**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**%matplotlib** inline

In [2]:

*# mount google drive and copy dataset*

**from** google.colab **import** drive

drive**.**mount('/content/drive')

**%cp** '/content/drive/My Drive/university\_admission\_prediction/dataset/Admission\_Predict.csv' '/content/'

Mounted at /content/drive

In [3]:

data **=** pd**.**read\_csv('Admission\_Predict.csv')

In [4]:

data**.**head()

Out[4]:

|  | **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 |

In [5]:

data**.**drop(["Serial No."],axis**=**1,inplace**=True**)

data**.**head()

Out[5]:

|  | **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 |

In [6]:

data**.**describe()

Out[6]:

|  | **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 |
| **mean** | 316.807500 | 107.410000 | 3.087500 | 3.400000 | 3.452500 | 8.598925 | 0.547500 | 0.724350 |
| **std** | 11.473646 | 6.069514 | 1.143728 | 1.006869 | 0.898478 | 0.596317 | 0.498362 | 0.142609 |
| **min** | 290.000000 | 92.000000 | 1.000000 | 1.000000 | 1.000000 | 6.800000 | 0.000000 | 0.340000 |
| **25%** | 308.000000 | 103.000000 | 2.000000 | 2.500000 | 3.000000 | 8.170000 | 0.000000 | 0.640000 |
| **50%** | 317.000000 | 107.000000 | 3.000000 | 3.500000 | 3.500000 | 8.610000 | 1.000000 | 0.730000 |
| **75%** | 325.000000 | 112.000000 | 4.000000 | 4.000000 | 4.000000 | 9.062500 | 1.000000 | 0.830000 |
| **max** | 340.000000 | 120.000000 | 5.000000 | 5.000000 | 5.000000 | 9.920000 | 1.000000 | 0.970000 |

In [7]:

data**.**info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 400 entries, 0 to 399

Data columns (total 8 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 GRE Score 400 non-null int64

1 TOEFL Score 400 non-null int64

2 University Rating 400 non-null int64

3 SOP 400 non-null float64

4 LOR 400 non-null float64

5 CGPA 400 non-null float64

6 Research 400 non-null int64

7 Chance of Admit 400 non-null float64

dtypes: float64(4), int64(4)

memory usage: 25.1 KB

In [8]:

data **=** data**.**rename(columns **=** {'Chance of Admit ':'Chance of Admit'})

data**.**head()

Out[8]:

|  | **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 |

In [9]:

data**.**isnull()**.**any()

Out[9]:

GRE Score False

TOEFL Score False

University Rating False

SOP False

LOR False

CGPA False

Research False

Chance of Admit False

dtype: bool

In [10]:

data**.**corr()

Out[10]:

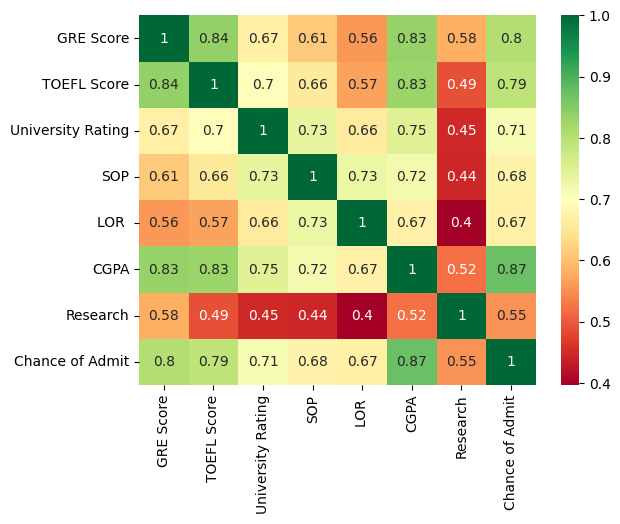
|  | **GRE Score** | **TOEFL Score** | **University Rating** | **SOP** | **LOR** | **CGPA** | **Research** | **Chance of Admit** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **GRE Score** | 1.000000 | 0.835977 | 0.668976 | 0.612831 | 0.557555 | 0.833060 | 0.580391 | 0.802610 |
| **TOEFL 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 |

