agn7xtqka

July 31, 2023

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

[ ]:
```
df=pd.read_csv("/content/13_placement.csv")
df
```

[ ]:

	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 x 3 columns]

[ ]:
```
df.head()
```

[ ]:

	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

1 DATA CLEANING AND DATA PREPROCESSING

[ ]:
```
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
```

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

[ ]:
```
              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
```

[ ]:
```
df.columns
```

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

[ ]:
```
df1=df.dropna(axis=1)
df1
```

[ ]:
```
      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 x 3 columns]
```

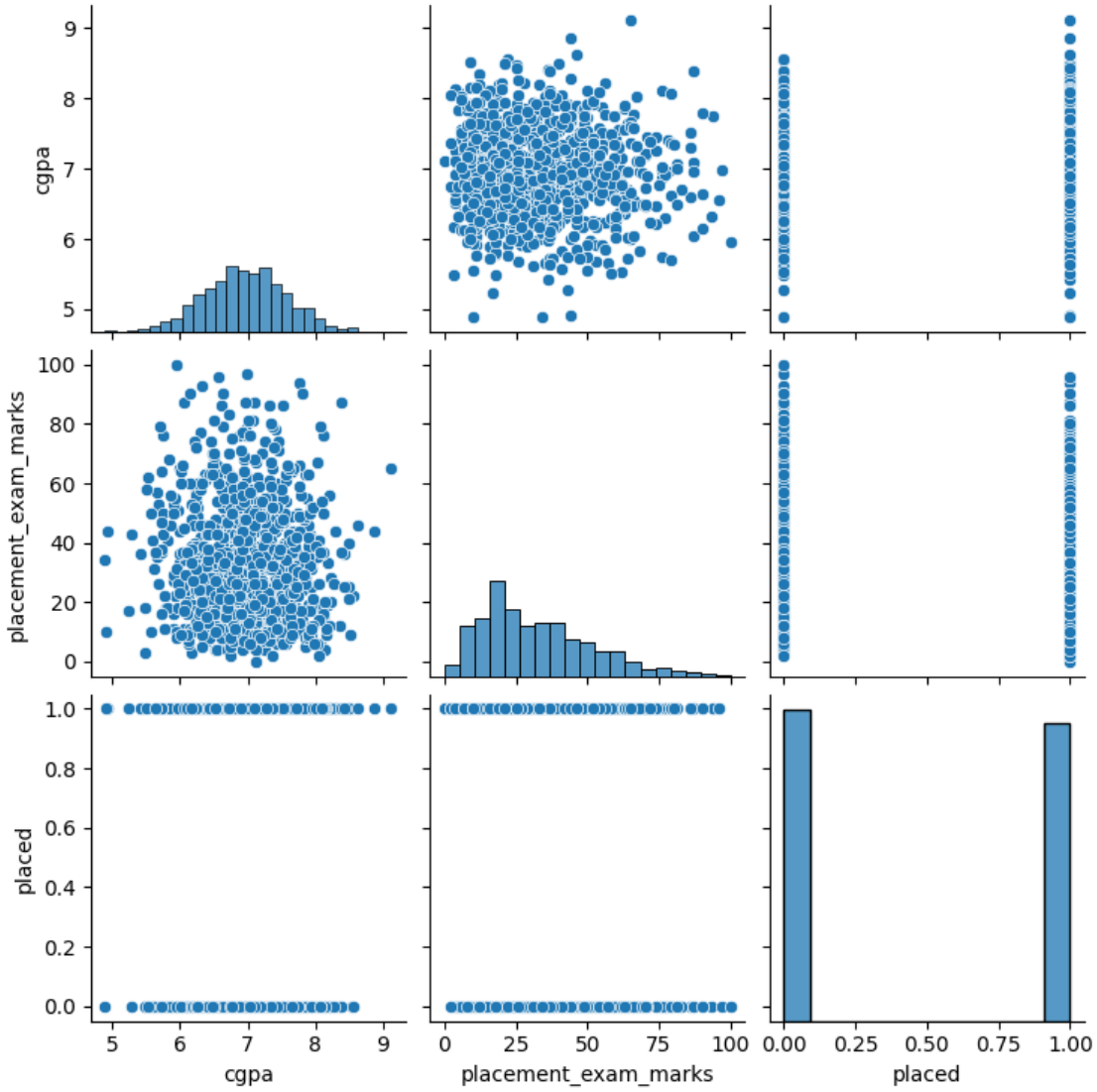
[ ]:
```
df1.columns
```

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

2 EDA AND VISUALIZATION

[ ]:
```
sns.pairplot(df1)
```

[ ]: <seaborn.axisgrid.PairGrid at 0x7d3cb5b85fc0>

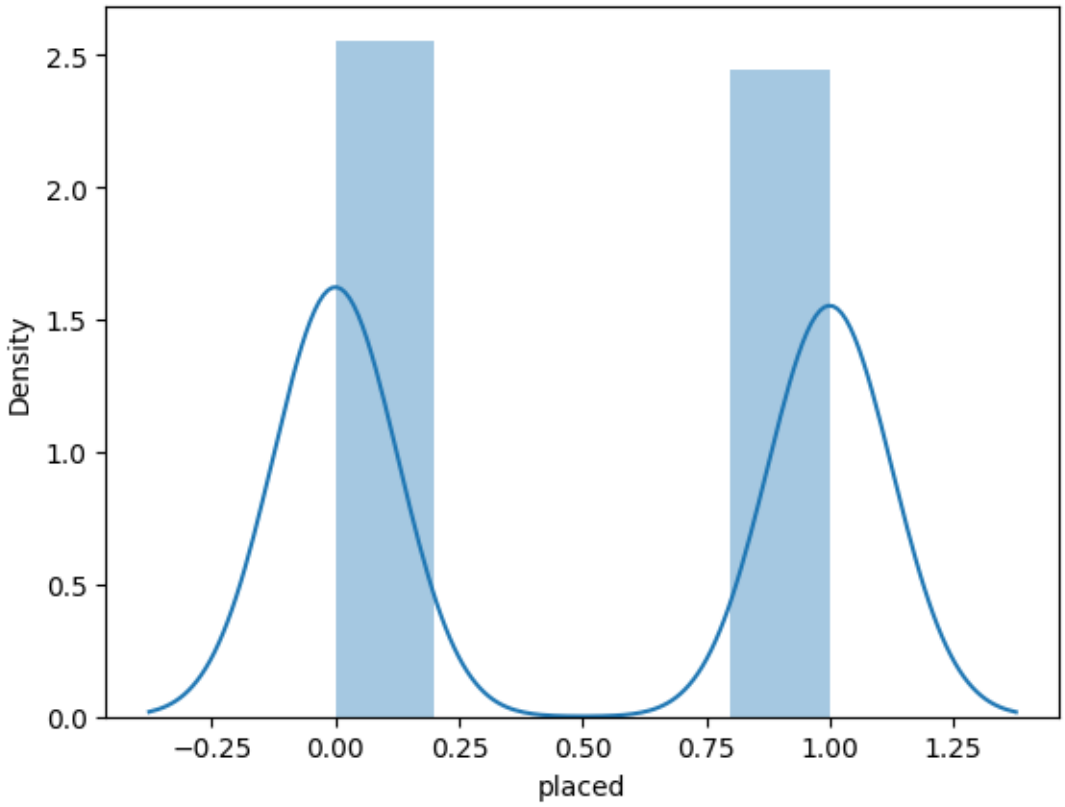


[ ]:
```
sns.distplot(df1['placed'])
```

```
<ipython-input-10-dc9f78aae914>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
```

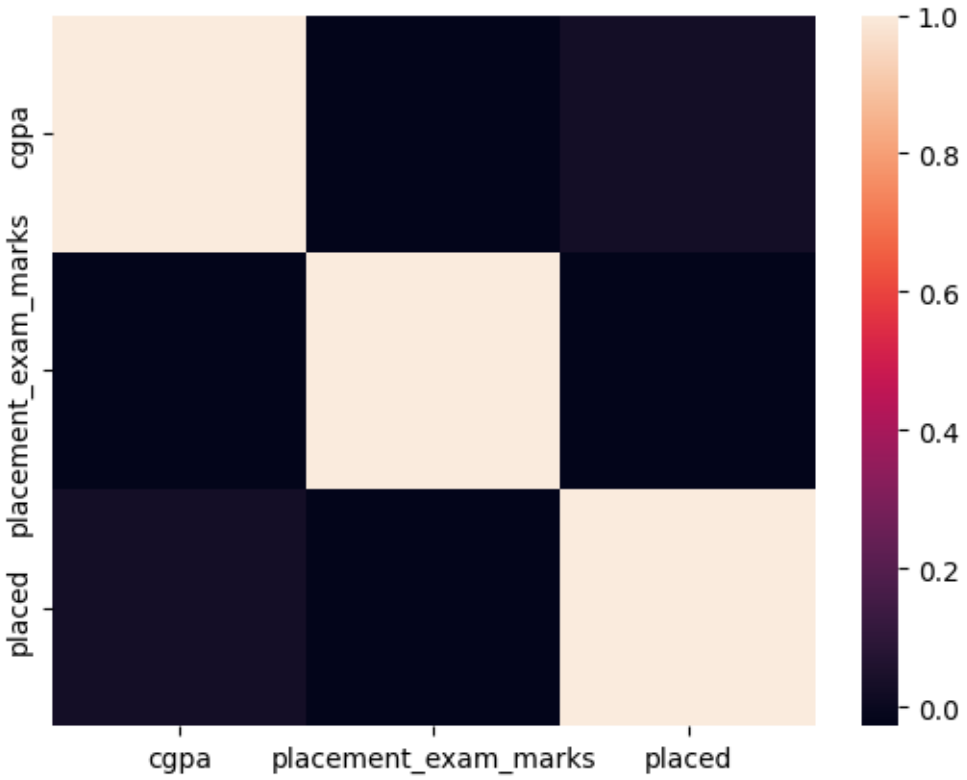
```
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df1['placed'])
```

