importing libraries

In [1]:

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

In [2]:

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

Out[2]:

	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
5	6	330	115	5	4.5	3.0	9.34	1	0.90
6	7	321	109	3	3.0	4.0	8.20	1	0.75
7	8	308	101	2	3.0	4.0	7.90	0	0.68
8	9	302	102	1	2.0	1.5	8.00	0	0.50
9	10	323	108	3	3.5	3.0	8.60	0	0.45
10	11	325	106	3	3.5	4.0	8.40	1	0.52
11	12	327	111	4	4.0	4.5	9.00	1	0.84
12	13	328	112	4	4.0	4.5	9.10	1	0.78
13	14	307	109	3	4.0	3.0	8.00	1	0.62
14	15	311	104	3	3.5	2.0	8.20	1	0.61
15	16	314	105	3	3.5	2.5	8.30	0	0.54
16	17	317	107	3	4.0	3.0	8.70	0	0.66
17	18	319	106	3	4.0	3.0	8.00	1	0.65
18	19	318	110	3	4.0	3.0	8.80	0	0.63
19	20	303	102	3	3.5	3.0	8.50	0	0.62
20	21	312	107	3	3.0	2.0	7.90	1	0.64
21	22	325	114	4	3.0	2.0	8.40	0	0.70
22	23	328	116	5	5.0	5.0	9.50	1	0.94
23	24	334	119	5	5.0	4.5	9.70	1	0.95
24	25	336	119	5	4.0	3.5	9.80	1	0.97
25	26	340	120	5	4.5	4.5	9.60	1	0.94
26	27	322	109	5	4.5	3.5	8.80	0	0.76
27	28	298	98	2	1.5	2.5	7.50	1	0.44
28	29	295	93	1	2.0	2.0	7.20	0	0.46
29	30	310	99	2	1.5	2.0	7.30	0	0.54
...	...	...	...	...	...	...	...	...	...
370	371	310	103	2	2.5	2.5	8.24	0	0.72
371	372	324	110	3	3.5	3.0	9.22	1	0.89
372	373	336	119	4	4.5	4.0	9.62	1	0.95

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
373	374	321	109	3	3.0	3.0	8.54	1	0.79
374	375	315	105	2	2.0	2.5	7.65	0	0.39
375	376	304	101	2	2.0	2.5	7.66	0	0.38
376	377	297	96	2	2.5	2.0	7.43	0	0.34
377	378	290	100	1	1.5	2.0	7.56	0	0.47
378	379	303	98	1	2.0	2.5	7.65	0	0.56
379	380	311	99	1	2.5	3.0	8.43	1	0.71
380	381	322	104	3	3.5	4.0	8.84	1	0.78
381	382	319	105	3	3.0	3.5	8.67	1	0.73
382	383	324	110	4	4.5	4.0	9.15	1	0.82
383	384	300	100	3	3.0	3.5	8.26	0	0.62
384	385	340	113	4	5.0	5.0	9.74	1	0.96
385	386	335	117	5	5.0	5.0	9.82	1	0.96
386	387	302	101	2	2.5	3.5	7.96	0	0.46
387	388	307	105	2	2.0	3.5	8.10	0	0.53
388	389	296	97	2	1.5	2.0	7.80	0	0.49
389	390	320	108	3	3.5	4.0	8.44	1	0.76
390	391	314	102	2	2.0	2.5	8.24	0	0.64
391	392	318	106	3	2.0	3.0	8.65	0	0.71
392	393	326	112	4	4.0	3.5	9.12	1	0.84
393	394	317	104	2	3.0	3.0	8.76	0	0.77
394	395	329	111	4	4.5	4.0	9.23	1	0.89
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

In [3]:

```
df.head()
```

Out[3]:

	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 [4]:

```
df.drop('Serial No.', axis = 1,inplace =True)
df
```

Out[4]:

	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

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
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
5	330	115	5	4.5	3.0	9.34	1	0.90
6	321	109	3	3.0	4.0	8.20	1	0.75
7	308	101	2	3.0	4.0	7.90	0	0.68
8	302	102	1	2.0	1.5	8.00	0	0.50
9	323	108	3	3.5	3.0	8.60	0	0.45
10	325	106	3	3.5	4.0	8.40	1	0.52
11	327	111	4	4.0	4.5	9.00	1	0.84
12	328	112	4	4.0	4.5	9.10	1	0.78
13	307	109	3	4.0	3.0	8.00	1	0.62
14	311	104	3	3.5	2.0	8.20	1	0.61
15	314	105	3	3.5	2.5	8.30	0	0.54
16	317	107	3	4.0	3.0	8.70	0	0.66
17	319	106	3	4.0	3.0	8.00	1	0.65
18	318	110	3	4.0	3.0	8.80	0	0.63
19	303	102	3	3.5	3.0	8.50	0	0.62
20	312	107	3	3.0	2.0	7.90	1	0.64
21	325	114	4	3.0	2.0	8.40	0	0.70
22	328	116	5	5.0	5.0	9.50	1	0.94
23	334	119	5	5.0	4.5	9.70	1	0.95
24	336	119	5	4.0	3.5	9.80	1	0.97
25	340	120	5	4.5	4.5	9.60	1	0.94
26	322	109	5	4.5	3.5	8.80	0	0.76
27	298	98	2	1.5	2.5	7.50	1	0.44
28	295	93	1	2.0	2.0	7.20	0	0.46
29	310	99	2	1.5	2.0	7.30	0	0.54
...	...	...	...	...	...	...	...	...
370	310	103	2	2.5	2.5	8.24	0	0.72
371	324	110	3	3.5	3.0	9.22	1	0.89
372	336	119	4	4.5	4.0	9.62	1	0.95
373	321	109	3	3.0	3.0	8.54	1	0.79
374	315	105	2	2.0	2.5	7.65	0	0.39
375	304	101	2	2.0	2.5	7.66	0	0.38
376	297	96	2	2.5	2.0	7.43	0	0.34
377	290	100	1	1.5	2.0	7.56	0	0.47
378	303	98	1	2.0	2.5	7.65	0	0.56
379	311	99	1	2.5	3.0	8.43	1	0.71
380	322	104	3	3.5	4.0	8.84	1	0.78
381	319	105	3	3.0	3.5	8.67	1	0.73
382	324	110	4	4.5	4.0	9.15	1	0.82
383	300	100	3	3.0	3.5	8.26	0	0.62
384	340	113	4	5.0	5.0	9.74	1	0.96
385	335	117	5	5.0	5.0	9.82	1	0.96
386	302	101	2	2.5	3.5	7.96	0	0.46
387	307	105	2	2.0	3.5	8.10	0	0.53
388	296	97	2	1.5	2.0	7.80	0	0.49
389	320	108	3	3.5	4.0	8.44	1	0.76
390	314	102	2	2.0	2.5	8.24	0	0.64

