INTELLIGENT ADMISSIONS : THE FUTURE OF UNIVERSITY DECISION MAKING WITH MACHINE LEARNING

INTELLIGENT ADMISSIONS : THE FUTURE OF UNIVERSITY DECISION MAKING WITH MACHINE LEARNING

Project

Submitted to Department of Computer Science shift - II

KAMARAJAR GOVERNMENT ARTS COLLEGE, SURANDAI - 627859

(Manonmaniam Sundaranar University,Tirunelveli)

Done By

TEAM ID: **NM2023TMID20845**

TEAM SIZE : 5

TEAM LEADER : **MATHAN . M** **(REG NO. 20201061506225)**

TEAM MEMBERS : **BALA SARAVANAN . S** **(REG NO. 20201061506207)**

MARI RAJAN . V **(REG NO. 20201061506224)**

MUTHU KUMAR . K **(REG NO. 20201061506228)**

SHANMUGA SIVA . K **(REG NO. 20201061506243)**

Guided By

Dr.SANTHANA DEVI

Department of Computer Science shift - II (KGAC) Surandai-627859 , April 2023

Table of Contents:

INTRODUCTION:

The admission management system is a digital tool that helps educational institutions manage the student enrollment process effortlessly. It lets admission teams capture student inquiries, check their eligibility, follow-up, collect documents, and complete the application process digitally.

Admission Management Software is a tool that automates and streamlines the process of managing and tracking student applications, admissions, and enrolment in educational institutions such as schools, colleges, and universities.

1.1 OVERVIEW:

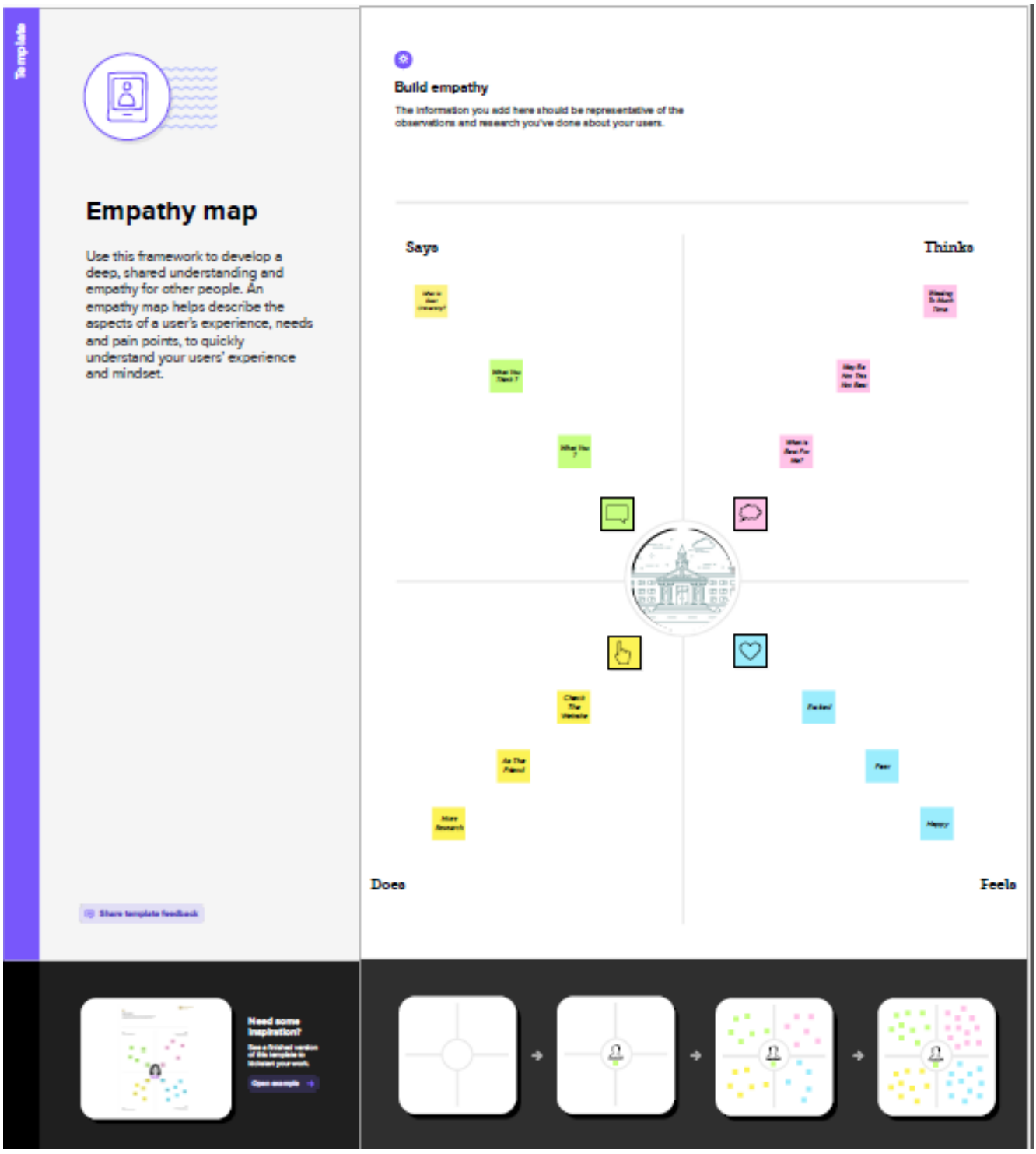
An admission management system is a digital solution to manage student enrollments in colleges, universities, and training institutions. educational institutions use education CRM to distribute inquiries to counselors/admission teams, follow-up with leads, and complete the enrollment process digitally.

