Deena 20104016

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as pp
```

Problem Statement

LINEAR REGRESSION

In [2]:
```
a = pd.read_csv("13_placement.csv")
```

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

HEAD

In [3]:

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

Data Cleaning and Preprocessing

In [4]:

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

In [5]:

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

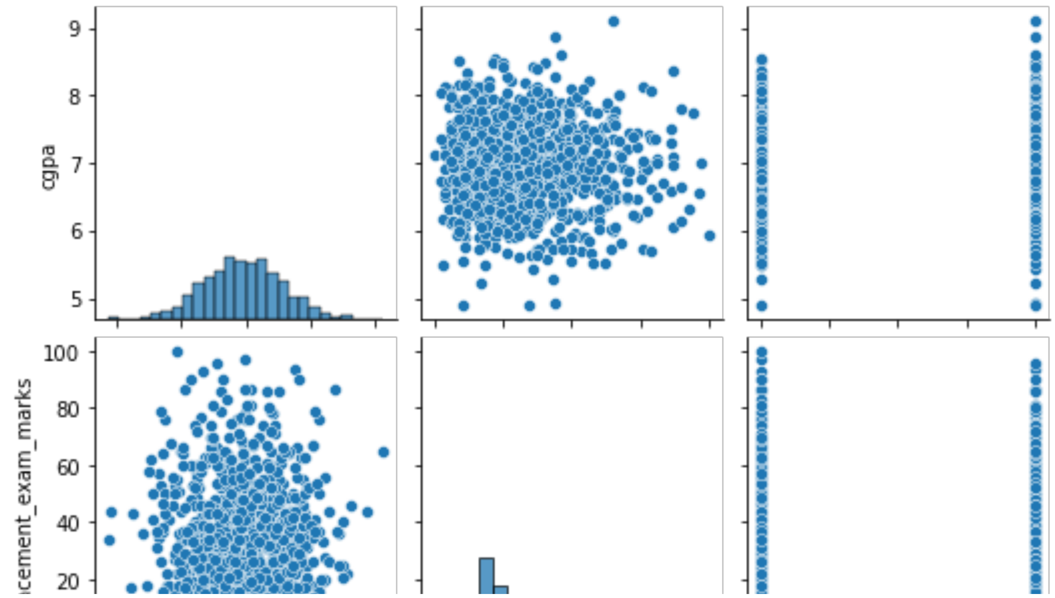
To display heading

In [6]:

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

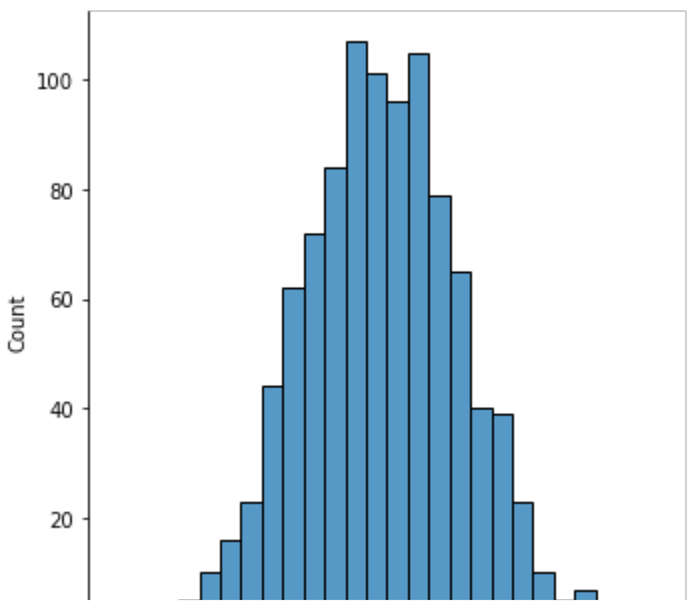
In [7]:

Out[7]: <seaborn.axisgrid.PairGrid at 0x1e11e7a5b80>



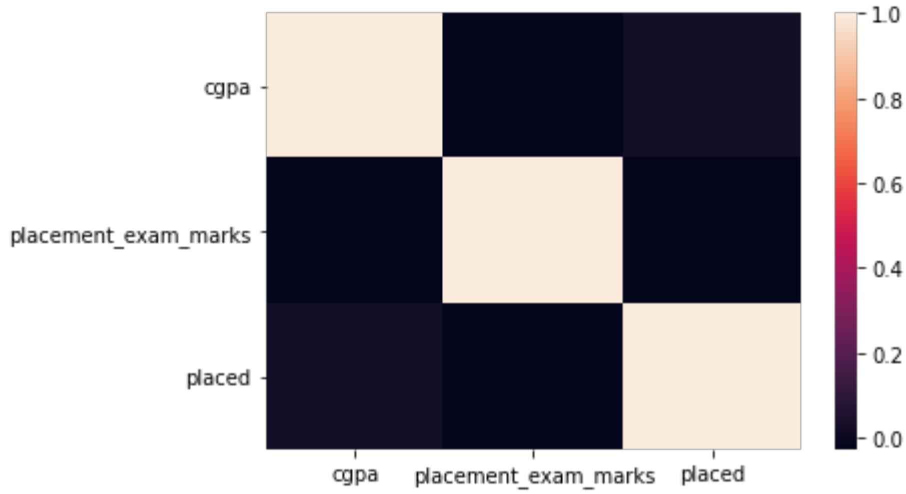
In [8]:

Out[8]: <seaborn.axisgrid.FacetGrid at 0x1e120451940>



In [9]:

Out[9]: <AxesSubplot:>



TO TRAIN THE MODEL - MODEL BUILDING

In [10]:

```python
x = a[['cgpa']]
```

In [11]:

```python
# to split my dataset into training and test data
from sklearn.model_selection import train_test_split
```

In [12]:

```python
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

Out[12]: LinearRegression()

In [13]:

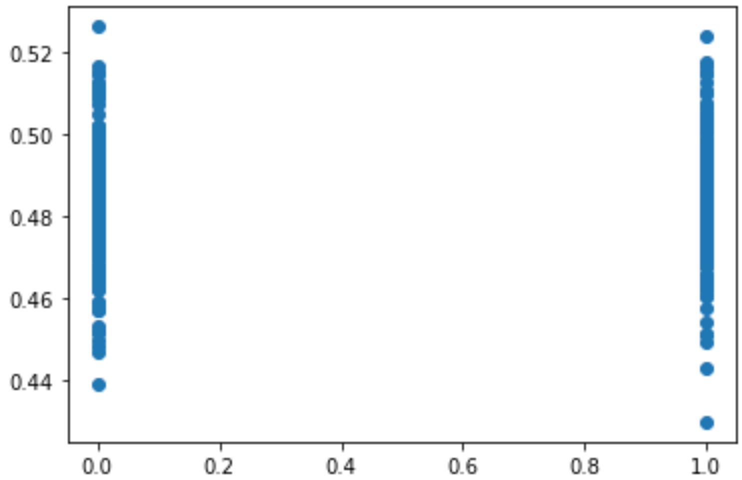
```python
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[13]:

	Co-efficient
cgpa	0.026521

In [14]:
```
prediction= lr.predict(x_test)
```

Out[14]: <matplotlib.collections.PathCollection at 0x1e120f18ee0>



In [15]:

Out[15]: -0.0011820259592059124

RIDGE & LASSO

In [16]:
```
from sklearn.linear_model import Ridge,Lasso
rr=Ridge(alpha=10)
```

Out[16]: Ridge(alpha=10)

In [17]:

Out[17]: -0.0011399185308611504

In [18]:
```
la=Lasso(alpha=10)
```

Out[18]: Lasso(alpha=10)

In [19]:

Out[19]: -0.0009877551020409658

In [20]:
```
from sklearn.linear_model import ElasticNet
a=ElasticNet()
```

Out[20]: ElasticNet()

In [21]:
```
print(a.coef_)
print(a.intercept_)
print(a.score(x_test,y_test))
```

```
[0.]
0.48428571428571426
-0.0009877551020409658
[0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571
```

```
0.48428571 0.48428571 0.48428571 0.48428571 0.48428571 0.48428571]
```

In [22]:
```
from sklearn import metrics
print(" Mean Absolute Error :",metrics.mean_absolute_error(y_test,prediction))
print(" Mean Squared Error :",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Absolute Error : 0.499769263706374
Mean Squared Error : 0.2502955064898015
Root Mean Absolute Error : 0.7069436071614015
```