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
391	318	106	3	2.0	3.0	8.65	0	0.71
392	326	112	4	4.0	3.5	9.12	1	0.84
393	317	104	2	3.0	3.0	8.76	0	0.77
394	329	111	4	4.5	4.0	9.23	1	0.89
395	324	110	3	3.5	3.5	9.04	1	0.82
396	325	107	3	3.0	3.5	9.11	1	0.84
397	330	116	4	5.0	4.5	9.45	1	0.91
398	312	103	3	3.5	4.0	8.78	0	0.67
399	333	117	4	5.0	4.0	9.66	1	0.95

400 rows × 8 columns

In [5]:

```
df.isnull().sum()
```

Out[5]:

```
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 8 columns):
GRE Score            400 non-null int64
TOEFL Score          400 non-null int64
University Rating    400 non-null int64
SOP                  400 non-null float64
LOR                  400 non-null float64
CGPA                 400 non-null float64
Research             400 non-null int64
Chance of Admit      400 non-null float64
dtypes: float64(4), int64(4)
memory usage: 25.1 KB
```

In [7]:

```
df.describe()
```

Out[7]:

	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 [8]:

```
uni = df.groupby(by = 'University Rating').mean()
uni
```

Out[8]:

	GRE Score	TOEFL Score	SOP	LOR	CGPA	Research	Chance of Admit
University Rating							
1	303.153846	99.076923	1.884615	2.211538	7.745769	0.192308	0.548077
2	309.177570	103.523364	2.705607	2.925234	8.183738	0.299065	0.625981
3	315.954887	106.887218	3.364662	3.402256	8.552256	0.533835	0.711880
4	324.824324	111.824324	4.108108	4.006757	9.021622	0.797297	0.818108
5	328.333333	113.666667	4.500000	4.358333	9.291167	0.866667	0.888167