In [11]:

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

Out[11]:

<Axes: >

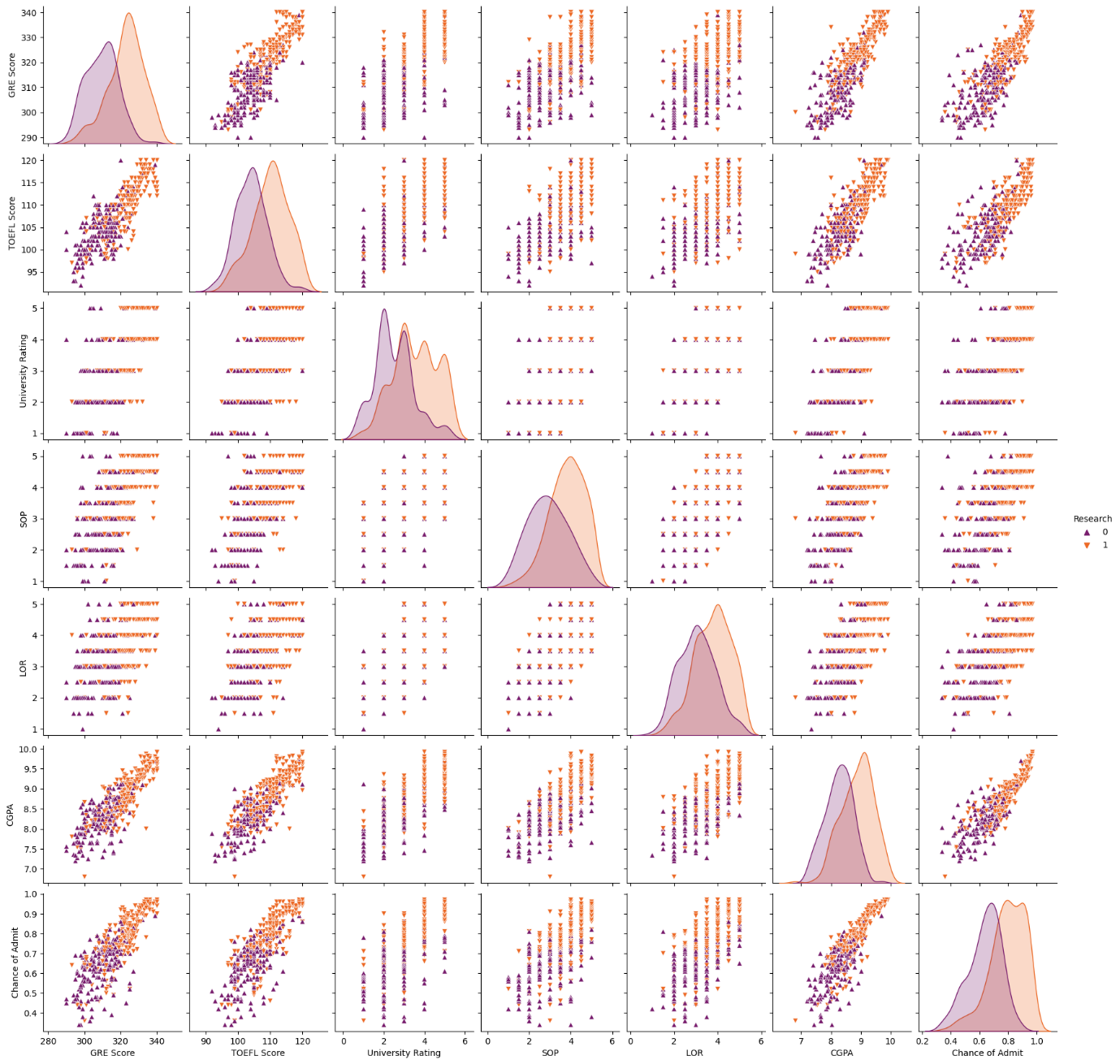


In [12]:

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

Out[12]:

<seaborn.axisgrid.PairGrid at 0x7f296c58b5e0>

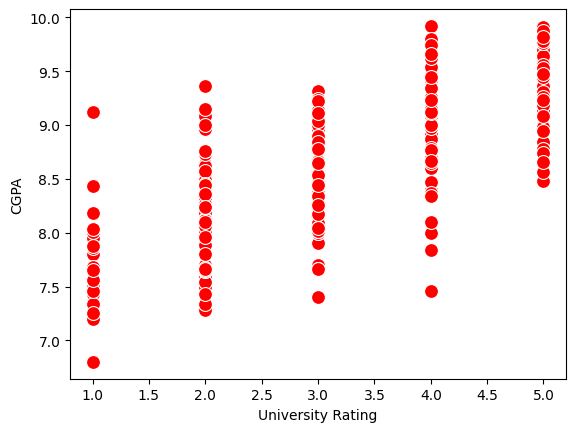


In [13]:

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

Out[13]:

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



In [14]:

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])

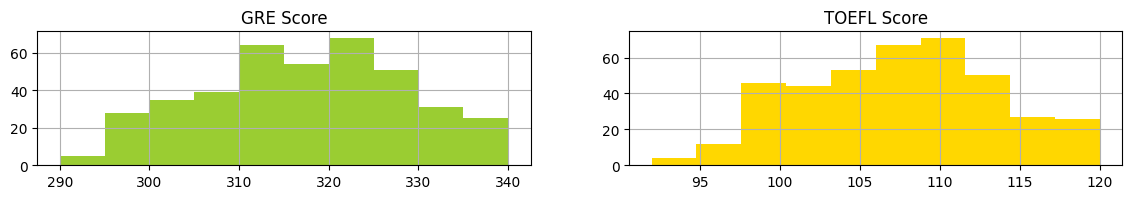
plt**.**subplot2grid((4,2), (i,1))

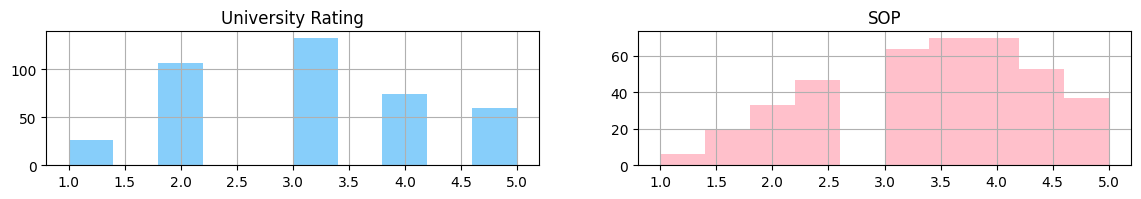
data[category[2**\***i**+**1]]**.**hist(color**=**color[2**\***i**+**1], bins**=**10)

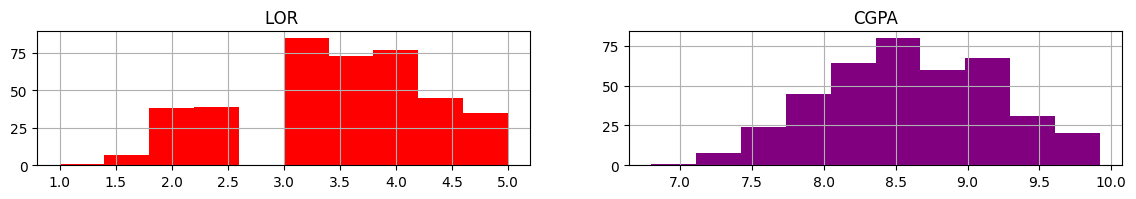
plt**.**title(category[2**\***i**+**1])