1.2 PURPOSE:

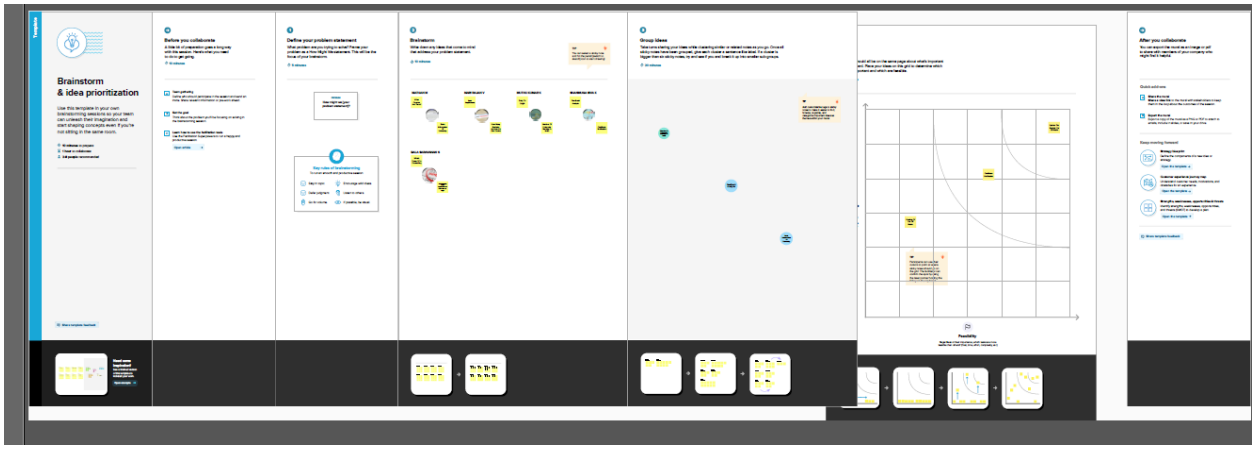
The admission management system is a digital tool that helps educational institutions manage the student enrollment process effortlessly. It lets admission teams capture student inquiries, check their eligibility, follow-up, collect documents, and complete the application process digitally.

2 .Problem Definition & Design Thinking:

2.1 Empathy Map:



2.2 Ideation & Brainstorming Map:



3. RESULT:

Go to web browser and write the localhost URL (http://127.0.0.1:5000) to get the below result

Home Page:

Prediction Form Page:

Example:

Enter GRE Score is - 220

Enter TOEFL Score - 50

Select University No - 2

Enter SOP - 1

Enter LOR - 3

Enter CGPA - 8

Research - No Research

Input the Predict the values

Result:

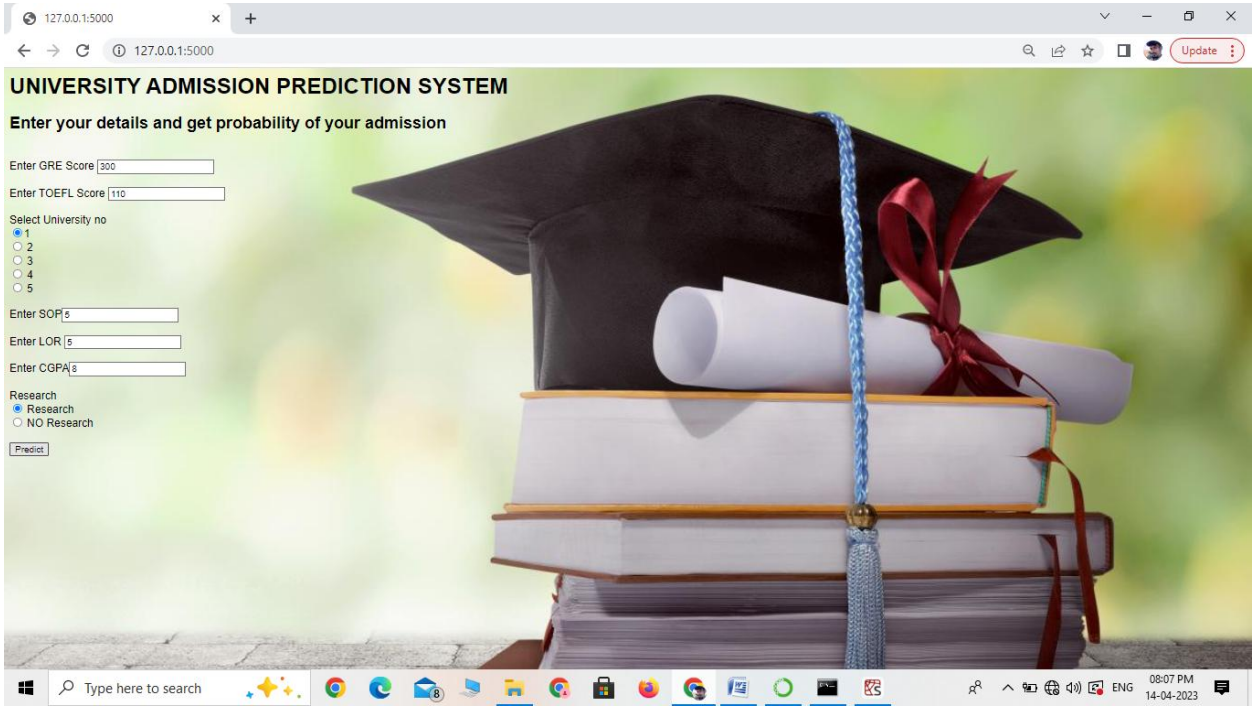
Predicting Chance of Admission

A Machine Learning Web App using Flask.

Prediction : You Dont have a chance of getting admission

prediction : You Dont have a change of getting admission

Another Example Prediction Form Page:



Input the Predict the values:

Enter GRE Score is - 300

Enter TOEFL Score - 110

Select University No - 1

Enter SOP - 5

Enter LOR - 5

Enter CGPA - 8

Research - Research

Result:

Predicting Chance of Admission

A Machine Learning Web App using Flask.