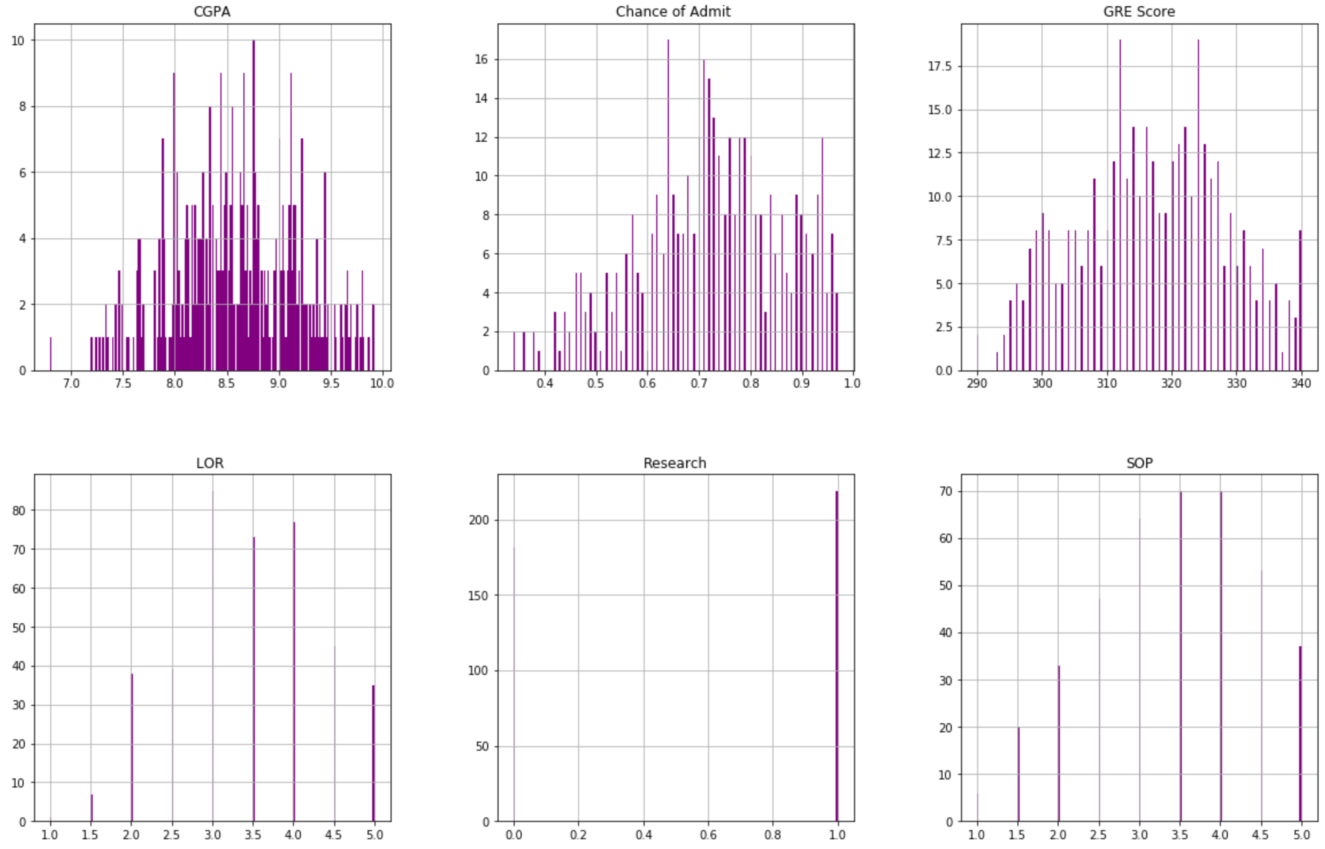
Data Visualization

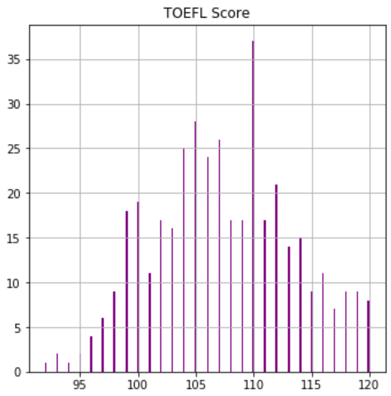
In [9]:

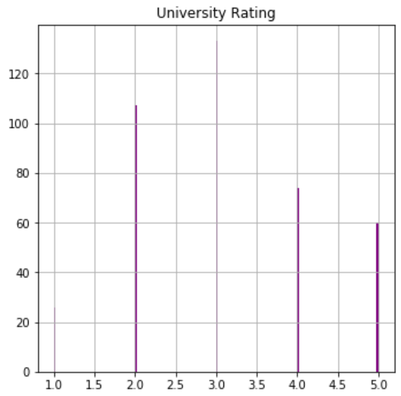
```
df.hist(bins = 200,figsize=(20,20),color='purple')
```

Out[9]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000002902188D2B0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029021902438>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029021929898>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000029021951E10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029021980358>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000290219A38D0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000290219CCE48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000290219FE438>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000290219FE470>]],
      dtype=object)
```





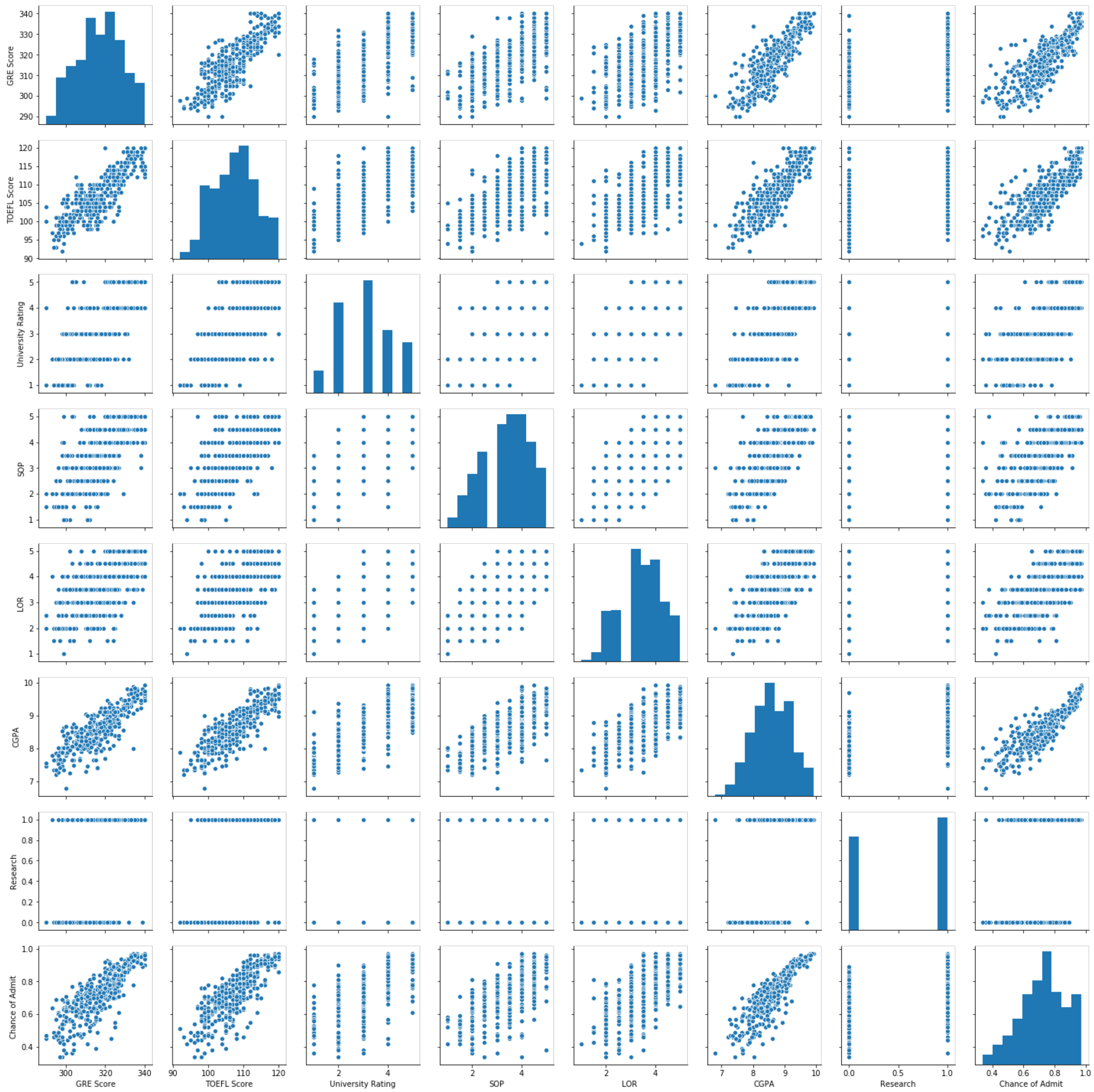


In [10]:

