D9

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\13_placement.csv")
df
```

Out[2]:

	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

1000 rows × 3 columns

In [3]:
```
df.head(10)
```

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
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

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 3 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   cgpa                  1000 non-null   float64
 1   placement_exam_marks  1000 non-null   float64
 2   placed                1000 non-null   int64  
dtypes: float64(2), int64(1)
memory usage: 23.6 KB
```

In [5]:
```
df.describe()
```

Out[5]:

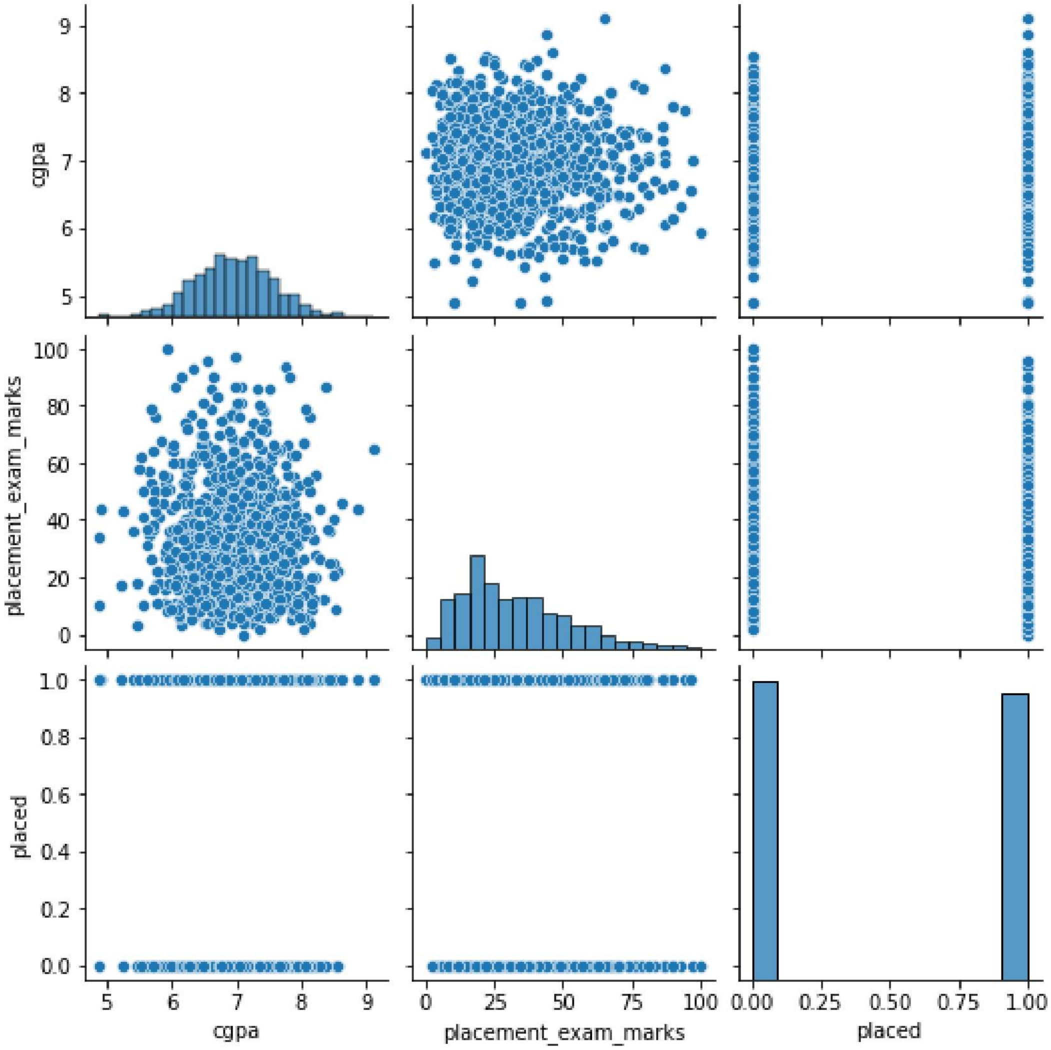
	cgpa	placement_exam_marks	placed
count	1000.000000	1000.000000	1000.000000
mean	6.961240	32.225000	0.489000
std	0.615898	19.130822	0.500129
min	4.890000	0.000000	0.000000
25%	6.550000	17.000000	0.000000
50%	6.960000	28.000000	0.000000
75%	7.370000	44.000000	1.000000
max	9.120000	100.000000	1.000000

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

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

In [7]:
```
sns.pairplot(df)
```

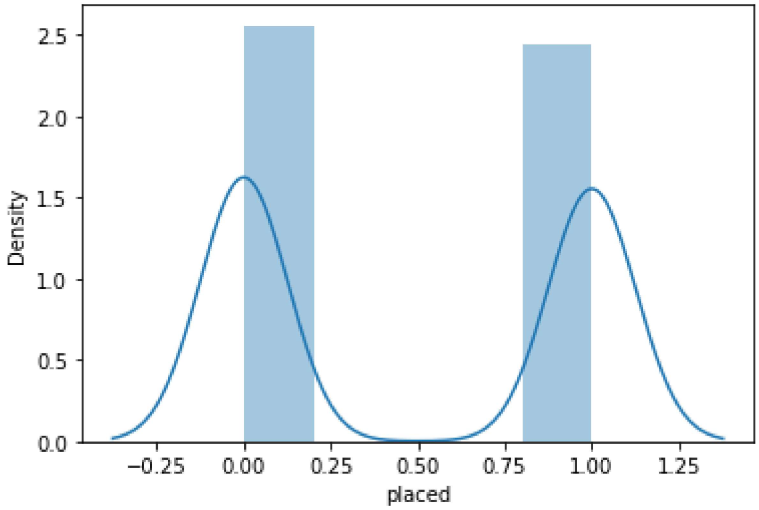
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x22fb1b719a0>
```



In [8]:
```
sns.distplot(df['placed'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