Prediction : You have a chance of getting admission

prediction : You have a change of getting admission

4. ADVANTAGES & DISADVANTAGES:

ADVANTAGES :

- One of the greatest advantages of the online application system is that applicants can choose to submit their applications at their convenience. All that is required is access to a computer and internet connectivity. Messy handwriting, lack of postal connectivity, delay in courier delivery etc. are unlikely to disrupt the application process. This is a great advantage to candidates in rural areas and candidates with disabilities.

- No more running out of paper application forms, picking the right colour ink pens, illegible prints and wondering if the application has been received at all. The online application process offers university applicants a uniform platform for filling in their applications and also provides prompts on which fields are mandatory. The acknowledgement is almost immediate and the system user-friendly.

- Universities and educational institutions are also at a major advantage when it comes to an online admission process. Quick access to student records and databases, efficient systems for filtering out candidates and processing of applications is possible through the online application process. The costs of processing applications and employing additional manpower during admissions are slashed with the implementation of an online application system.

- Those who have seen university officials accepting thousands of paper applications each day at office counters understand that high fatigue and monotony involved in the paperwork is a catalyst for errors. Each error could cost students their academic career and educational prospects. The online admission system is highly reliable and efficient and eliminates chances of such errors.

- Another great advantage of the online admission system is that it makes it possible for candidates from across the country and even abroad to apply to Indian universities without any hassles. It eliminates the inconveniences caused by ailments and exigencies, providing deserving candidates a convenience that has never before been available.

DISADVANTAGES:

- In India, though Internet penetration is rather high, Internet connectivity and speed issues are major impediments to bring any real advantage to university applicants. Most rural areas experience high blackouts and electricity issues. This means, once again candidates in urban districts and areas are placed at a significant advantage.

- Another major concern is the low rate of computer literacy in India. Current estimates say that only about 6.5 percent Indians are computer savvy. A sudden shift to the online admission process is likely to cause confusion and despondency among a great many applicants.

- In a country like India where security fails of online systems have become increasingly common over the years, online applications make it easier for systems to be breached and for applications or scores to be manipulated. The fear that hackers may target universities and educational institutions is a grave one. Unintentional system failures or server crashes may disrupt the entire admission process of universities and educational institutions. Another important concern is the confidentiality of student information and associated security risks involved in online application processing.

- Building a robust and secure online admission process is a task that requires financial and infrastructural resources. Many universities and educational institutions may not have the necessary resources and all these costs will ultimately be borne by the students. In a country where higher education is a luxury few can afford, increased costs may be a deterrent for education.

APPLICATIONS:

The primary objective of this work is to make a Machine Learning model which could be utilized by understudies who need to seek after their Education. Many AI algorithms were used for this examination. Linear Regression model contrasted with different models gives the best outcome. Understudies can utilize the model to survey their shots at getting induction into a specific University with a normal exactness of 82%. An ultimate objective of examination will be cultivated

effectively, as the framework permits understudies to save the parcel of time and cash that they would spend oninstructive guides and application charges for schools where they have less shots at getting affirmations. In future this module of expectation can be incorporated with module of robotized handling framework and different models like neural organization. Likewise, segregate investigation can be utilized independently or joined for upgrading dependability and precision forecast. At long last, understudies can have an open-source AI model which will assist the understudies with knowing their opportunity of entrance into a specific college with high exactness.

CONCLUSION:

The subject of this examination was to determine if the below variables contribute to the admission of student to Master's degree program.

GRE Score
TOEFL Score
University Rating
SOP
LOR
CGPA
Research
Chance of Admit

The results of this examination appear to indicate that it greatly contributes to the response variable 'Chance of Admit'. Higher the GRE, TOEFL score then higher the admit chances. The model predicts 87.5% accuracy and can be used for predicting the admit chances based on the above factors. This model will be helpful for the universities to predict the admission and ease their process of selection and timelines.

As part of the hypothesis, the model proved that admission to Master's degree program is dependent on GRE, TOEFL and other scores.

This model would likely be greatly improved by the gathering of additional data of students from different universities which has similar selection criteria to choose the candidates for Master's program.

FUTURE SCOPE:

The scope of this project is a web application that allows users to enter their academic data and get predictions of their chances of admissions in the university tier of their choosing. It also provides them answers to the most common FAQ's that arise when thinking of admissions abroad for Post Graduate studies. It also provides an analysis based on the data set used that shows how the different parameters affect chances of admissions. A Database will also be implemented for the system so that students can save their data and review and edit it as they progress with the most recent predictions being saved with their profile. Issues of web security other than password protection within the website are not part of this project

APPENDIX :

A. Source Code:

INTELLIGENT ADMISSIONS : THE FUTURE OF UNIVERSITY DECISION MAKING WITH MACHINE LEARNING

Importing the libraries

```
# Import Necessary Libraries
```

```
#import necessary libraries
import pandas as pd
import numpy as np
import pickle
```

```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import  RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix,
f1_score
```