```
sns.pairplot(df)
```

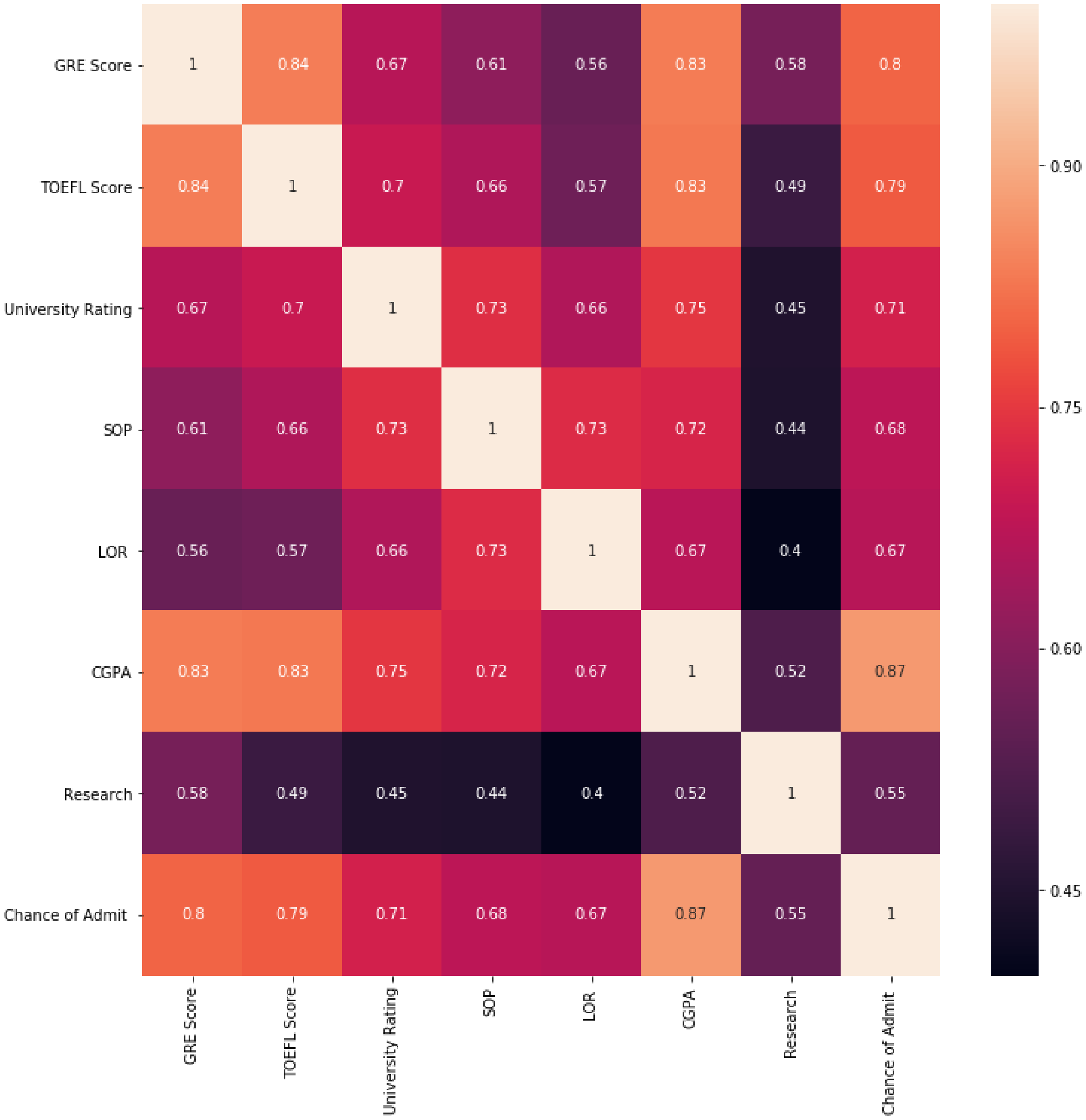
Out[10]:

```
<seaborn.axisgrid.PairGrid at 0x290244ff470>
```



In [11]:

```
corr_matrix=df.corr()
plt.figure(figsize=(12,12))
sns.heatmap(corr_matrix,annot=True)
plt.show()
```



create ,training and testing dataset

In [12]:

```
df.columns
```

Out[12]:

```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',
       'Research', 'Chance of Admit '],
      dtype='object')
```

In [13]:

```
x = df.drop('Chance of Admit ',axis =1 )
x
```

Out[13]:

	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
5	330	115	5	4.5	3.0	9.34	1
6	321	109	3	3.0	4.0	8.20	1
7	308	101	2	3.0	4.0	7.90	0
8	302	102	1	2.0	1.5	8.00	0
9	323	108	3	3.5	3.0	8.60	0
10	325	106	3	3.5	4.0	8.40	1
11	327	111	4	4.0	4.5	9.00	1
12	328	112	4	4.0	4.5	9.10	1
13	307	109	3	4.0	3.0	8.00	1
14	311	104	3	3.5	2.0	8.20	1
15	314	105	3	3.5	2.5	8.30	0
16	317	107	3	4.0	3.0	8.70	0
17	319	106	3	4.0	3.0	8.00	1
18	318	110	3	4.0	3.0	8.80	0
19	303	102	3	3.5	3.0	8.50	0
20	312	107	3	3.0	2.0	7.90	1
21	325	114	4	3.0	2.0	8.40	0
22	328	116	5	5.0	5.0	9.50	1
23	334	119	5	5.0	4.5	9.70	1
24	336	119	5	4.0	3.5	9.80	1
25	340	120	5	4.5	4.5	9.60	1
26	322	109	5	4.5	3.5	8.80	0
27	298	98	2	1.5	2.5	7.50	1
28	295	93	1	2.0	2.0	7.20	0
29	310	99	2	1.5	2.0	7.30	0
...	...	...	...	...	...	...	...
370	310	103	2	2.5	2.5	8.24	0
371	324	110	3	3.5	3.0	9.22	1
372	336	119	4	4.5	4.0	9.62	1
373	321	109	3	3.0	3.0	8.54	1
374	315	105	2	2.0	2.5	7.65	0
375	304	101	2	2.0	2.5	7.66	0
376	297	96	2	2.5	2.0	7.43	0
377	290	100	1	1.5	2.0	7.56	0
378	303	98	1	2.0	2.5	7.65	0
379	311	99	1	2.5	3.0	8.43	1
380	322	104	3	3.5	4.0	8.84	1
381	319	105	3	3.0	3.5	8.67	1
382	324	110	4	4.5	4.0	9.15	1
383	300	100	3	3.0	3.5	8.26	0
384	340	113	4	5.0	5.0	9.74	1
385	335	117	5	5.0	5.0	9.82	1
386	302	101	2	2.5	3.5	7.96	0
387	307	105	2	2.0	3.5	8.10	0

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
388	296	97	2	1.5	2.0	7.80	0
389	320	108	3	3.5	4.0	8.44	1
390	314	102	2	2.0	2.5	8.24	0
391	318	106	3	2.0	3.0	8.65	0
392	326	112	4	4.0	3.5	9.12	1
393	317	104	2	3.0	3.0	8.76	0
394	329	111	4	4.5	4.0	9.23	1
395	324	110	3	3.5	3.5	9.04	1
396	325	107	3	3.0	3.5	9.11	1
397	330	116	4	5.0	4.5	9.45	1
398	312	103	3	3.5	4.0	8.78	0
399	333	117	4	5.0	4.0	9.66	1

400 rows × 7 columns

In [14]:

```
y = df['Chance of Admit ']
y
```

Out[14]:

```
0      0.92
1      0.76
2      0.72
3      0.80
4      0.65
5      0.90
6      0.75
7      0.68
8      0.50
9      0.45
10     0.52
11     0.84
12     0.78
13     0.62
14     0.61
15     0.54
16     0.66
17     0.65
18     0.63
19     0.62
20     0.64
21     0.70
22     0.94
23     0.95
24     0.97
25     0.94
26     0.76
27     0.44
28     0.46
29     0.54
       ... 
370    0.72
371    0.89
372    0.95
373    0.79
374    0.39
375    0.38
376    0.34
377    0.47
378    0.56
379    0.71
380    0.78
381    0.73
382    0.82
383    0.62
384    0.96
385    0.96
386    0.46
```

```
386    0.46
387    0.53
388    0.49
389    0.76
390    0.64
391    0.71
392    0.84
393    0.77
394    0.89
395    0.82
396    0.84
397    0.91
398    0.67
399    0.95
Name: Chance of Admit , Length: 400, dtype: float64
```

In [15]:

```
x.shape
```

Out[15]:

```
(400, 7)
```

In [16]:

```
y.shape
```

Out[16]:

```
(400,)
```

In [17]:

```
X = np.array(x)
X
```

Out[17]:

```
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.  ]])
```

In [18]:

```
Y = np.array(y)
Y
```

Out[18]:

```
array([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,
       0.7 , 0.72, 0.73, 0.72, 0.97, 0.97, 0.69, 0.57, 0.65, 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,
       0.78, 0.89, 0.7 , 0.66, 0.67, 0.68, 0.8 , 0.81, 0.8 , 0.94, 0.93,
       0.92, 0.89, 0.82, 0.79, 0.58, 0.56, 0.56, 0.64, 0.61, 0.68, 0.76,
       0.86, 0.9 , 0.71, 0.62, 0.66, 0.65, 0.73, 0.62, 0.74, 0.79, 0.8 ,
       0.69, 0.7 , 0.76, 0.84, 0.78, 0.67, 0.66, 0.65, 0.54, 0.58, 0.79,
       0.8 , 0.75, 0.73, 0.72, 0.62, 0.67, 0.81, 0.63, 0.69, 0.8 , 0.43,
       0.8 , 0.73, 0.75, 0.71, 0.73, 0.83, 0.72, 0.94, 0.81, 0.81, 0.75,
       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])
```

In [19]:

```
Y = Y.reshape(-1,1)
Y.shape
```

Out[19]:

```
(400, 1)
```

In [20]:

```
from sklearn.preprocessing import StandardScaler,MinMaxScaler
scalar_x = StandardScaler()
X = scalar_x.fit_transform(X)
```

In [21]:

```
scalar_y = StandardScaler()
Y = scalar_y.fit_transform(Y)
```

In [22]:

```
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)
```

Train and Evaluate a linear regression model

In [23]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,accuracy_score
```