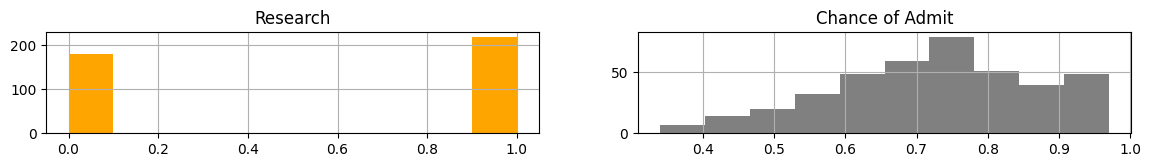
plt**.**subplots\_adjust(hspace**=**0.7, wspace**=**0.2)

plt**.**show()









In [16]:

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

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

In [17]:

**from** sklearn.preprocessing **import** MinMaxScaler

sc **=** MinMaxScaler()

x **=** sc**.**fit\_transform(x)

In [18]:

**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)

y\_train **=** (y\_train**>**0.5)

y\_test **=** (y\_test**>**0.5)

In [36]:

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.metrics **import** accuracy\_score, recall\_score, roc\_auc\_score, classification\_report, confusion\_matrix

cls **=** LogisticRegression(random\_state**=**0)

lr **=** cls**.**fit(x\_train, y\_train)

y\_pred **=** lr**.**predict(x\_test)

print("Logistic Regression")

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("Confusion Matrix \n", confusion\_matrix(y\_test, y\_pred))

Logistic Regression

Accuracy score : 87.500000

Recall score : 100.000000

ROC score : 50.000000

Confusion Matrix

[[ 0 10]

[ 0 70]]

In [23]:

**import** keras

**from** keras.models **import** Sequential

**from** keras.layers **import** Dense

In [24]:

model **=** Sequential()

In [25]:

model**.**add(Dense(units**=**7, activation**=**'relu', input\_dim**=**7))

In [26]:

model**.**add(Dense(units**=**7, activation**=**'relu'))

In [27]:

model**.**add(Dense(units**=**1, activation**=**'linear'))

In [28]:

model**.**compile(optimizer**=**'adam', loss**=**'binary\_crossentropy', metrics**=**['accuracy'])

In [30]:

model**.**fit(x\_train, y\_train, batch\_size**=**20, epochs**=**100)

Epoch 1/100

16/16 [==============================] - 0s 6ms/step - loss: 0.2118 - accuracy: 0.9281

Epoch 2/100

16/16 [==============================] - 0s 5ms/step - loss: 0.2103 - accuracy: 0.9281

Epoch 3/100

16/16 [==============================] - 0s 8ms/step - loss: 0.2092 - accuracy: 0.9281

Epoch 4/100

16/16 [==============================] - 0s 6ms/step - loss: 0.2086 - accuracy: 0.9281

Epoch 5/100

16/16 [==============================] - 0s 6ms/step - loss: 0.2084 - accuracy: 0.9281

Epoch 6/100

16/16 [==============================] - 0s 6ms/step - loss: 0.2072 - accuracy: 0.9312

Epoch 7/100

16/16 [==============================] - 0s 7ms/step - loss: 0.2066 - accuracy: 0.9312

Epoch 8/100

16/16 [==============================] - 0s 9ms/step - loss: 0.2067 - accuracy: 0.9312

Epoch 9/100

16/16 [==============================] - 0s 4ms/step - loss: 0.2058 - accuracy: 0.9312

Epoch 10/100

16/16 [==============================] - 0s 4ms/step - loss: 0.2050 - accuracy: 0.9312

Epoch 11/100

16/16 [==============================] - 0s 6ms/step - loss: 0.2044 - accuracy: 0.9312

Epoch 12/100

16/16 [==============================] - 0s 6ms/step - loss: 0.2046 - accuracy: 0.9312

Epoch 13/100

16/16 [==============================] - 0s 6ms/step - loss: 0.2035 - accuracy: 0.9312

Epoch 14/100

16/16 [==============================] - 0s 7ms/step - loss: 0.2028 - accuracy: 0.9312

Epoch 15/100

16/16 [==============================] - 0s 6ms/step - loss: 0.2024 - accuracy: 0.9344