Read the Dataset

```
#read_csv is a pandas function to read csv files
data=pd.read_csv(r"C:\Users\Shivani_SB\OneDrive\Desktop\Projects\10.University_Admission_
Prediction-main\10.University_Admission_Prediction-main\Dataset\Admission_Predict.csv")
```

```
#head() method is used to return top n (5 by default) rows of a DataFrame or series.
data.head(8)
```

```
...     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
    5           6        330          115                  5  4.5   3.0  9.34
    6           7        321          109                  3  3.0   4.0  8.20
    7           8        308          101                  2  3.0   4.0  7.90

       Research  Chance of Admit
    0         1             0.92
    1         1             0.76
    2         1             0.72
    3         1             0.80
    4         0             0.65
    5         1             0.90
    6         1             0.75
    7         0             0.68
```

```
#let us drop Serial No. Column as it is not required for prediction
data.drop(["Serial No."],axis=1,inplace=True)
data.head()
```

```
   GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  Research  \
0        337          118                  4  4.5   4.5  9.65         1
1        324          107                  4  4.0   4.5  8.87         1
2        316          104                  3  3.0   3.5  8.00         1
3        322          110                  3  3.5   2.5  8.67         1
4        314          103                  2  2.0   3.0  8.21         0

   Chance of Admit
0             0.92
1             0.76
2             0.72
3             0.80
4             0.65
```

```
data.describe()
```

```
       GRE Score  TOEFL Score  University Rating         SOP        LOR   \
count  400.000000   400.000000         400.000000  400.000000  400.000000
mean   316.807500   107.410000           3.087500    3.400000    3.452500
std     11.473646     6.069514           1.143728    1.006869    0.898478
min    290.000000    92.000000           1.000000    1.000000    1.000000
25%    308.000000   103.000000           2.000000    2.500000    3.000000
50%    317.000000   107.000000           3.000000    3.500000    3.500000
75%    325.000000   112.000000           4.000000    4.000000    4.000000
max    340.000000   120.000000           5.000000    5.000000    5.000000

              CGPA    Research  Chance of Admit
count   400.000000  400.000000       400.000000
mean      8.598925    0.547500         0.724350
std       0.596317    0.498362         0.142609
min       6.800000    0.000000         0.340000
25%       8.170000    0.000000         0.640000
50%       8.610000    1.000000         0.730000
75%       9.062500    1.000000         0.830000
max       9.920000    1.000000         0.970000
```

From the data we infer that there are only decimal values and no categorical values

```
data.describe()
```

```
       GRE Score  TOEFL Score  University Rating         SOP        LOR   \
count  400.000000   400.000000         400.000000  400.000000  400.000000
mean   316.807500   107.410000           3.087500    3.400000    3.452500
std     11.473646     6.069514           1.143728    1.006869    0.898478
min    290.000000    92.000000           1.000000    1.000000    1.000000
25%    308.000000   103.000000           2.000000    2.500000    3.000000
50%    317.000000   107.000000           3.000000    3.500000    3.500000
75%    325.000000   112.000000           4.000000    4.000000    4.000000
max    340.000000   120.000000           5.000000    5.000000    5.000000

              CGPA    Research  Chance of Admit
count   400.000000  400.000000       400.000000
mean      8.598925    0.547500         0.724350
std       0.596317    0.498362         0.142609
min       6.800000    0.000000         0.340000
25%       8.170000    0.000000         0.640000
50%       8.610000    1.000000         0.730000
75%       9.062500    1.000000         0.830000
max       9.920000    1.000000         0.970000
```

```
#Let us rename the column Chance of Admit because it has trainling space
data=data.rename(columns = {'Chance of Admit ':'Chance of Admit'})
```

```
data.isnull().any()
```

```
GRE Score          False
TOEFL Score        False
University Rating  False
SOP                False
LOR                False
CGPA               False
Research           False
Chance of Admit    False
dtype: bool
```

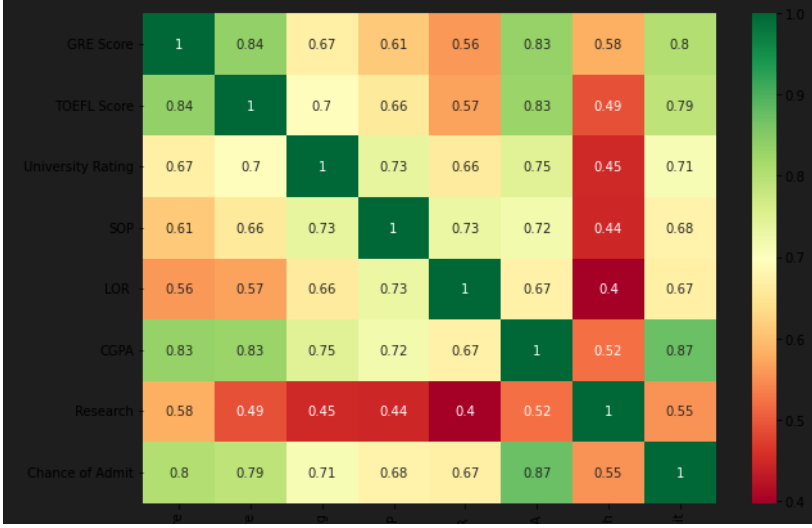
```
data.corr()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
GRE Score	1.000000	0.835977	0.668976	0.612831	0.557555	0.833060	0.580391	0.802610
TOEFL Score	0.835977	1.000000	0.695590	0.657981	0.567721	0.828417	0.489858	0.791594
University Rating	0.668976	0.695590	1.000000	0.734523	0.660123	0.746479	0.447783	0.711250
SOP	0.612831	0.657981	0.734523	1.000000	0.729593	0.718144	0.444029	0.675732
LOR	0.557555	0.567721	0.660123	0.729593	1.000000	0.670211	0.396859	0.669889
CGPA	0.833060	0.828417	0.746479	0.718144	0.670211	1.000000	0.521654	0.873289
Research	0.580391	0.489858	0.447783	0.444029	0.396859	0.521654	1.000000	0.553202
Chance of Admit	0.802610	0.791594	0.711250	0.675732	0.669889	0.873289	0.553202	1.000000

Visual analyisis

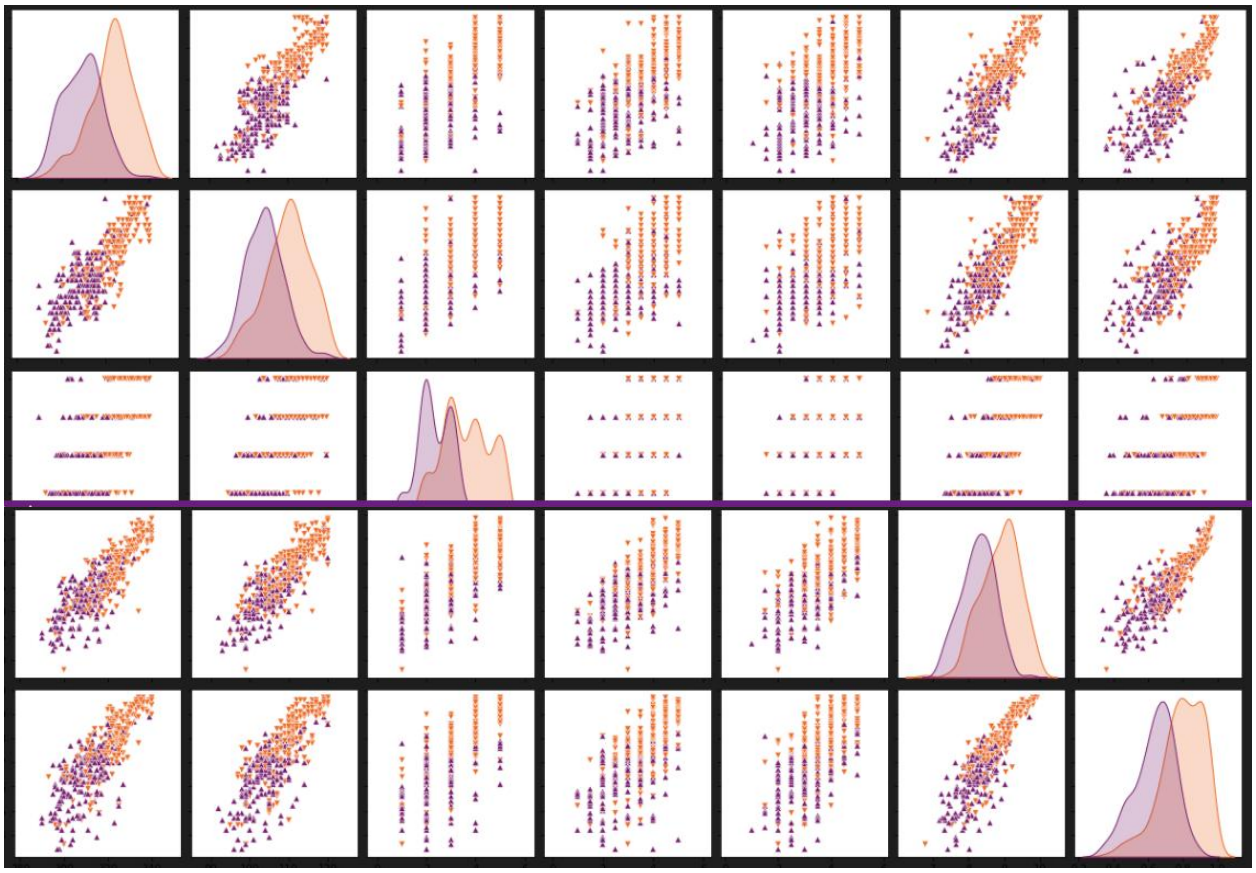
```
plt.figure(figsize=(10,7))

