In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\13_placement.csv")
a
```

Out[3]:

	cgpa	placement_exam_marks	placed
0	7.19	26.0	1
1	7.46	38.0	1
2	7.54	40.0	1
3	6.42	8.0	1
4	7.23	17.0	0
...	...	...	...
995	8.87	44.0	1
996	9.12	65.0	1
997	4.89	34.0	0
998	8.62	46.0	1
999	4.90	10.0	1

In [4]:
```
a=a.head(50)
a
```

Out[4]:

	cgpa	placement_exam_marks	placed
0	7.19	26.0	1
1	7.46	38.0	1
2	7.54	40.0	1
3	6.42	8.0	1
4	7.23	17.0	0
5	7.30	23.0	1
6	6.69	11.0	0
7	7.12	39.0	1
8	6.45	38.0	0
9	7.75	94.0	1
10	6.82	16.0	1
11	6.38	7.0	1
12	6.58	16.0	1
13	5.68	26.0	0
14	7.91	43.0	0
15	7.10	21.0	0
16	6.53	19.0	0
17	7.56	22.0	1
18	6.93	27.0	0
19	7.63	29.0	0
20	6.69	47.0	0
21	7.43	33.0	1
22	6.76	54.0	1
23	6.05	11.0	0
24	6.44	11.0	0
25	6.28	58.0	1
26	7.45	8.0	1
27	6.53	46.0	0
28	7.23	19.0	0
29	6.51	15.0	1
30	7.46	16.0	0
31	7.66	44.0	0
32	5.91	11.0	1
33	6.23	27.0	0
34	8.15	9.0	0
35	7.48	12.0	0
36	6.85	16.0	1
37	8.51	9.0	1

	cgpa	placement_exam_marks	placed
38	6.58	20.0	1
39	7.25	17.0	0
40	6.60	86.0	1
41	6.70	38.0	0
42	7.46	71.0	1
43	7.85	63.0	0
44	7.88	55.0	0
45	6.92	10.0	1
46	7.30	15.0	0
47	6.92	46.0	0
48	6.29	42.0	0
49	8.23	28.0	1

In [5]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 3 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   cgpa                 50 non-null     float64
 1   placement_exam_marks  50 non-null     float64
 2   placed               50 non-null     int64  
dtypes: float64(2), int64(1)
memory usage: 1.3 KB
```

In [6]:
```
a.columns
```

Out[6]:
```
Index(['cgpa', 'placement_exam_marks', 'placed'], dtype='object')
```

In [7]:
```
a.head()
```

Out[7]:

	cgpa	placement_exam_marks	placed
0	7.19	26.0	1
1	7.46	38.0	1
2	7.54	40.0	1
3	6.42	8.0	1
4	7.23	17.0	0

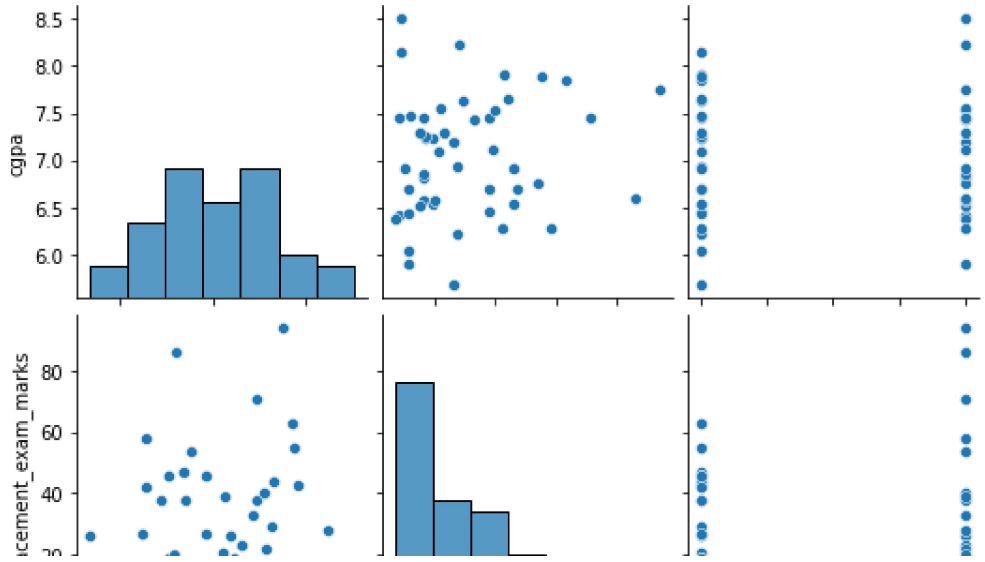
In [8]:
```
a.describe()
```

Out[8]:

	cgpa	placement_exam_marks	placed
count	50.000000	50.000000	50.000000
mean	7.037400	29.940000	0.480000
std	0.629619	20.398489	0.504672
min	5.680000	7.000000	0.000000
25%	6.542500	15.250000	0.000000
50%	7.015000	24.500000	0.000000
75%	7.460000	41.500000	1.000000
max	8.510000	94.000000	1.000000

