Importing Libraries

In [1]:

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

Loading the Datasets

In [4]:

```
data = pd.read_csv('Admission_Predict.csv')
data
```

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
...	...	...	...	...	...	...	...	...	...
395	396	324	110	3	3.5	3.5	9.04	1	0.82
396	397	325	107	3	3.0	3.5	9.11	1	0.84
397	398	330	116	4	5.0	4.5	9.45	1	0.91
398	399	312	103	3	3.5	4.0	8.78	0	0.67
399	400	333	117	4	5.0	4.0	9.66	1	0.95

400 rows × 9 columns

Training and Testing the Model

In [7]:

```
x=data.drop(['Chance of Admit '],axis=1) #input data_set
y=data['Chance of Admit '] #output labels
```

In [8]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.15)
```

In [9]:

```
x_train
```

Out[9]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
155	156	312	109	3	3.0	3.0	8.69	0
254	255	321	114	4	4.0	5.0	9.12	0
260	261	327	108	5	5.0	3.5	9.13	1
257	258	324	100	3	4.0	5.0	8.64	1
181	182	305	107	2	2.5	2.5	8.42	0
...	...	...	...	...	...	...	...	...
326	327	299	100	3	2.0	2.0	8.02	0
145	146	320	113	2	2.0	2.5	8.64	1
22	23	328	116	5	5.0	5.0	9.50	1
18	19	318	110	3	4.0	3.0	8.80	0
43	44	332	117	4	4.5	4.0	9.10	0

340 rows × 8 columns

In [10]:

```
y_train
```

```
Out[10]: 155    0.77
         254    0.85
         260    0.87
         257    0.78
         181    0.71
                ...
         326    0.63
         145    0.81
         22     0.94
         18     0.63
         43     0.87
         Name: Chance of Admit , Length: 340, dtype: float64
```

```
In [11]: x_test
```

Out[11]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
121	122	334	119	5	4.5	4.5	9.48	1
199	200	313	107	3	4.0	4.5	8.69	0
365	366	330	114	4	4.5	3.0	9.17	1
87	88	317	107	2	3.5	3.0	8.28	0
175	176	320	111	4	4.5	3.5	8.87	1
317	318	300	99	1	1.0	2.5	8.01	0
47	48	339	119	5	4.5	4.0	9.70	0
28	29	295	93	1	2.0	2.0	7.20	0
30	31	300	97	2	3.0	3.0	8.10	1
102	103	314	106	2	4.0	3.5	8.25	0
385	386	335	117	5	5.0	5.0	9.82	1
337	338	332	118	5	5.0	5.0	9.47	1
206	207	315	99	2	3.5	3.0	7.89	0
316	317	298	101	2	1.5	2.0	7.86	0
62	63	304	105	2	3.0	3.0	8.20	1

174	175	321	111	4	4.0	4.0	8.97	1
319	320	327	113	4	3.5	3.0	8.69	1
190	191	324	111	5	4.5	4.0	9.16	1
220	221	313	103	3	4.0	4.0	8.75	0
168	169	293	97	2	2.0	4.0	7.80	1
65	66	325	112	4	3.5	3.5	8.92	0
49	50	327	111	4	3.0	4.0	8.40	1
330	331	327	113	3	3.5	3.0	8.66	1
165	166	322	110	5	4.5	4.0	8.97	0
204	205	298	105	3	3.5	4.0	8.54	0
89	90	316	109	4	4.5	3.5	8.76	1
202	203	340	120	5	4.5	4.5	9.91	1
157	158	309	104	2	2.0	2.5	8.26	0
184	185	316	106	2	2.5	4.0	8.32	0
12	13	328	112	4	4.0	4.5	9.10	1
221	222	316	110	3	3.5	4.0	8.56	0
250	251	320	104	3	3.0	2.5	8.57	1
308	309	312	108	3	3.5	3.0	8.53	0
111	112	321	109	4	4.0	4.0	8.68	1
272	273	294	95	1	1.5	1.5	7.64	0
143	144	340	120	4	4.5	4.0	9.92	1
314	315	305	105	2	3.0	4.0	8.13	0
163	164	317	105	3	3.5	3.0	8.56	0
214	215	331	117	4	4.5	5.0	9.42	1
0	1	337	118	4	4.5	4.5	9.65	1
61	62	307	101	3	4.0	3.0	8.20	0

348	349	302	99	1	2.0	2.0	7.25	0
183	184	314	110	3	4.0	4.0	8.80	0
277	278	320	101	2	2.5	3.0	8.62	0
307	308	325	112	4	4.0	4.0	9.00	1
17	18	319	106	3	4.0	3.0	8.00	1
82	83	320	110	5	5.0	4.5	9.22	1
301	302	319	108	2	2.5	3.0	8.76	0
350	351	318	107	3	3.0	3.5	8.27	1
38	39	304	105	1	3.0	1.5	7.50	0
201	202	315	110	2	3.5	3.0	8.46	1
259	260	331	119	4	5.0	4.5	9.34	1
291	292	300	102	2	1.5	2.0	7.87	0
41	42	316	105	2	2.5	2.5	8.20	1
116	117	299	102	3	4.0	3.5	8.62	0
299	300	305	112	3	3.0	3.5	8.65	0
188	189	331	115	5	4.5	3.5	9.36	1
269	270	308	108	4	4.5	5.0	8.34	0
129	130	333	118	5	5.0	5.0	9.35	1
320	321	317	106	3	4.0	3.5	8.50	1

In [12]:

```
y_test
```

Out[12]:

```
121    0.94
199    0.72
365    0.86
87     0.66
175    0.85
317    0.58
47     0 89
```

```
27     0.46
30     0.65
102    0.62
385    0.96
337    0.94
206    0.63
316    0.54
62     0.54
174    0.87
319    0.80
190    0.90
220    0.76
168    0.64
65     0.55
49     0.78
330    0.80
165    0.78
204    0.69
89     0.74
202    0.97
157    0.65
184    0.72
12     0.78
221    0.75
250    0.74
308    0.69
111    0.69
272    0.49
143    0.97
314    0.66
163    0.68
214    0.94
0      0.92
61     0.47
348    0.57
183    0.75
277    0.70
307    0.80
17     0.65
82     0.92
301    0.66
350    0.74
38     0.52
201    0.72
```

```
259    0.90
291    0.56
41     0.49
116    0.56
299    0.71
188    0.93
269    0.77
129    0.92
320    0.75
Name: Chance of Admit , dtype: float64
```

Model Evaluation

In [13]:

```
from sklearn.ensemble import GradientBoostingRegressor
model = GradientBoostingRegressor()
model.fit(x_train,y_train)
```

Out[13]:

```
GradientBoostingRegressor()
```

In [14]:

```
model.score(x_test,y_test)
```

Out[14]:

```
0.9001495780119302
```

In [16]:

```
y_predict=model.predict(x_test)
```

In [17]:

```
from sklearn.metrics import mean_squared_error, r2_score,mean_absolute_error
import numpy as np
print('Mean Absolute Error:', mean_absolute_error(y_test, y_predict))
print('Mean Squared Error:', mean_squared_error(y_test, y_predict))
print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_predict)))
```

```
Mean Absolute Error: 0.032036490150839
Mean Squared Error: 0.0019843024735634158
Root Mean Squared Error: 0.044545510139220715
```

In [18]:

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

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

In [19]:

```
from sklearn.linear_model._logistic import LogisticRegression

lore = LogisticRegression(random_state=0, max_iter=1000)

lr = lore.fit(x_train, y_train)
```

In [20]:

```
y_pred = lr.predict(x_test)
```

In [ ]:

```

```