sns.heatmap(data.corr(),annot=True,cmap="RdYlGn")
```

<AxesSubplot:>



We see that the output variable "Chance of Admit" depends on CGPA,GRE,TOEFEL.The columns SOP,LOR and Reserach have less impact on university admission.

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



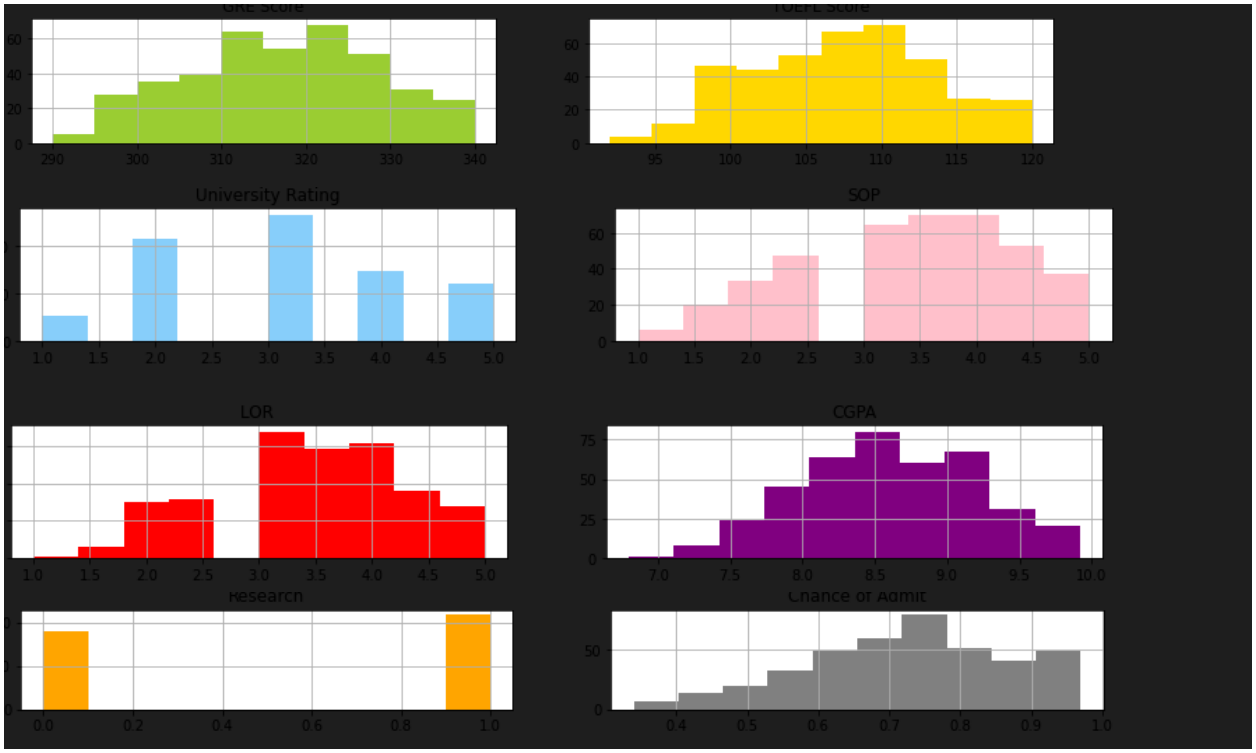
Pair plot usually gives pair wise relationships of the columns in the dataset
From the above pairplot we infer that
1.GRE score TOEFL score and CGPA all are linearly related to each other
2. Students in research score high in TOEFL and GRE compared to non research candidates.

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

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