Epoch 16/100

16/16 [==============================] - 0s 7ms/step - loss: 0.2017 - accuracy: 0.9344

Epoch 17/100

16/16 [==============================] - 0s 6ms/step - loss: 0.2012 - accuracy: 0.9344

Epoch 18/100

16/16 [==============================] - 0s 3ms/step - loss: 0.2009 - accuracy: 0.9344

Epoch 19/100

16/16 [==============================] - 0s 3ms/step - loss: 0.2003 - accuracy: 0.9344

Epoch 20/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1998 - accuracy: 0.9344

Epoch 21/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1993 - accuracy: 0.9312

Epoch 22/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1990 - accuracy: 0.9375

Epoch 23/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1985 - accuracy: 0.9344

Epoch 24/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1978 - accuracy: 0.9375

Epoch 25/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1974 - accuracy: 0.9375

Epoch 26/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1971 - accuracy: 0.9375

Epoch 27/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1968 - accuracy: 0.9375

Epoch 28/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1971 - accuracy: 0.9281

Epoch 29/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1955 - accuracy: 0.9344

Epoch 30/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1955 - accuracy: 0.9375

Epoch 31/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1954 - accuracy: 0.9312

Epoch 32/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1951 - accuracy: 0.9344

Epoch 33/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1945 - accuracy: 0.9312

Epoch 34/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1946 - accuracy: 0.9312

Epoch 35/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1936 - accuracy: 0.9312

Epoch 36/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1934 - accuracy: 0.9312

Epoch 37/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1928 - accuracy: 0.9312

Epoch 38/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1924 - accuracy: 0.9312

Epoch 39/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1916 - accuracy: 0.9312

Epoch 40/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1909 - accuracy: 0.9312

Epoch 41/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1907 - accuracy: 0.9312

Epoch 42/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1901 - accuracy: 0.9312

Epoch 43/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1901 - accuracy: 0.9312

Epoch 44/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1896 - accuracy: 0.9312

Epoch 45/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1901 - accuracy: 0.9312

Epoch 46/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1896 - accuracy: 0.9312

Epoch 47/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1892 - accuracy: 0.9312

Epoch 48/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1885 - accuracy: 0.9312

Epoch 49/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1890 - accuracy: 0.9312

Epoch 50/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1880 - accuracy: 0.9312

Epoch 51/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1875 - accuracy: 0.9312

Epoch 52/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1881 - accuracy: 0.9312

Epoch 53/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1875 - accuracy: 0.9312

Epoch 54/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1877 - accuracy: 0.9312

Epoch 55/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1875 - accuracy: 0.9312

Epoch 56/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1870 - accuracy: 0.9312

Epoch 57/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1877 - accuracy: 0.9312

Epoch 58/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1871 - accuracy: 0.9312

Epoch 59/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1874 - accuracy: 0.9312

Epoch 60/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1869 - accuracy: 0.9312

Epoch 61/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1862 - accuracy: 0.9312

Epoch 62/100

16/16 [==============================] - 0s 3ms/step - loss: 0.1863 - accuracy: 0.9312

Epoch 63/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1858 - accuracy: 0.9312

Epoch 64/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1858 - accuracy: 0.9312

Epoch 65/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1859 - accuracy: 0.9312

Epoch 66/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1851 - accuracy: 0.9312

Epoch 67/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1855 - accuracy: 0.9312

Epoch 68/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1855 - accuracy: 0.9312

Epoch 69/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1854 - accuracy: 0.9312

Epoch 70/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1846 - accuracy: 0.9312

Epoch 71/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1847 - accuracy: 0.9375

Epoch 72/100

16/16 [==============================] - 0s 3ms/step - loss: 0.1844 - accuracy: 0.9312

Epoch 73/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1846 - accuracy: 0.9312

Epoch 74/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1848 - accuracy: 0.9375

Epoch 75/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1841 - accuracy: 0.9375

Epoch 76/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1840 - accuracy: 0.9375