In [24]:

```
reg = LinearRegression()
reg.fit(X_train,Y_train)
```

Out[24]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [25]:

```
acc = reg.score(X_test,Y_test)
acc
```

Out[25]:

```
0.794036289144673
```

Train and Evaluate an Artificial Network Model

In [26]:

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

In [29]:

```
mo = keras.Sequential()
mo.add(Dense(50,input_dim=7))
mo.add(Activation('relu'))


mo.add(Dense(150))
mo.add(Activation('relu'))
mo.add(Dropout(0.5))

mo.add(Dense(150))
mo.add(Activation('relu'))
mo.add(Dropout(0.5))

mo.add(Dense(50))
mo.add(Activation('linear'))
mo.add(Dense(1))

mo.compile(loss='mse',optimizer='adam')
mo.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_8 (Dense)              (None, 50)                400       
_________________________________________________________________
activation_8 (Activation)    (None, 50)                0         
_________________________________________________________________
dense_9 (Dense)              (None, 150)               7650      
_________________________________________________________________
activation_9 (Activation)    (None, 150)               0         
_________________________________________________________________
dropout_6 (Dropout)          (None, 150)               0         
_________________________________________________________________
dense_10 (Dense)             (None, 150)               22650     
_________________________________________________________________
activation_10 (Activation)   (None, 150)               0         
_________________________________________________________________
dropout_7 (Dropout)          (None, 150)               0         
_________________________________________________________________
dense_11 (Dense)             (None, 50)                7550      
_________________________________________________________________
activation_11 (Activation)   (None, 50)                0         
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 51        
=================================================================
Total params: 38,301
Trainable params: 38,301
Non-trainable params: 0
_________________________________________________________________
```

