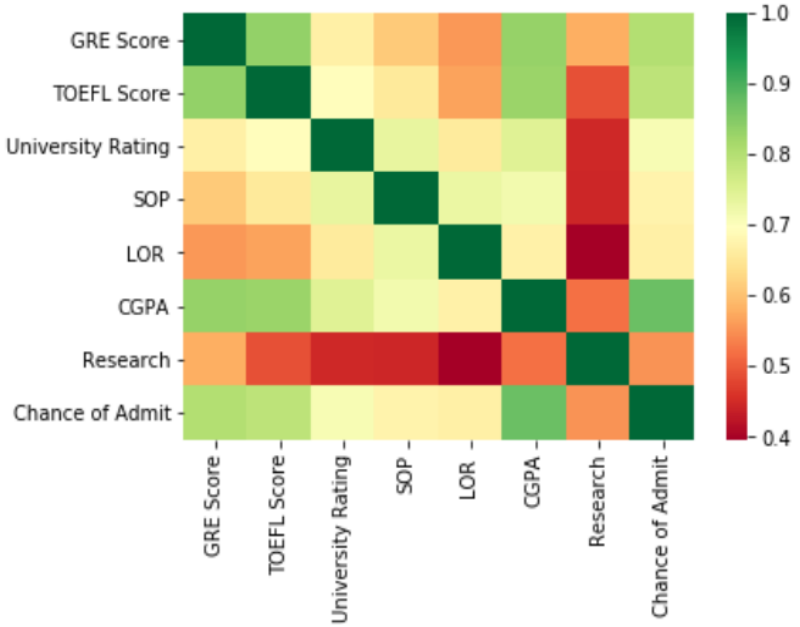
	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	200.500000	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.547500	0.724350
std	115.614301	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

```
<Axes: xlabel='GRE Score', ylabel='Density'>
```