In [9]:
```
sns.pairplot(a)
```

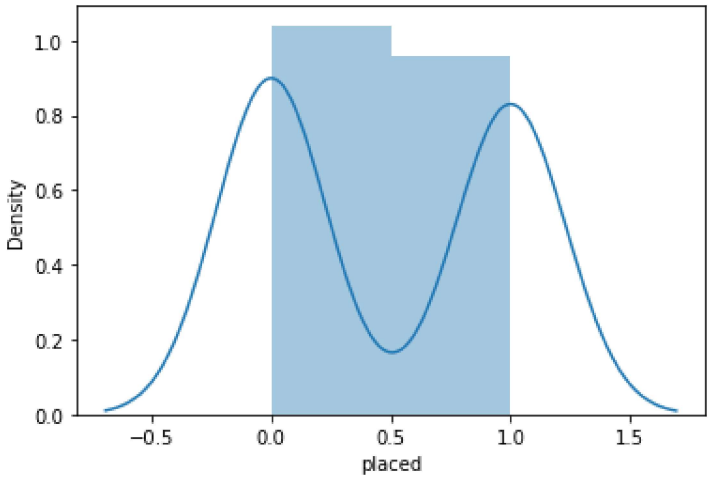
Out[9]: <seaborn.axisgrid.PairGrid at 0x22ea3c6bca0>



In [10]:
```
sns.distplot(a['placed'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

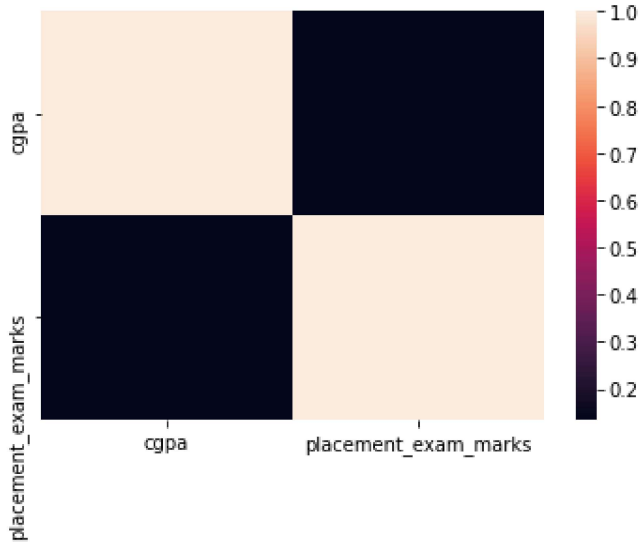
Out[10]: <AxesSubplot:xlabel='placed', ylabel='Density'>



In [11]:
```
x1=a[['cgpa', 'placement_exam_marks']]
```

In [12]:
```
sns.heatmap(x1.corr())
```

Out[12]: <AxesSubplot:>



In [13]:
```
x=a[['cgpa', 'placement_exam_marks']]
y=a['placed']
```

In [14]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [15]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[15]:
```
LinearRegression()
```

In [16]:
```
print(lr.intercept_)
```

```
-0.4274121363355689
```

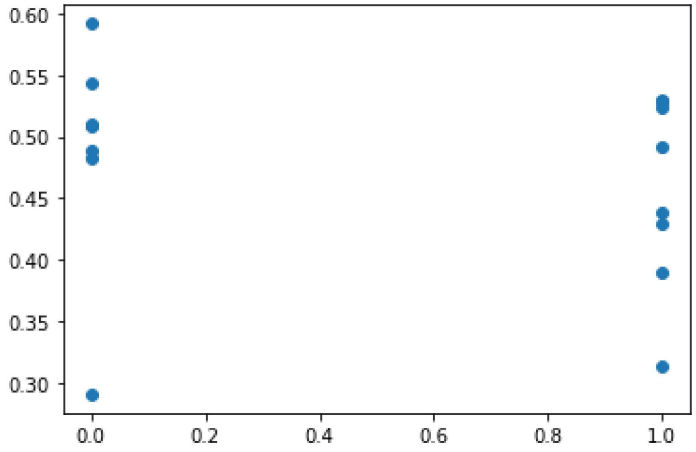
In [17]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[17]:

	Co-efficient
cgpa	0.124675
placement_exam_marks	0.000374

In [18]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[18]:
```
<matplotlib.collections.PathCollection at 0x22ea6360670>
```



In [19]:
```
print(lr.score(x_test,y_test))
```

```
-0.10787488726719108
```

In [20]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[21]:
```
Ridge(alpha=10)
```

In [22]:
```
rr.score(x_test,y_test)
```

Out[22]:
```
-0.06197645191152246
```

In [23]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[23]:
```
Lasso(alpha=10)
```

In [24]:
```
la.score(x_test,y_test)
```

Out[24]:
```
-0.023323615160349753
```

In [29]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[29]:
```
ElasticNet()
```

In [30]:
```
print(en.coef_)
```

```
[0. 0.]
```

In [32]:
```
print(en.intercept_)
```

```
0.45714285714285713
```

In [33]:
```
print(en.predict(x_test))
```

```
[0.45714286 0.45714286 0.45714286 0.45714286 0.45714286 0.45714286
 0.45714286 0.45714286 0.45714286 0.45714286 0.45714286 0.45714286
 0.45714286 0.45714286 0.45714286]
```

In [36]:
```
en.score(x_test,y_test)
```

Out[36]:
```
-0.023323615160349753
```

In [38]:
```
from sklearn import metrics
```

In [39]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 0.5182154600086817
```

In [40]:
```
print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error 0.27573774971983417
```

In [43]:
```
print(" Root Mean Squared Error",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
 Root Mean Squared Error 0.525107369706267
```