```
From the above scatter plot we infer that as the CGPA increases the university ratings increases
```

```
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]].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()
```



```
print('Mean CGPA Score is :',int(data['CGPA'].mean()))
print('Mean GRE Score is :',int(data['GRE Score'].mean()))
print('Mean TOEFL Score is :',int(data['TOEFL Score'].mean()))
#print('Mean University rating is :',int(data[data['University Rating']<=500].University Rating.mean()))
```

```
Mean CGPA Score is : 8
Mean GRE Score is : 316
Mean TOEFL Score is : 107
```

```
The chance of admission is high if the aspirant score more than the above mean values
```

```
data.head()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65

```
x=data.iloc[:,0:-1].values
x
```

```
array([[337.  , 118.  ,   4.  , ...,   4.5 ,   9.65,   1.  ],
       [324.  , 107.  ,   4.  , ...,   4.5 ,   8.87,   1.  ],
       [316.  , 104.  ,   3.  , ...,   3.5 ,   8.  ,   1.  ],
       ...,
       [330.  , 116.  ,   4.  , ...,   4.5 ,   9.45,   1.  ],
       [312.  , 103.  ,   3.  , ...,   4.  ,   8.78,   0.  ],
       [333.  , 117.  ,   4.  , ...,   4.  ,   9.66,   1.  ]])
```

AAN Model

```
y=data['Chance of Admit'].values
y
```

```
ray([0.92, 0.76, 0.72, 0.8 , 0.65, 0.9 , 0.75, 0.68, 0.5 , 0.45, 0.52,
     0.84, 0.78, 0.62, 0.61, 0.54, 0.66, 0.65, 0.63, 0.62, 0.64, 0.7 ,
     0.94, 0.95, 0.97, 0.94, 0.76, 0.44, 0.46, 0.54, 0.65, 0.74, 0.91,
     0.9 , 0.94, 0.88, 0.64, 0.58, 0.52, 0.48, 0.46, 0.49, 0.53, 0.87,
     0.91, 0.88, 0.86, 0.89, 0.82, 0.78, 0.76, 0.56, 0.78, 0.72, 0.7 ,
     0.64, 0.64, 0.46, 0.36, 0.42, 0.48, 0.47, 0.54, 0.56, 0.52, 0.55,
     0.61, 0.57, 0.68, 0.78, 0.94, 0.96, 0.93, 0.84, 0.74, 0.72, 0.74,
     0.64, 0.44, 0.46, 0.5 , 0.96, 0.92, 0.92, 0.94, 0.76, 0.72, 0.66,
     0.64, 0.74, 0.64, 0.38, 0.34, 0.44, 0.36, 0.42, 0.48, 0.86, 0.9 ,
     0.79, 0.71, 0.64, 0.62, 0.57, 0.74, 0.69, 0.87, 0.91, 0.93, 0.68,
     0.61, 0.69, 0.62, 0.72, 0.59, 0.66, 0.56, 0.45, 0.47, 0.71, 0.94,
     0.94, 0.57, 0.61, 0.57, 0.64, 0.85, 0.78, 0.84, 0.92, 0.96, 0.77,
     0.71, 0.79, 0.89, 0.82, 0.76, 0.71, 0.8 , 0.78, 0.84, 0.9 , 0.92,
     0.97, 0.8 , 0.81, 0.75, 0.83, 0.96, 0.79, 0.93, 0.94, 0.86, 0.79,
     0.8 , 0.77, 0.7 , 0.65, 0.61, 0.52, 0.57, 0.53, 0.67, 0.68, 0.81,
     0.78, 0.65, 0.64, 0.64, 0.65, 0.68, 0.89, 0.86, 0.89, 0.87, 0.85,
     0.9 , 0.82, 0.72, 0.73, 0.71, 0.71, 0.68, 0.75, 0.72, 0.89, 0.84,
     0.93, 0.93, 0.88, 0.9 , 0.87, 0.86, 0.94, 0.77, 0.78, 0.73, 0.73,
     0.7 , 0.72, 0.73, 0.72, 0.97, 0.97, 0.69, 0.57, 0.63, 0.66, 0.64,
     0.68, 0.79, 0.82, 0.95, 0.96, 0.94, 0.93, 0.91, 0.85, 0.84, 0.74,
        0.76, 0.75, 0.76, 0.71, 0.67, 0.61, 0.63, 0.64, 0.71, 0.82, 0.73,
        0.74, 0.69, 0.64, 0.91, 0.88, 0.85, 0.86, 0.7 , 0.59, 0.6 , 0.65,
        0.7 , 0.76, 0.63, 0.81, 0.72, 0.71, 0.8 , 0.77, 0.74, 0.7 , 0.71,
        0.93, 0.85, 0.79, 0.76, 0.78, 0.77, 0.9 , 0.87, 0.71, 0.7 , 0.7 ,
        0.75, 0.71, 0.72, 0.73, 0.83, 0.77, 0.72, 0.54, 0.49, 0.52, 0.58,
```

```
show more (open the raw output data in a text editor) ...
```

```
        0.79, 0.58, 0.59, 0.47, 0.49, 0.47, 0.42, 0.57, 0.62, 0.74, 0.73,
        0.64, 0.63, 0.59, 0.73, 0.79, 0.68, 0.7 , 0.81, 0.85, 0.93, 0.91,
        0.69, 0.77, 0.86, 0.74, 0.57, 0.51, 0.67, 0.72, 0.89, 0.95, 0.79,
        0.39, 0.38, 0.34, 0.47, 0.56, 0.71, 0.78, 0.73, 0.82, 0.62, 0.96,
        0.96, 0.46, 0.53, 0.49, 0.76, 0.64, 0.71, 0.84, 0.77, 0.89, 0.82,
        0.84, 0.91, 0.67, 0.95])
```