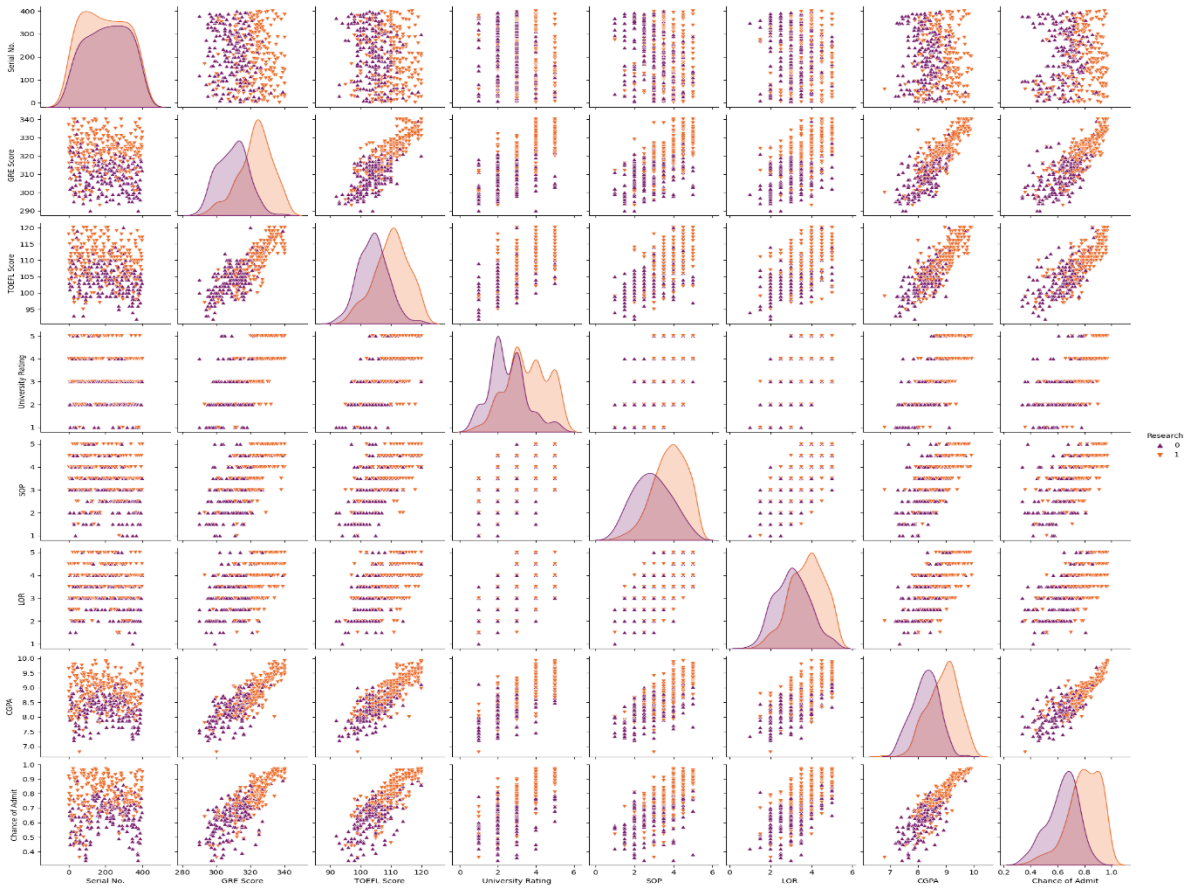


Result 7



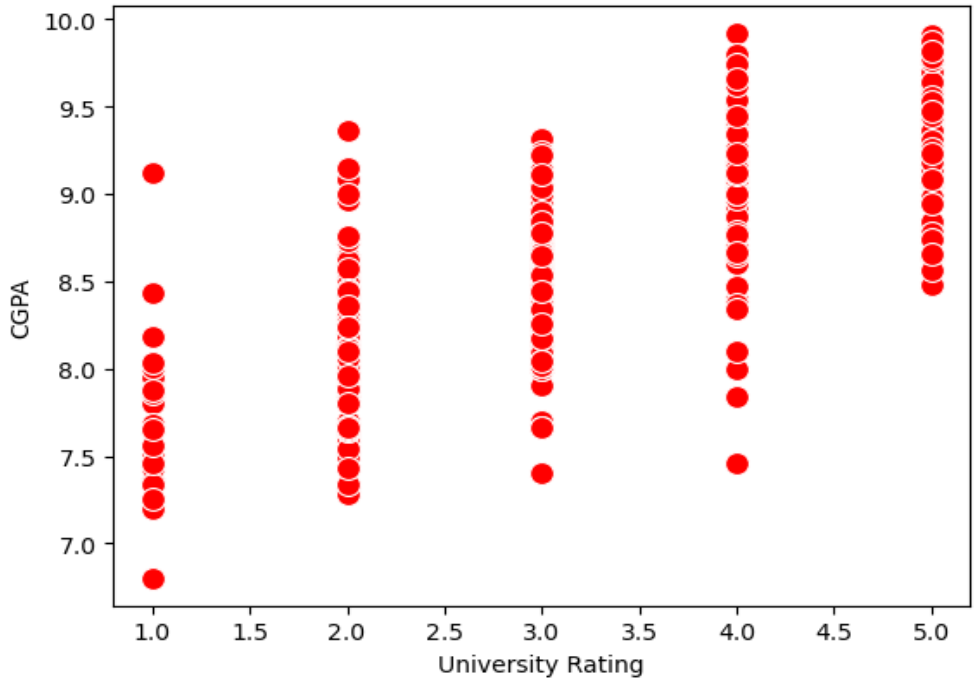
Result 8

```
<seaborn.axisgrid.PairGrid at 0x1394b9b94b0>
```