[ ]:
```
<Axes: xlabel='placed', ylabel='Density'>
```



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

[ ]:
```
<Axes: >
```



3 TO TRAIN THE MODEL AND MODEL BULDING

[ ]:
```
x=df[['cgpa', 'placement_exam_marks']]
y=df['placed']
```

[ ]:
```
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)
```

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

[ ]:
```
LinearRegression()
```

[ ]:
```
lr.intercept_
```

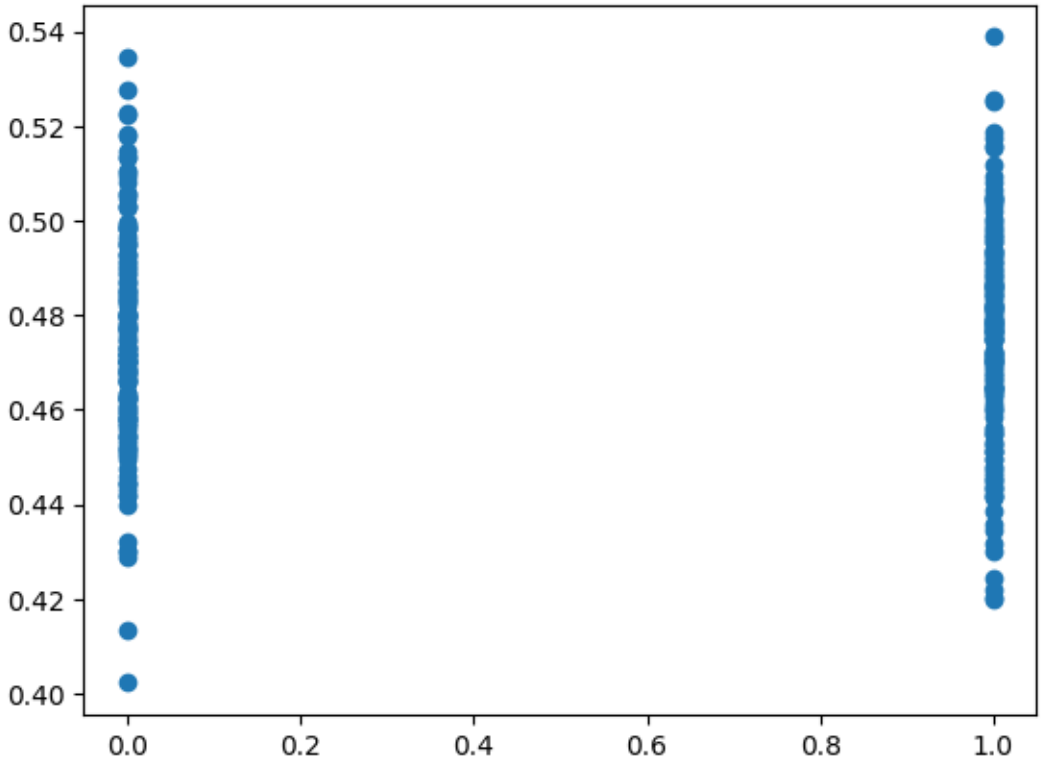
[ ]:
```
0.24409262552245464
```

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

[ ]:
```
                      Co-efficient
cgpa                      0.035047
placement_exam_marks     -0.000390
```

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

[ ]:
```
<matplotlib.collections.PathCollection at 0x7d3cae7bc2e0>
```



4 ACCURACY

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

[ ]:
```
-0.009647703157610321
```

[ ]:
```
lr.score(x_train,y_train)
```

[ ]:
```
0.0021530144080373903
```

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

```
rr.fit(x_train,y_train)
```

[ ]: Ridge(alpha=10)

[ ]:
```
rr.score(x_train,y_train)
```

[ ]: 0.0021506207437649305

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

[ ]: -0.009479740435849315

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

[ ]: Lasso(alpha=10)

[ ]:
```
la.score(x_train,y_train)
```

[ ]: 0.0

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

[ ]: -0.007857469911041193

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

[ ]: ElasticNet()

[ ]:
```
print(en.coef_)
print(en.intercept_)
```

```
[ 0. -0.]
0.4757142857142857
```

[ ]:
```
prediction = en.predict(x_test)
prediction
```

[ ]:
```
array([0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
       0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
       0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
       0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
       0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
       0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
```

0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,

```
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429,
0.47571429, 0.47571429, 0.47571429, 0.47571429, 0.47571429])
```

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

[ ]:
```
-0.007857469911041193
```

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

```
Mean Absolute Error:  0.5009714285714285
Mean Squared Error:  0.2515612244897959
Root Mean Squared Error:  0.5015587946490381
```