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

```
array([[0.94      , 0.92857143, 0.75      , ..., 0.875     , 0.91346154,
        1.        ],
       [0.68      , 0.53571429, 0.75      , ..., 0.875     , 0.66346154,
        1.        ],
       [0.52      , 0.42857143, 0.5       , ..., 0.625     , 0.38461538,
        1.        ],
       ...,
       [0.8       , 0.85714286, 0.75      , ..., 0.875     , 0.84935897,
        1.        ],
       [0.44      , 0.39285714, 0.5       , ..., 0.75      , 0.63461538,
        0.        ],
       [0.86      , 0.89285714, 0.75      , ..., 0.75      , 0.91666667,
        1.        ]])
```

Splitting data into x and y

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.20,random_state=42)
#random_state acts as the seed for the random number generator during the split
```

```
y_train.shape
```

```
(320,)
```

```
x_train
```

```
array([[0.64      , 0.64285714, 0.5       , ..., 0.375     , 0.59935897,
        1.        ],
       [0.56      , 0.64285714, 0.5       , ..., 0.5       , 0.64102564,
        0.        ],
       [1.        , 1.        , 1.        , ..., 0.875     , 0.99679487,
        1.        ],
       ...,
       [0.32      , 0.46428571, 0.25      , ..., 0.5       , 0.45512821,
        1.        ],
       [0.24      , 0.25      , 0.        , ..., 0.25      , 0.14423077,
        0.        ],
       [0.48      , 0.5       , 0.25      , ..., 0.625     , 0.46474359,
        0.        ]])
```

Testing the model

```
### Let us convert it into classification problem
chance of admit>0.5 as true
chance of admit<0.5 as false
```

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

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

show more (open the raw output data in a text editor) ...

        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  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])
```