Epoch 77/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1838 - accuracy: 0.9375

Epoch 78/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1838 - accuracy: 0.9375

Epoch 79/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1838 - accuracy: 0.9375

Epoch 80/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1835 - accuracy: 0.9375

Epoch 81/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1833 - accuracy: 0.9375

Epoch 82/100

16/16 [==============================] - 0s 3ms/step - loss: 0.1831 - accuracy: 0.9344

Epoch 83/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1832 - accuracy: 0.9375

Epoch 84/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1832 - accuracy: 0.9375

Epoch 85/100

16/16 [==============================] - 0s 3ms/step - loss: 0.1830 - accuracy: 0.9375

Epoch 86/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1828 - accuracy: 0.9375

Epoch 87/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1828 - accuracy: 0.9375

Epoch 88/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1828 - accuracy: 0.9375

Epoch 89/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1818 - accuracy: 0.9375

Epoch 90/100

16/16 [==============================] - 0s 3ms/step - loss: 0.1826 - accuracy: 0.9375

Epoch 91/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1824 - accuracy: 0.9375

Epoch 92/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1832 - accuracy: 0.9375

Epoch 93/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1825 - accuracy: 0.9375

Epoch 94/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1813 - accuracy: 0.9406

Epoch 95/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1818 - accuracy: 0.9375

Epoch 96/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1814 - accuracy: 0.9375

Epoch 97/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1810 - accuracy: 0.9438

Epoch 98/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1809 - accuracy: 0.9375

Epoch 99/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1808 - accuracy: 0.9375

Epoch 100/100

16/16 [==============================] - 0s 2ms/step - loss: 0.1811 - accuracy: 0.9438

Out[30]:

<keras.callbacks.History at 0x7f28fae9fd90>

In [31]:

ann\_pred\_tr **=** model**.**predict(x\_train)

print(ann\_pred\_tr)

10/10 [==============================] - 0s 2ms/step

[[1.1659797 ]

[1.2464826 ]

[1.9496695 ]

[0.9997218 ]

[0.7984851 ]

[1.161485 ]

[1.3990479 ]

[1.2182487 ]

[1.812677 ]

[1.0665838 ]

[0.90455514]

[1.5609401 ]

[0.55158216]

[1.1878558 ]

[0.8404469 ]

[1.0434524 ]

[1.1380981 ]

[1.2840469 ]

[0.7263538 ]

[1.307327 ]

[1.0989944 ]

[0.84802777]

[1.1670483 ]

[1.3818789 ]

[1.2268969 ]

[0.9124365 ]

[0.9491717 ]

[0.788024 ]

[1.0271555 ]

[0.8622387 ]

[0.96847606]

[0.995249 ]

[0.6654349 ]

[0.8857574 ]

[1.6144438 ]

[0.8832684 ]

[0.47630844]

[1.5982643 ]

[1.2998719 ]

[1.5456327 ]

[0.9479347 ]

[0.93734545]

[0.99973524]

[0.35015008]

[1.724817 ]

[1.3941329 ]

[1.3325704 ]

[0.9513187 ]

[0.9453948 ]

[1.2890598 ]

[1.505834 ]

[1.1241236 ]

[1.3667225 ]

[1.3546215 ]

[1.1741034 ]

[1.214382 ]

[1.292426 ]

[1.0055559 ]

[1.3226632 ]

[1.489653 ]

[1.0797732 ]

[1.4036678 ]

[0.6865502 ]

[1.180652 ]

[0.97194666]

[1.810835 ]

[0.8962222 ]

[1.3445202 ]

[1.2842536 ]

[1.0048044 ]

[1.1620904 ]

[1.1765063 ]

[1.3038629 ]

[1.654493 ]

[1.2024684 ]

[1.4665554 ]

[1.1469942 ]

[1.9529418 ]

[0.661456 ]

[1.1938838 ]

[0.72139055]

[0.9942846 ]

[1.6508802 ]

[0.75466305]

[0.7256388 ]

[1.3212556 ]

[0.59665066]