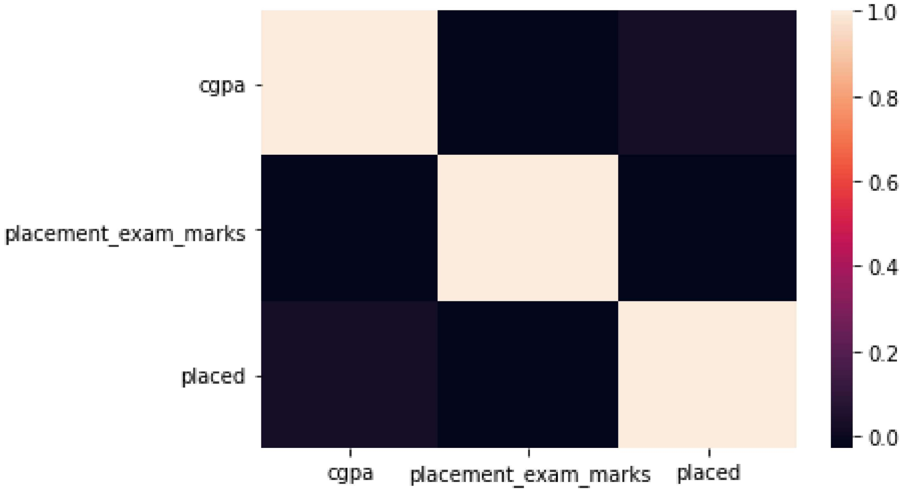
Out[8]:
```
<AxesSubplot:xlabel='placed', ylabel='Density'>
```



In [9]:
```
df1=df[['cgpa', 'placement_exam_marks', 'placed']]
```

In [10]:
```
sns.heatmap(df1.corr())
```

Out[10]:
```
<AxesSubplot:>
```



In [11]:
```
x=df1[['placement_exam_marks', 'placed']]
y=df1['cgpa']
```

In [12]:
```
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 [13]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[13]:
```
LinearRegression()
```

In [14]:
```
print(lr.intercept_)
```

```
6.932030195129635
```

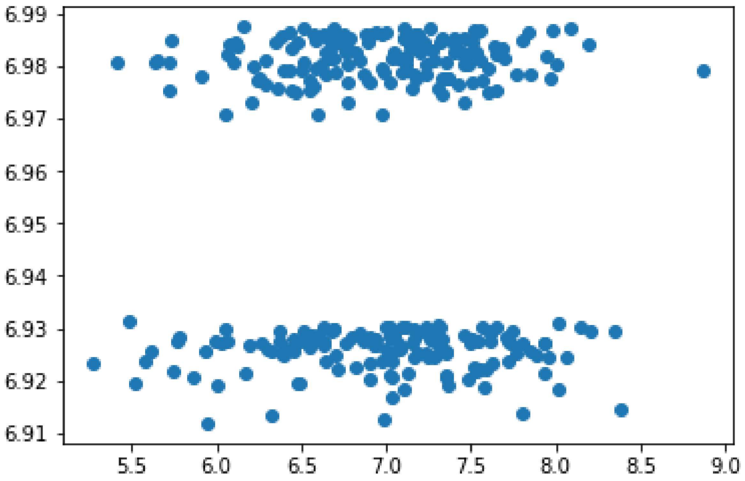
In [15]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[15]:

	Co-efficient
placement_exam_marks	-0.000202
placed	0.056144

In [16]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[16]:
```
<matplotlib.collections.PathCollection at 0x22fb3c5b430>
```



In [17]:
```
print(lr.score(x_test,y_test))
```

```
-0.005019613850439919
```

In [18]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [19]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[19]:
```
Ridge(alpha=10)
```

In [20]:
```
rr.score(x_test,y_test)
```

Out[20]:
```
-0.004696700062281289
```

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

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

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

Out[22]:
```
-0.0023332592216946946
```

In [23]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[23]:
```
ElasticNet()
```

In [24]:
```
print(en.coef_)
```

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

In [25]:
```
print(en.intercept_)
```

```
6.952428571428572
```

In [26]:
```
print(en.predict(x_train))
```

```
[6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857 6.95242857
```

In [27]:
```
print(en.score(x_train,y_train))
```

```
0.0
```

In [28]:
```
from sklearn import metrics
```

In [29]:
```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 0.49109082087398404
```

In [30]:
```
print("Mean Square Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Square Error: 0.37158800566998146
```

In [31]:
```
print("Root Mean Square Error:",np.sqrt(metrics.mean_absolute_error(y_test,pre
```

```
Root Mean Square Error: 0.7007787246156836
```

In [ ]:
```

```