Result 9

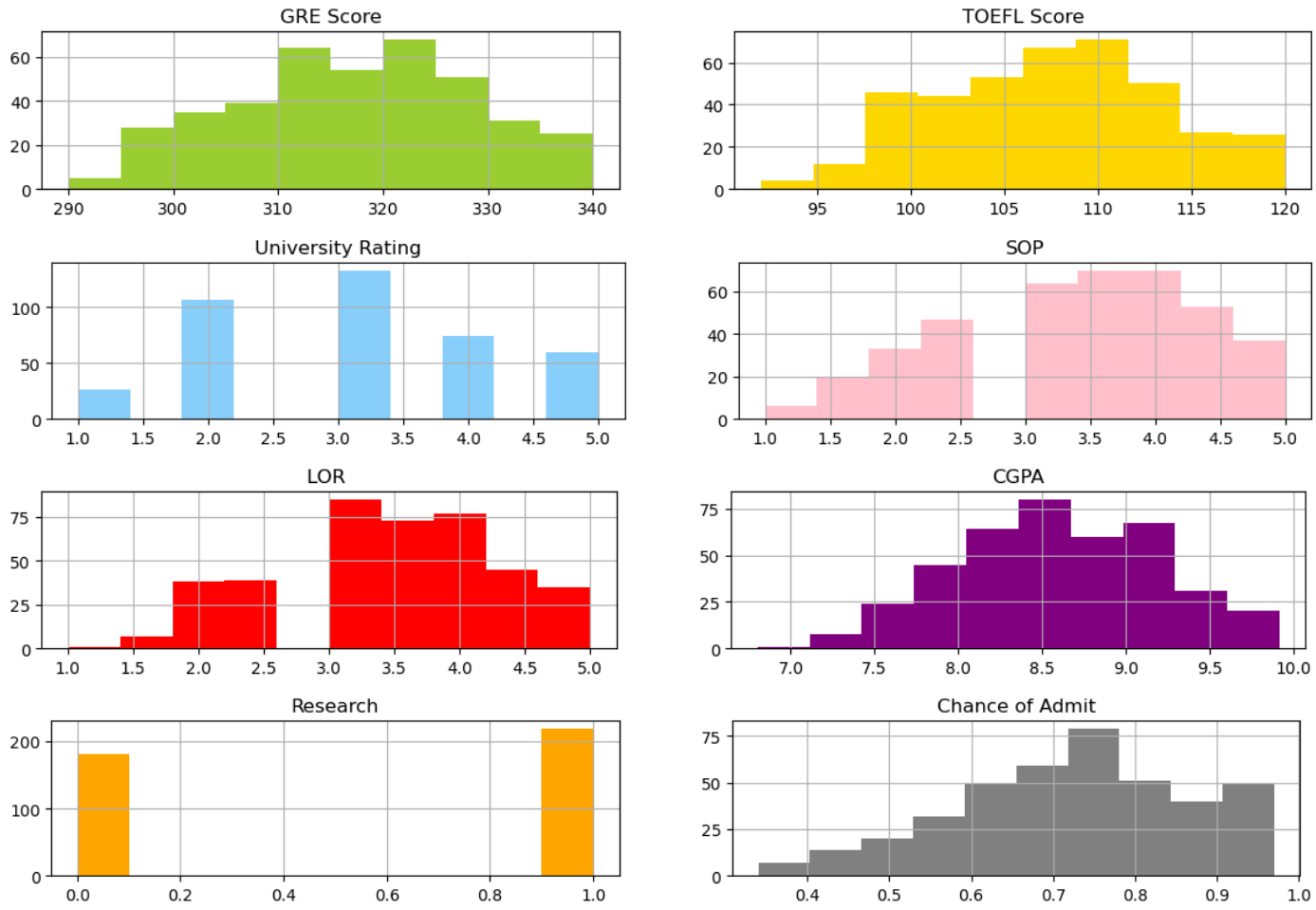
```
<Axes: xlabel='University Rating', ylabel='CGPA'>
```



Result10

```
<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]])
```

Result13

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

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

```
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
```

Result 16

```
Epoch 1/100
14/14 [==============================] - 3s 14ms/step - loss: 14.1579 -
accuracy: 0.0821
Epoch 2/100
14/14 [==============================] - 0s 11ms/step - loss: 14.1579 -
accuracy: 0.0821
Epoch 3/100
14/14 [==============================] - 0s 9ms/step - loss: 14.1579 -
accuracy: 0.0821
Epoch 4/100
14/14 [==============================] - 0s 12ms/step - loss: 14.1579 -
accuracy: 0.0821
Epoch 5/100
14/14 [==============================] - 0s 10ms/step - loss: 14.1579 -
accuracy: 0.0821
Epoch 6/100
14/14 [==============================] - 0s 11ms/step - loss: 14.1579 -
accuracy: 0.0821
Epoch 7/100
14/14 [==============================] - 0s 10ms/step - loss: 14.1579 -
accuracy: 0.0821
Epoch 8/100
14/14 [==============================] - 0s 10ms/step - loss: 14.1579 -
accuracy: 0.0821
Epoch 9/100
14/14 [==============================] - 0s 10ms/step - loss: 14.1579 -
accuracy: 0.0821
```

Result 17

```
9/9 [==============================] - 1s 6ms/step
[[ -98.857574]
 [-107.67336 ]
 [ -94.33377 ]
 [ -79.77212 ]
 [ -71.00769 ]
 [ -72.63707 ]
 [-113.770065]
 [ -76.36096 ]
 [-104.20984 ]
 [ -95.82141 ]
 [ -74.685486]
 [ -92.32373 ]
 [-105.89253 ]
 [ -72.50631 ]
 [-112.265015]
 [-118.16605 ]
```

Result 18

```
Accuracy score: 88.333333
Recall score: 96.296296
ROC score: 56.481481

[[  2  10]
 [  4 104]]
```

Result 19

CLASSIFICATION REPORT

	precision	recall	f1-score	support
False	0.33	0.17	0.22	12
True	0.91	0.96	0.94	108
accuracy			0.88	120
macro avg	0.62	0.56	0.58	120
weighted avg	0.85	0.88	0.87	120