[1.0907456 ]

[1.3877077 ]

[1.2769487 ]

[0.69504917]

[1.6217114 ]

[1.049429 ]

[0.85034245]

[1.9330329 ]

[0.5832885 ]

[1.6388519 ]

[1.0419348 ]

[1.4691683 ]

[1.8734323 ]

[0.75007653]

[1.3822492 ]

[0.85402995]

[0.81791896]

[1.2276056 ]

[1.0800661 ]

[0.96448475]

[1.0648762 ]

[0.80171984]

[1.150174 ]

[0.92156947]

[0.98223644]

[1.3777925 ]

[1.0558765 ]

[1.1821424 ]

[1.0690422 ]

[1.8650087 ]

[0.6847035 ]

[1.3205104 ]

[1.0586442 ]

[1.3169018 ]

[1.9607621 ]

[1.2391902 ]

[1.5616592 ]

[1.2647372 ]

[1.8399249 ]

[0.68876255]

[1.5336125 ]

[1.5797288 ]

[1.630488 ]

[1.3530236 ]

[1.1115552 ]

[0.94175375]

[0.62033916]

[1.2432376 ]

[0.5198856 ]

[0.7360702 ]

[1.0641536 ]

[1.8813597 ]

[1.6042653 ]

[1.7586545 ]

[1.5127057 ]

[1.8322536 ]

[0.98831517]

[1.2821462 ]

[1.2716278 ]

[1.014998 ]

[1.4653614 ]

[0.9552186 ]

[1.0710257 ]

[1.4663862 ]

[1.6088854 ]

[1.7100195 ]

[1.1584363 ]

[1.8909638 ]

[0.9495287 ]

[0.899702 ]

[1.4558784 ]

[0.6900437 ]

[1.0838922 ]

[0.9149205 ]

[1.0849272 ]

[1.3526169 ]

[1.1245396 ]

[0.5477795 ]

[1.6368581 ]

[1.2411474 ]

[1.7466811 ]

[1.4012731 ]

[1.1863843 ]

[1.3469716 ]

[1.2115396 ]

[1.1253372 ]

[0.82508546]

[1.5909547 ]

[0.58669645]

[1.3356922 ]

[1.0436403 ]

[1.3385843 ]

[1.1682891 ]

[0.71853906]

[1.579703 ]

[1.6370277 ]

[0.42931566]

[0.88963765]

[1.5592724 ]

[1.0991896 ]

[0.5055347 ]

[1.2448555 ]

[0.5787552 ]

[0.7697874 ]

[0.7798321 ]

[1.7006959 ]

[0.983754 ]

[0.9892668 ]

[0.8913196 ]

[1.1632844 ]

[0.81196445]

[1.5375746 ]

[0.8360475 ]

[0.96561044]

[1.0119462 ]

[0.75459236]

[1.173541 ]

[1.2231892 ]

[1.3132006 ]

[1.8884324 ]

[1.6502391 ]

[1.8091196 ]

[1.7303351 ]

[0.81279165]

[1.7164847 ]

[1.7154964 ]

[1.0509739 ]

[0.99643147]

[1.3585087 ]

[0.48917875]

[1.4741632 ]

[1.211169 ]

[1.4146845 ]

[1.2457489 ]

[1.1363358 ]

[0.9355455 ]

[1.5236021 ]

[0.9652453 ]

[1.3273345 ]

[1.3823997 ]

[0.7418794 ]

[1.2726296 ]

[0.9110289 ]

[0.85955435]

[0.9469427 ]

[1.0217112 ]

[1.1944525 ]

[0.8826204 ]

[1.6472629 ]

[1.1740183 ]

[0.5880517 ]

[1.5946962 ]

[1.0083128 ]

[2.008243 ]

[0.99491113]

[1.2533044 ]

[1.1419464 ]

[0.92129713]

[1.4860212 ]

[1.1995591 ]

[1.0704529 ]

[1.1704453 ]

[0.67674184]

[1.5184383 ]

[1.4676707 ]