In [30]:

```
mo.compile(optimizer = 'Adam',loss='mean_squared_error')
```

In [31]:

```
ep =mo.fit(X_train,Y_train,epochs=100,batch_size=20)
```

```
result=mo.evaluate(X_test,Y_test)
acc=1-result
print("Accuracy : {}".format(acc))
```

```
WARNING:tensorflow:From C:\Users\Anshal\Anaconda3\lib\site-
packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/100
340/340 [==============================] - 4s 13ms/sample - loss: 0.6472
Epoch 2/100
340/340 [==============================] - 0s 284us/sample - loss: 0.3395
Epoch 3/100
340/340 [==============================] - 0s 420us/sample - loss: 0.3195
Epoch 4/100
340/340 [==============================] - 0s 326us/sample - loss: 0.2817
Epoch 5/100
340/340 [==============================] - 0s 265us/sample - loss: 0.2686
Epoch 6/100
340/340 [==============================] - 0s 421us/sample - loss: 0.2974
Epoch 7/100
340/340 [==============================] - 0s 323us/sample - loss: 0.2868
Epoch 8/100
340/340 [==============================] - 0s 261us/sample - loss: 0.2395
Epoch 9/100
340/340 [==============================] - 0s 325us/sample - loss: 0.2332
Epoch 10/100
340/340 [==============================] - 0s 449us/sample - loss: 0.2390
Epoch 11/100
340/340 [==============================] - 0s 1ms/sample - loss: 0.2281
Epoch 12/100
340/340 [==============================] - 0s 827us/sample - loss: 0.2375
Epoch 13/100
340/340 [==============================] - 0s 557us/sample - loss: 0.2408
Epoch 14/100
340/340 [==============================] - 0s 640us/sample - loss: 0.2186
Epoch 15/100
340/340 [==============================] - 0s 604us/sample - loss: 0.2383
Epoch 16/100
340/340 [==============================] - 0s 303us/sample - loss: 0.2283
Epoch 17/100
340/340 [==============================] - 0s 317us/sample - loss: 0.2554
Epoch 18/100
340/340 [==============================] - 0s 397us/sample - loss: 0.2179
Epoch 19/100
340/340 [==============================] - 0s 282us/sample - loss: 0.2015
Epoch 20/100
340/340 [==============================] - 0s 311us/sample - loss: 0.2021
Epoch 21/100
340/340 [==============================] - 0s 288us/sample - loss: 0.2166
Epoch 22/100
340/340 [==============================] - 0s 273us/sample - loss: 0.2388
Epoch 23/100
340/340 [==============================] - 0s 308us/sample - loss: 0.2132
Epoch 24/100
340/340 [==============================] - 0s 327us/sample - loss: 0.2164
Epoch 25/100
340/340 [==============================] - 0s 324us/sample - loss: 0.2076
Epoch 26/100
340/340 [==============================] - 0s 267us/sample - loss: 0.2052
Epoch 27/100
340/340 [==============================] - 0s 282us/sample - loss: 0.1708
Epoch 28/100
340/340 [==============================] - 0s 458us/sample - loss: 0.2152
Epoch 29/100
340/340 [==============================] - 0s 285us/sample - loss: 0.1860
Epoch 30/100
340/340 [==============================] - 0s 289us/sample - loss: 0.1936
Epoch 31/100
340/340 [==============================] - 0s 297us/sample - loss: 0.1880
Epoch 32/100
340/340 [==============================] - 0s 268us/sample - loss: 0.2083
Epoch 33/100
340/340 [==============================] - 0s 293us/sample - loss: 0.2196
Epoch 34/100
340/340 [==============================] - 0s 514us/sample - loss: 0.1997
```

```
340/340 [==============================] - 0s 314us/sample - loss: 0.1997
Epoch 35/100
340/340 [==============================] - 0s 343us/sample - loss: 0.1994
Epoch 36/100
340/340 [==============================] - 0s 337us/sample - loss: 0.1992
Epoch 37/100
340/340 [==============================] - 0s 418us/sample - loss: 0.1789
Epoch 38/100
340/340 [==============================] - 0s 323us/sample - loss: 0.1935
Epoch 39/100
340/340 [==============================] - 0s 293us/sample - loss: 0.1797
Epoch 40/100
340/340 [==============================] - 0s 276us/sample - loss: 0.1793
Epoch 41/100
340/340 [==============================] - 0s 293us/sample - loss: 0.1631
Epoch 42/100
340/340 [==============================] - 0s 324us/sample - loss: 0.1720
Epoch 43/100
340/340 [==============================] - 0s 466us/sample - loss: 0.1930
Epoch 44/100
340/340 [==============================] - 0s 290us/sample - loss: 0.1885
Epoch 45/100
340/340 [==============================] - 0s 311us/sample - loss: 0.1835
Epoch 46/100
340/340 [==============================] - 0s 396us/sample - loss: 0.1726
Epoch 47/100
340/340 [==============================] - 0s 390us/sample - loss: 0.1896
Epoch 48/100
340/340 [==============================] - 0s 443us/sample - loss: 0.1748
Epoch 49/100
340/340 [==============================] - 0s 306us/sample - loss: 0.1826
Epoch 50/100
340/340 [==============================] - 0s 316us/sample - loss: 0.1493
Epoch 51/100
340/340 [==============================] - 0s 295us/sample - loss: 0.1620
Epoch 52/100
340/340 [==============================] - 0s 424us/sample - loss: 0.1720
Epoch 53/100
340/340 [==============================] - 0s 300us/sample - loss: 0.1519
Epoch 54/100
340/340 [==============================] - 0s 337us/sample - loss: 0.1721
Epoch 55/100
340/340 [==============================] - 0s 432us/sample - loss: 0.1676s - loss: 0.12
Epoch 56/100
340/340 [==============================] - 0s 276us/sample - loss: 0.1751
Epoch 57/100
340/340 [==============================] - 0s 317us/sample - loss: 0.1700
Epoch 58/100
340/340 [==============================] - 0s 305us/sample - loss: 0.1662
Epoch 59/100
340/340 [==============================] - 0s 261us/sample - loss: 0.1524
Epoch 60/100
340/340 [==============================] - 0s 353us/sample - loss: 0.1819
Epoch 61/100
340/340 [==============================] - 0s 396us/sample - loss: 0.1738
Epoch 62/100
340/340 [==============================] - 0s 374us/sample - loss: 0.1736
Epoch 63/100
340/340 [==============================] - 0s 185us/sample - loss: 0.1668
Epoch 64/100
340/340 [==============================] - 0s 160us/sample - loss: 0.1583
Epoch 65/100
340/340 [==============================] - 0s 337us/sample - loss: 0.1682
Epoch 66/100
340/340 [==============================] - 0s 642us/sample - loss: 0.1591
Epoch 67/100
340/340 [==============================] - 0s 291us/sample - loss: 0.1357
Epoch 68/100
340/340 [==============================] - 0s 308us/sample - loss: 0.1482
Epoch 69/100
340/340 [==============================] - 0s 280us/sample - loss: 0.1541
Epoch 70/100
340/340 [==============================] - 0s 518us/sample - loss: 0.1549
Epoch 71/100
340/340 [==============================] - 0s 440us/sample - loss: 0.1499
Epoch 72/100
340/340 [==============================] - 0s 782us/sample - loss: 0.1338
Epoch 73/100
Epoch 73/100
340/340 [==============================] - 0s 259us/sample - loss: 0.1504
Epoch 74/100
340/340 [==============================] - 0s 694us/sample - loss: 0.1607
Epoch 75/100
340/340 [==============================] - 0s 1ms/sample - loss: 0.1528
Epoch 76/100
340/340 [==============================] - 0s 290us/sample - loss: 0.1591
Epoch 77/100
340/340 [==============================] - 0s 455us/sample - loss: 0.1593
Epoch 78/100
340/340 [==============================] - 0s 302us/sample - loss: 0.1527
Epoch 79/100
340/340 [==============================] - 0s 252us/sample - loss: 0.1441
Epoch 80/100
340/340 [==============================] - 0s 201us/sample - loss: 0.1358
Epoch 81/100
340/340 [==============================] - 0s 311us/sample - loss: 0.1500
Epoch 82/100
340/340 [==============================] - 0s 293us/sample - loss: 0.1387
Epoch 83/100
340/340 [==============================] - 0s 253us/sample - loss: 0.1348
Epoch 84/100
340/340 [==============================] - 0s 450us/sample - loss: 0.1420
Epoch 85/100
340/340 [==============================] - 0s 305us/sample - loss: 0.1388
Epoch 86/100
340/340 [==============================] - 0s 198us/sample - loss: 0.1352
Epoch 87/100
340/340 [==============================] - 0s 456us/sample - loss: 0.1412
Epoch 88/100
340/340 [==============================] - 0s 414us/sample - loss: 0.1260
Epoch 89/100
340/340 [==============================] - 0s 287us/sample - loss: 0.1401
Epoch 90/100
340/340 [==============================] - 0s 300us/sample - loss: 0.1259
Epoch 91/100
340/340 [==============================] - 0s 294us/sample - loss: 0.1236
Epoch 92/100
340/340 [==============================] - 0s 291us/sample - loss: 0.1182
Epoch 93/100
340/340 [==============================] - 0s 342us/sample - loss: 0.1318
Epoch 94/100
340/340 [==============================] - 0s 311us/sample - loss: 0.1354
Epoch 95/100
340/340 [==============================] - 0s 253us/sample - loss: 0.1161
Epoch 96/100
340/340 [==============================] - 0s 418us/sample - loss: 0.1130
Epoch 97/100
340/340 [==============================] - 0s 320us/sample - loss: 0.1216
Epoch 98/100
340/340 [==============================] - 0s 253us/sample - loss: 0.1244
Epoch 99/100
340/340 [==============================] - 0s 261us/sample - loss: 0.1207
Epoch 100/100
340/340 [==============================] - 0s 249us/sample - loss: 0.1347
60/60 [==============================] - 0s 6ms/sample - loss: 0.2733
Accuracy : 0.7267367045084636
```