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

```
y_test
```

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

Model Builiding

```
#Model building - Logistic Regression
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(yPred_lr,y_test))
    print("***Logistic Regression***")
    print("Confusion_Matrix")
    print(confusion_matrix(y_test,yPred_lr))
    print("Classification Report")
    print(classification_report(y_test,yPred_lr))
```

```
#printing the train accuracy and test accuracy respectively
logreg(x_train,x_test,y_train,y_test)
```

```
0.934375
0.875
***Logistic Regression***
Confusion_Matrix
[[ 0 10]
 [ 0 70]]
Classification Report
              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
```

```
    #testing on test & random input values
    lr = LogisticRegression(random_state=0)
    lr.fit(x_train,y_train)
    print("Predicting on test values")
    lr_pred =lr.predict(x_test)
    print("output is: ",lr_pred)
    print("Predicting on random input")
    lr_pred_own = lr.predict(sc.transform([[337,118,4,4.5,4.5,9.65,1]]))
    print("output is: ",lr_pred_own)
```

```
Predicting on test values
output is:  [ 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  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  True  True  True  True  True  True]
Predicting on random input
output is:  [ True]
```

```
#Model building - Decision Tree Classifier
def decisionTree(x_train,x_test,y_train,y_test):
    dtc = DecisionTreeClassifier(criterion="entropy",random_state=0)
    dtc.fit(x_train,y_train)
    y_dt_tr = dtc.predict(x_train)
    print(accuracy_score(y_dt_tr,y_train))
    yPred_dt = dtc.predict(x_test)
    print(accuracy_score(yPred_dt,y_test))
    print("***Decision Tree***")
    print("Confusion_Matrix")
    print(confusion_matrix(y_test,yPred_dt))
    print("Classification Report")
    print(classification_report(y_test,yPred_dt))
```

```
    #printing the train accuracy and test accuracy respectively
    decisionTree(x_train,x_test,y_train,y_test)
```

```
1.0
0.8875
***Decision Tree***
Confusion_Matrix
[[ 5  5]
 [ 4 66]]
Classification Report
              precision    recall  f1-score   support

       False       0.56      0.50      0.53        10
        True       0.93      0.94      0.94        70

    accuracy                           0.89        80
   macro avg       0.74      0.72      0.73        80
weighted avg       0.88      0.89      0.88        80
```

```
    #testing on test & random input values
    dtc = DecisionTreeClassifier(criterion="entropy",random_state=0)
    dtc.fit(x_train,y_train)
    print("Predicting on test values")
    dtc_pred =dtc.predict(x_test)
    print("output is: ",dtc_pred)
    print("Predicting on random input")
    dtc_pred_own = dtc.predict(sc.transform([[337,118,4,4.5,4.5,9.65,1]]))
    print("output is: ",dtc_pred_own)
```

```
Predicting on test values
output is:  [ True  True  True  True  True  True  True False  True  True  True  True
 False  True  True  True  True False  True  True False  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 False  True  True  True  True  True
  True False  True False False  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True]
Predicting on random input
output is:  [ True]
```

```
#Model building - Random Forest Classifier
def RandomForest(x_tarin,x_test,y_train,y_test):
    rf = RandomForestClassifier(criterion="entropy",n_estimators=10,random_state=0)
    rf.fit(x_train,y_train)
    y_rf_tr = rf.predict(x_train)
    print(accuracy_score(y_rf_tr,y_train))
    yPred_rf = rf.predict(x_test)
    print(accuracy_score(yPred_rf,y_test))
    print("***Random Forest***")
    print("Confusion_Matrix")
    print(confusion_matrix(y_test,yPred_rf))
    print("Classification Report")
    print(classification_report(y_test,yPred_rf))
```

```
    #printing the train accuracy and test accuracy respectively
    RandomForest(x_train,x_test,y_train,y_test)
```

```
0.996875
0.9
***Random Forest***
Confusion_Matrix
[[ 2  8]
 [ 0 70]]
Classification Report
              precision    recall  f1-score   support

       False       1.00      0.20      0.33        10
        True       0.90      1.00      0.95        70

    accuracy                           0.90        80
   macro avg       0.95      0.60      0.64        80
weighted avg       0.91      0.90      0.87        80
```

```
#testing on test & random input values
rf = RandomForestClassifier(criterion="entropy",n_estimators=10,random_state=0)
rf.fit(x_train,y_train)
print("Predicting on test values")
rf_pred =rf.predict(x_test)
print("output is: ",rf_pred)
print("Predicting on random input")
rf_pred_own = rf.predict(sc.transform([[337,118,4,4.5,4.5,9.65,1]]))
print("output is: ",rf_pred_own)
```

```
Predicting on test values
output is:  [ 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  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 False  True  True  True  True  True]
Predicting on random input
output is:  [ True]
```

```python
# Importing the Keras libraries and packages
import keras
from keras.models import Sequential
from keras.layers import Dense
```

```python
# Initialising the ANN
classifier = Sequential()
```

```python
# Adding the input layer and the first hidden layer
classifier.add(Dense(units=7, activation='relu', input_dim=7))
```

```python
# Adding the second hidden layer
classifier.add(Dense(units=7, activation='relu'))
```

```python
# Adding the output layer
classifier.add(Dense(units=1, activation='linear'))
```

```python
# Compiling the ANN
classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```python
# Fitting the ANN to the Training set
model = classifier.fit(x_train, y_train, batch_size=10, validation_split=0.33, epochs=20)
```

```
Epoch 1/20
22/22 [==============================] - 1s 15ms/step - loss: 1.1571 - accuracy: 0.2196 - val_loss: 0.8039 - val_accuracy: 0.4811
Epoch 2/20
22/22 [==============================] - 0s 4ms/step - loss: 0.6113 - accuracy: 0.6542 - val_loss: 0.4385 - val_accuracy: 0.8208
Epoch 3/20
22/22 [==============================] - 0s 3ms/step - loss: 0.3562 - accuracy: 0.8738 - val_loss: 0.3825 - val_accuracy: 0.9057
Epoch 4/20
22/22 [==============================] - 0s 4ms/step - loss: 0.3839 - accuracy: 0.9112 - val_loss: 0.3369 - val_accuracy: 0.9245
Epoch 5/20
22/22 [==============================] - 0s 3ms/step - loss: 0.3551 - accuracy: 0.9252 - val_loss: 0.3206 - val_accuracy: 0.9245
Epoch 6/20
22/22 [==============================] - 0s 3ms/step - loss: 0.3898 - accuracy: 0.9346 - val_loss: 0.2987 - val_accuracy: 0.9151
Epoch 7/20
22/22 [==============================] - 0s 4ms/step - loss: 0.4319 - accuracy: 0.9346 - val_loss: 0.2828 - val_accuracy: 0.9245
Epoch 8/20
22/22 [==============================] - 0s 4ms/step - loss: 0.4206 - accuracy: 0.9346 - val_loss: 0.2781 - val_accuracy: 0.9245
Epoch 9/20
22/22 [==============================] - 0s 4ms/step - loss: 0.4167 - accuracy: 0.9299 - val_loss: 0.2751 - val_accuracy: 0.9245
Epoch 10/20
22/22 [==============================] - 0s 5ms/step - loss: 0.4120 - accuracy: 0.9299 - val_loss: 0.2728 - val_accuracy: 0.9245
Epoch 11/20
```

```
ann_pred = classifier.predict(x_test)
ann_pred = (ann_pred>0.5)
print(accuracy_score(ann_pred,y_test))
print("***ANN Model***")
print("Confusion_Matrix")
print(confusion_matrix(y_test,ann_pred))
print("Classification Report")
print(classification_report(y_test,ann_pred))
```

```
3/3 [==============================] - 0s 0s/step
0.8875
***ANN Model***
Confusion_Matrix
[[ 1  9]
 [ 0 70]]
Classification Report
              precision    recall  f1-score   support

       False       1.00      0.10      0.18        10
        True       0.89      1.00      0.94        70

    accuracy                           0.89        80
   macro avg       0.94      0.55      0.56        80
weighted avg       0.90      0.89      0.84        80
```

```
#testing on test & random input values
print("Predicting on test input")
ann_pred = classifier.predict(x_test)
ann_pred = (ann_pred>0.5)
print("output is: ",ann_pred)
print("Predicting on random input")
ann_pred_own = classifier.predict(sc.transform([[337,118,4,4.5,4.5,9.65,1]]))
ann_pred_own = (ann_pred_own>0.5)
print("output is: ",ann_pred_own)
```

```
Predicting on test input
3/3 [==============================] - 0s 2ms/step
output is:  [[ True]
 [ True]
 [ True]
 [ True]
 [ True]
 [ True]
 [ True]
 [ True]
 [ True]
 [ True]
 [ True]
 [ True]
 [ True]
```

```
ann_pred_train = classifier.predict(x_train)
ann_pred_train = (ann_pred_train>0.5)
print(accuracy_score(ann_pred_train,y_train))
print("***ANN Model***")
print("Confusion_Matrix")
print(confusion_matrix(ann_pred_train,y_train))
print("Classification Report")
print(classification_report(ann_pred_train,y_train))
```

```
10/10 [==============================] - 0s 4ms/step
0.934375
***ANN Model***
Confusion_Matrix
[[  9   5]
 [ 16 290]]
Classification Report
              precision    recall  f1-score   support

       False       0.36      0.64      0.46        14
        True       0.98      0.95      0.97       306

    accuracy                           0.93       320
   macro avg       0.67      0.80      0.71       320
weighted avg       0.96      0.93      0.94       320
```

```
pickle.dump(lr,open('university.pkl','wb'))
```


```
pickle.dump(lr,open('university.pkl','wb'))
```