[1.4956404 ]

[1.0270195 ]

[1.8315779 ]

[1.4539346 ]

[1.0664052 ]

[1.4245287 ]

[1.2466913 ]

[0.8497558 ]

[0.6801288 ]

[1.8208221 ]

[1.0380583 ]

[0.49700457]

[1.2633067 ]

[1.1807802 ]

[0.54863703]

[1.1758933 ]

[0.87821263]

[1.035088 ]

[0.935908 ]

[0.98511493]

[0.5300671 ]

[1.0586201 ]

[1.366802 ]

[1.1536832 ]

[1.5769297 ]

[1.4341843 ]

[0.88842267]

[1.275788 ]

[1.3992635 ]

[1.0435504 ]

[1.0054433 ]

[1.4585556 ]

[1.6519226 ]

[1.8118066 ]

[1.6366047 ]

[0.8310694 ]

[0.36863607]

[1.5064498 ]

[0.43683797]

[1.5236229 ]

[1.0174628 ]

[1.0272447 ]

[0.72946656]

[0.7956009 ]

[1.7111436 ]

[1.5506397 ]

[0.88910174]

[0.97198933]

[1.2195182 ]

[1.1253276 ]

[1.0408169 ]

[1.7971307 ]

[1.7926351 ]

[0.9478293 ]

[1.4845046 ]

[1.8280452 ]

[1.0601386 ]

[1.2807096 ]

[1.8487395 ]

[1.7980226 ]

[1.8259779 ]

[0.798749 ]

[1.5590528 ]

[1.8810157 ]

[1.617749 ]

[0.94139415]

[0.5480608 ]

[1.0712168 ]]

In [32]:

train\_acc **=** model**.**evaluate(x\_train, y\_train, verbose**=**0)[1]

print(train\_acc)

0.9437500238418579

In [33]:

test\_acc **=** model**.**evaluate(x\_test, y\_test, verbose**=**0)[1]

print(test\_acc)

0.9125000238418579

In [34]:

ann\_pred **=** model**.**predict(x\_test)

ann\_pred **=** (ann\_pred**>**0.5)

ann\_pred\_tr **=** (ann\_pred\_tr**>**0.5)

3/3 [==============================] - 0s 3ms/step

In [35]:

ann\_pred

Out[35]:

array([[ True],

[ True],

[ True],

[ True],

[ True],

[ True],

[False],

[ 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],

[ True],

[ True],

[ True],

[ True],

[ True],

[ True],

[ True],

[False],

[ True],

[ True],

[ True],

[ True],

[ True],

[ True],

[ True],

[ True],

[ True],

[ True],

[ True],

[ True],

[ True],

[ True],

[ True],

[ True]])

In [37]:

print("ANN Model \n")

print("Train Prediction \n")

print("Confusion Matrix \n", confusion\_matrix(y\_train, ann\_pred\_tr))

print("\n Classification Report \n", classification\_report(y\_train, ann\_pred\_tr))

print("Prediction \n")

print("Confusion Matrix \n",confusion\_matrix(y\_test, ann\_pred))

print("\nClassification Report \n", classification\_report(y\_test, ann\_pred))

ANN Model

Train Prediction

Confusion Matrix

[[ 7 18]

[ 0 295]]

Classification Report

precision recall f1-score support

False 1.00 0.28 0.44 25

True 0.94 1.00 0.97 295

accuracy 0.94 320

macro avg 0.97 0.64 0.70 320

weighted avg 0.95 0.94 0.93 320

Prediction

Confusion Matrix

[[ 3 7]

[ 0 70]]

Classification Report

precision recall f1-score support

False 1.00 0.30 0.46 10

True 0.91 1.00 0.95 70

accuracy 0.91 80

macro avg 0.95 0.65 0.71 80

weighted avg 0.92 0.91 0.89 80

In [38]:

*# save and copy the saved model to mounted google drive*

model**.**save('model.h5')

**%cp** '/content/model.h5' '/content/drive/My Drive/university\_admission\_prediction/training/'