In [32]:

```
ep.history.keys()
```

Out[32]:

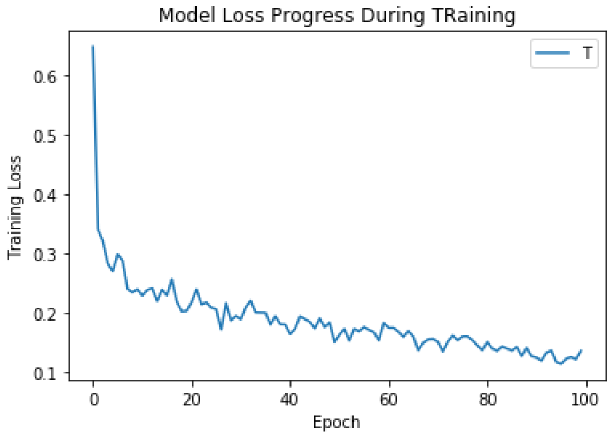
```
dict_keys(['loss'])
```

In [33]:

```
plt.plot(ep.history['loss'])
plt.title('Model Loss Progress During TRaining')
plt.xlabel('Epoch')
plt.ylabel('Training Loss')
plt.legend('Training Loss')
```

Out[33]:

```
<matplotlib.legend.Legend at 0x2902e46b630>
```



Train and Evaluate a Decision Tree and a Random forest models

In [34]:

```
from sklearn.tree import DecisionTreeRegressor
dt =  DecisionTreeRegressor()
dt.fit(X_train,Y_train)
```

Out[34]:

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

In [35]:

```
accc=dt.score(X_test,Y_test)
accc
```

Out[35]:

```
0.6972753142819761
```

In [36]:

```
from sklearn.ensemble import RandomForestRegressor
rm = RandomForestRegressor(n_estimators = 100,max_depth = 10)
rm.fit(X_train,Y_train)
```

```
C:\Users\Anshal\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: DataConversionWarning: A colu
mn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,),
for example using ravel().
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[36]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=10,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

In [39]:

```
ax = rm.score(X_test,Y_test)
```

```
ax
```

Out[39]:

```
0.7266698120311084
```

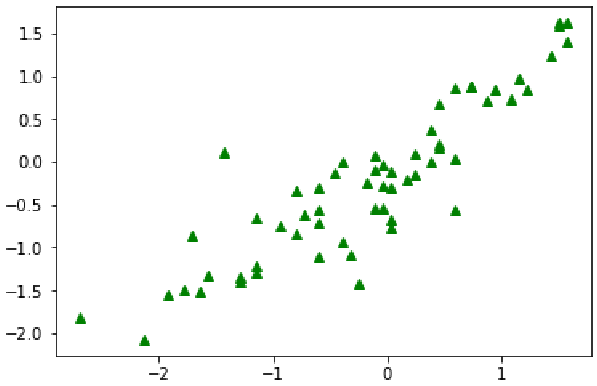
Calculating Regression model KPI's

In [40]:

```
ypr = reg.predict(X_test)
plt.plot(Y_test,ypr,'^',color='green')
```

Out[40]:

```
[<matplotlib.lines.Line2D at 0x29030a03b70>]
```



In [41]:

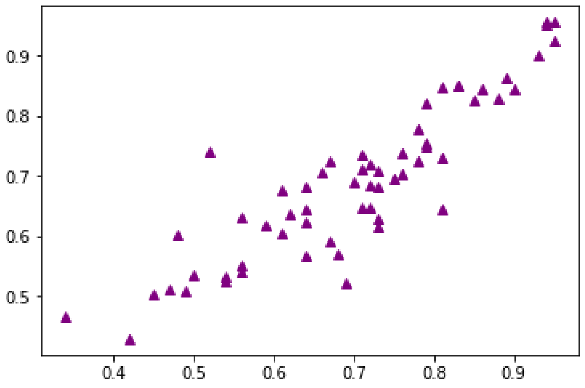
```
ori = scalar_y.inverse_transform(ypr)
tor = scalar_y.inverse_transform(Y_test)
```

In [42]:

```
plt.plot(tor,ori,'^',color='purple')
```

Out[42]:

```
[<matplotlib.lines.Line2D at 0x290308a5dd8>]
```



In [43]:

```
k = X_test.shape[1]
n=len(X_test)
n
```

Out[43]:

```
60
```

```
60
```

In [44]:

```
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
from math import sqrt
```

In [45]:

```
RMSE = float(format(np.sqrt(mean_squared_error(tor,ori)),'.3f'))
MSE = mean_squared_error(tor,ori)
MAE = mean_absolute_error(tor,ori)
r2 = r2_score(tor,ori)
adj_r2 = 1-(1-r2)*(n-1)/(n-k-1)

print('RMSE -',RMSE,'\nMSE -',MSE,'\nMAE -',MAE,'\nR2 -',r2,'\nAdjusted R2 -',adj_r2)
```

```
RMSE - 0.065
MSE - 0.004197860815226231
MAE - 0.04723610074681444
R2 - 0.7940362891446728
Adjusted R2 - 0.7663104049910712
```

In [ ]:

```

```

In [ ]:

```

```