In [1]:

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

In [2]:

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

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.shape
```

Out[4]:

```
(500, 9)
```

Here we can see that the data having 500 data points and 9 features

In [5]:

```
df.columns
```

Out[5]:

```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

In [6]:

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

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   GRE Score          500 non-null    int64  
 1   TOEFL Score        500 non-null    int64  
 2   University Rating  500 non-null    int64  
 3   SOP                500 non-null    float64
 4   LOR                500 non-null    float64
 5   CGPA               500 non-null    float64
 6   Research           500 non-null    int64  
 7   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 31.4 KB
```

In [8]:

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

Out[8]:

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

Here can can see that their is no missing value

In [9]:

```
df.describe()
```

Out[9]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	C of
count	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.
mean	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.
std	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0
min	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.
25%	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.
50%	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.
75%	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.
max	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.

Here we can see that the mean value of GRE score is 316.47 and median value is 317

the mean value of TOEFL score is 107.19 an dthe median is 107

the mena value of Research is 0.56 and the median value is 1

the mean value of CGPA is 8.57 and the median value is 8.56

from all the above values we can conclude that the data is not effected by outlyers

In [10]:

```
df['GRE Score'].value_counts()
```

Out[10]:

```
312    24
324    23
316    18
321    17
322    17
327    17
314    16
311    16
320    16
317    15
325    15
315    13
308    13
323    13
318    12
319    12
326    12
304    12
300    12
313    12
310    11
305    11
301    11
329    10
307    10
299    10
298    10
309     9
340     9
331     9
328     9
330     8
332     8
334     8
306     7
302     7
297     6
296     5
303     5
336     5
295     5
335     4
333     4
338     4
339     3
337     2
294     2
290     2
293     1
Name: GRE Score, dtype: int64
```

Here we can see that the GRE score is an contines value not catagorical value

In [11]:

```
df['TOEFL Score'].value_counts()
```

Out[11]:

```
110    44
105    37
104    29
112    28
107    28
106    28
103    25
102    24
100    24
99     23
111    20
101    20
113    19
109    19
108    19
114    18
116    16
115    11
119    10
98     10
118    10
120     9
117     8
97      7
96      6
95      3
94      2
93      2
92      1
Name: TOEFL Score, dtype: int64
```

Here the tofel score is also an continues value not categorical value

In [12]:

```
df['University Rating'].value_counts()
```

Out[12]:

```
3    162
2    126
4    105
5     73
1     34
Name: University Rating, dtype: int64
```

University rating is an categorical value and the highest number is for the rating 3 followed by 2

In [13]:

```
df['SOP'].value_counts()
```

Out[13]:

```
4.0    89
3.5    88
3.0    80
2.5    64
4.5    63
2.0    43
5.0    42
1.5    25
1.0     6
Name: SOP, dtype: int64
```

the highest SOP score is 5.0 obtained by 42 and the score 4.0 is obtained by 89 followed by 3.5 by 88 students

In [14]:

```
df['LOR '].value_counts()
```

Out[14]:

```
3.0    99
4.0    94
3.5    86
4.5    63
5.0    50
2.5    50
2.0    46
1.5    11
1.0     1
Name: LOR , dtype: int64
```

A large number of students are getting the LOR value as 3.0 followed by 4 and followed by 3.5

In [15]:

```
df['CGPA'].value_counts()
```

Out[15]:

```
8.00    9
8.76    9
8.54    7
8.45    7
8.56    7
       ..
8.72    1
7.23    1
7.87    1
9.67    1
7.57    1
Name: CGPA, Length: 184, dtype: int64
```

In [16]:

```
df['Research'].value_counts()
```

Out[16]:

```
1    280
0    220
Name: Research, dtype: int64
```

Research is almoist balenced

In [17]:

```
df['Chance of Admit '].value_counts()
```

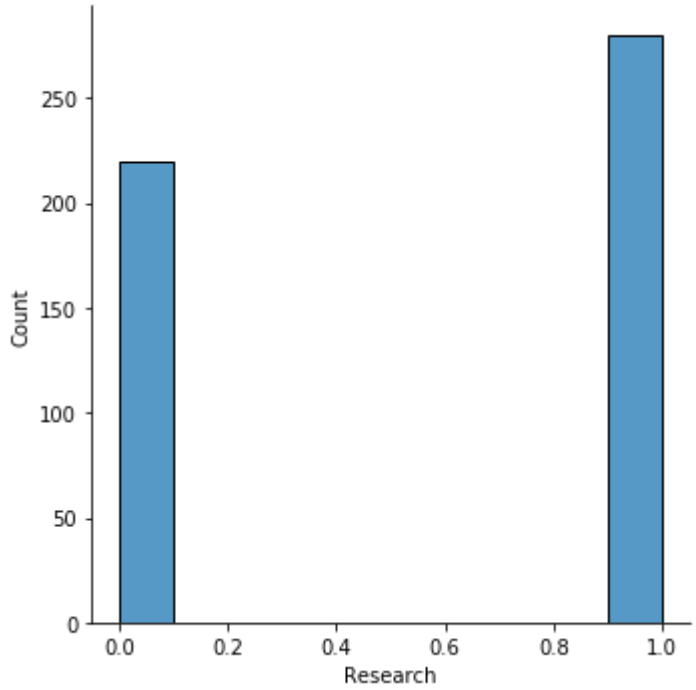
Out[17]:

```
0.71    23
0.64    19
0.73    18
0.72    16
0.79    16
        ..
0.34     2
0.50     2
0.43     1
0.37     1
0.39     1
Name: Chance of Admit , Length: 61, dtype: int64
```

The highest number of people are getting the chance of admit as 0.71 followed by 0.64

In [18]:

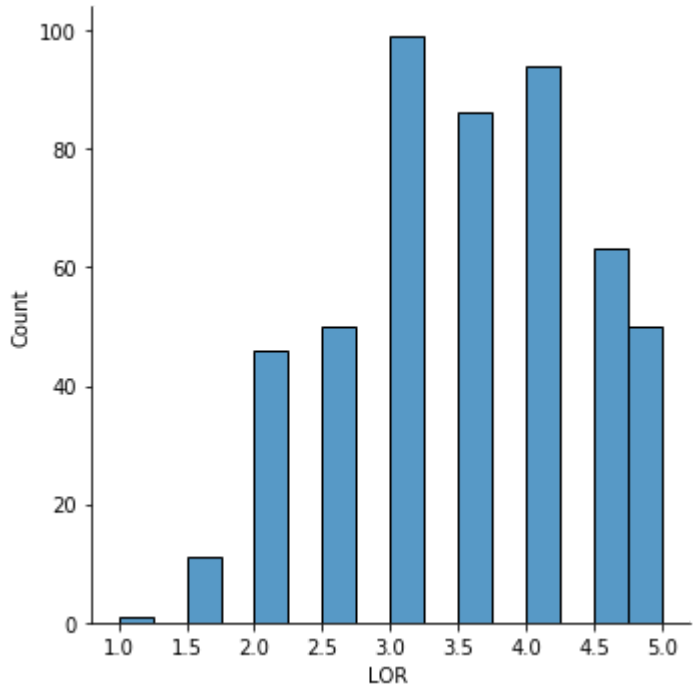
```
sns.displot(df['Research'])
plt.show()
```



research recomendation is almost balenced

In [19]:

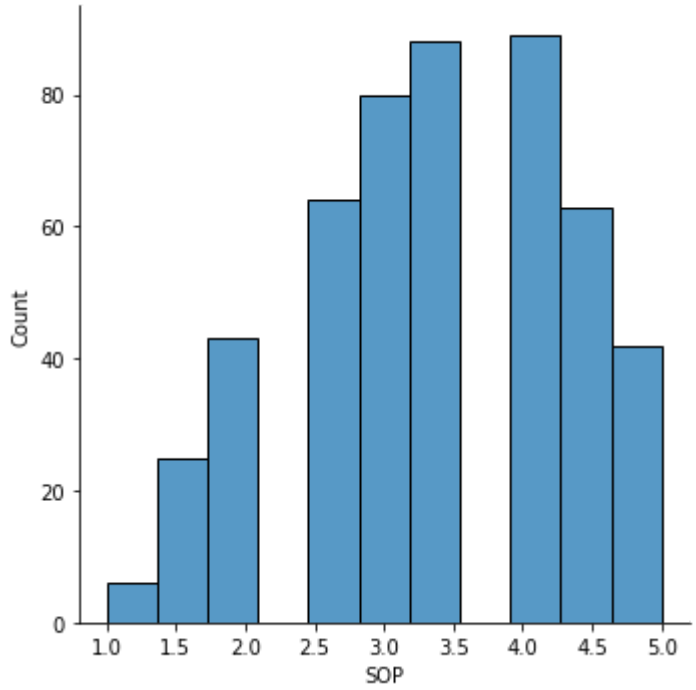
```
sns.displot(df['LOR '])
plt.show()
```



large number of LOR is concentrated between [3.0,4.0]

In [20]:

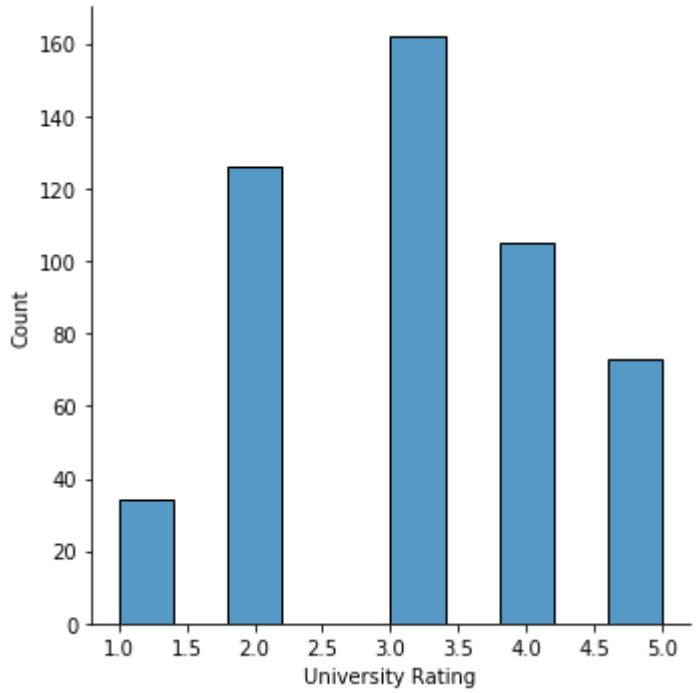
```
sns.displot(df['SOP'])
plt.show()
```



the large number of SOP are concentrated around the value [2.5,4.5]

In [21]:

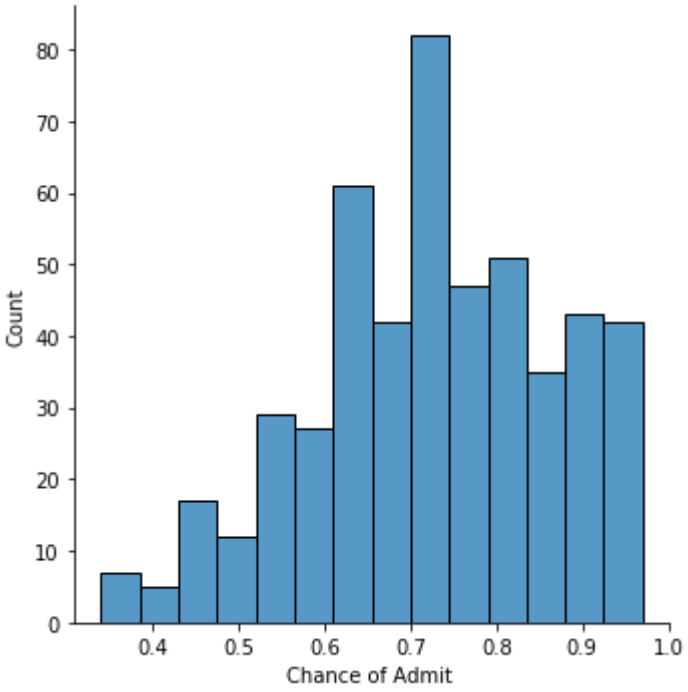
```
sns.displot(df['University Rating'])
plt.show()
```

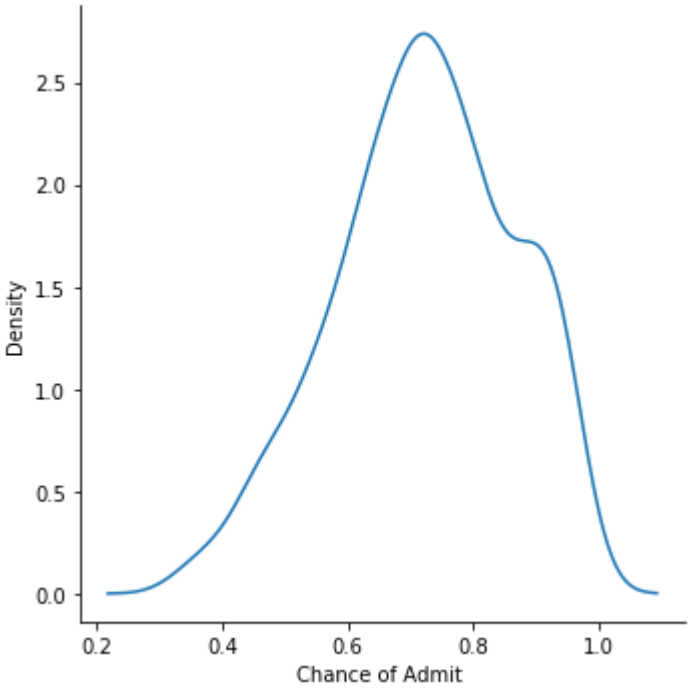


large number of university are having teh rating as 3 followed by 2

In [22]:

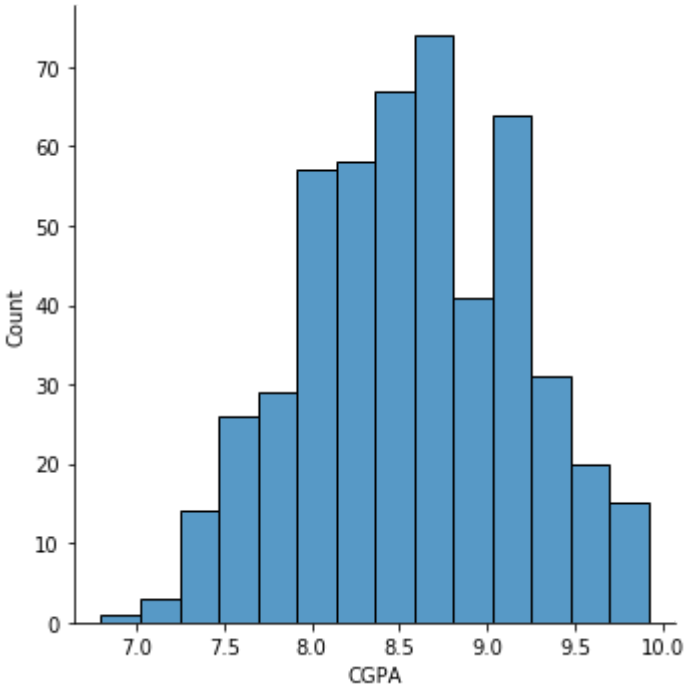
```
sns.displot(df['Chance of Admit '])
sns.displot(df['Chance of Admit '],kind='kde')
plt.show()
```

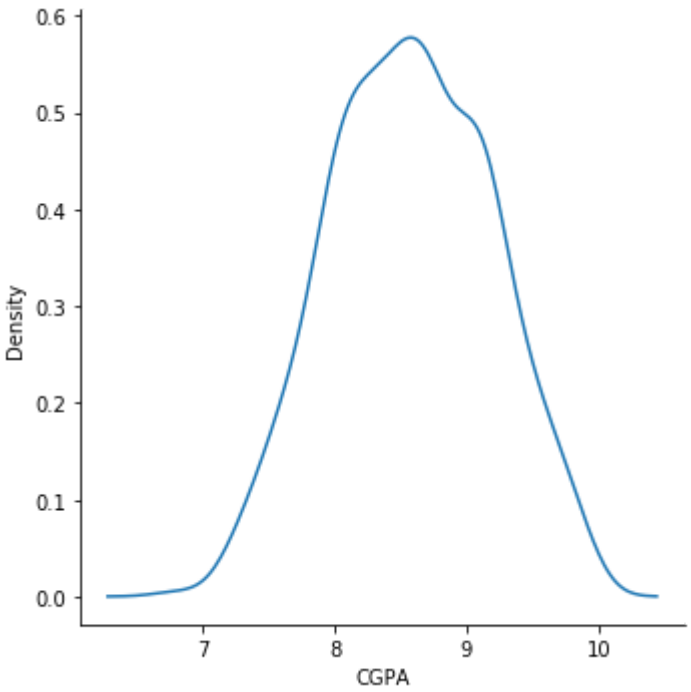




In [23]:

```
sns.displot(df['CGPA'])
sns.displot(df['CGPA'],kind='kde')
plt.show()
```





In [24]:

```
df.head()
```

Out[24]:

	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
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

In [25]:

```
X = df.drop(columns='Chance of Admit ')
Y = df['Chance of Admit ']
```

dividding the data into features and labels

In [26]:

```
from sklearn.model_selection import train_test_split
```

In [27]:

```
from sklearn.preprocessing import StandardScaler
```

In [28]:

```
scaler = StandardScaler()
```

In [29]:

```
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.20)
```

splitting the data into train test with test data as 20% of total data

In [30]:

```
import statsmodels.api as sm
```

In [31]:

```
col = x_train.columns
```

In [32]:

```
x_train = scaler.fit_transform(x_train)
```

standardizing the data using mean centering and varience scalimng technique

In [33]:

```
x_train = pd.DataFrame(x_train,columns=col)
```

In [34]:

```
x_train
```

Out[34]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	-0.145569	-0.376393	-0.092305	-1.368010	-1.066208	-0.391328	-1.122447
1	-0.413077	-0.376393	-0.971403	-1.871881	-0.521530	-0.191819	-1.122447
2	-0.234738	-0.043302	-0.971403	-0.864139	0.567827	-0.507708	-1.122447
3	1.102803	0.622880	0.786792	1.151345	1.112506	1.005233	0.890911
4	0.121939	-1.042575	1.665890	0.143603	1.657185	0.340204	0.890911
...	...	...	...	...	...	...	...
395	0.121939	-0.209847	-0.092305	-1.368010	-0.521530	0.124069	-1.122447
396	-1.393941	-0.043302	-0.092305	0.143603	0.023149	-0.391328	0.890911
397	0.300278	-0.709484	-0.092305	-0.360268	-0.521530	-1.455374	-1.122447
398	-0.502247	-0.376393	-0.092305	0.143603	-0.521530	-0.208445	0.890911
399	0.032770	-0.209847	-0.092305	0.647474	0.023149	-0.125316	0.890911

400 rows × 7 columns

In [35]:

```
x_train = sm.add_constant(x_train)
```

In [36]:

```
model = sm.OLS(y_train.values,x_train)
```

training the linear regression model with training data

In [37]:

```
res = model.fit()
```

In [38]:

```
print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
====
Dep. Variable:                      y   R-squared:
0.817
Model:                            OLS   Adj. R-squared:
0.813
Method:                 Least Squares   F-statistic:                      2
49.5
Date:                Tue, 15 Nov 2022   Prob (F-statistic):           3.85e
-140
Time:                        23:03:06   Log-Likelihood:                  56
2.92
No. Observations:                 400   AIC:                             -1
110.
Df Residuals:                     392   BIC:                             -1
078.
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
===========
                     coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-----------
const                0.7232      0.003    241.714      0.000       0.717
0.729
GRE Score            0.0197      0.006      3.118      0.002       0.007
0.032
TOEFL Score          0.0144      0.006      2.451      0.015       0.003
0.026
University Rating    0.0076      0.005      1.601      0.110      -0.002
0.017
SOP                  0.0069      0.005      1.390      0.165      -0.003
0.017
LOR                  0.0154      0.004      3.685      0.000       0.007
0.024
CGPA                 0.0688      0.006     10.602      0.000       0.056
0.082
Research             0.0112      0.004      3.111      0.002       0.004
0.018
==============================================================================
====
Omnibus:                       98.246   Durbin-Watson:
1.891
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               24
2.262
Skew:                          -1.206   Prob(JB):                     2.47
e-53
Kurtosis:                       5.952   Cond. No.
5.63
==============================================================================
====

Warnings:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
```

Here we can see that the R-squared value is 0.817 and adjusted r-squared value is 0.813 so teh train error is less

In [39]:

```
print(res.params)
```

```
const               0.723175
GRE Score           0.019657
TOEFL Score         0.014370
University Rating   0.007564
SOP                 0.006927
LOR                 0.015384
CGPA                0.068792
Research            0.011211
dtype: float64
```

In [40]:

```
print(res.rsquared)
```

```
0.8166889434759697
```

In [41]:

```
print(res.rsquared_adj)
```

```
0.8134155317523263
```

In [42]:

```
res_lassso = model.fit_regularized()
```

training the data with lasso regulizer

In [43]:

```
print(res_lassso.params)
```

```
const               0.723175
GRE Score           0.019657
TOEFL Score         0.014370
University Rating   0.007564
SOP                 0.006927
LOR                 0.015384
CGPA                0.068792
Research            0.011211
dtype: float64
```

In [46]:

```
y_pred = res.predict(sm.add_constant(x_test))
```

predicting the unseen data

In [47]:

```
residual = y_test.values-y_pred
```

getting the residuals of the testdata and the predicted data

In [48]:

```
np.mean(residual)
```

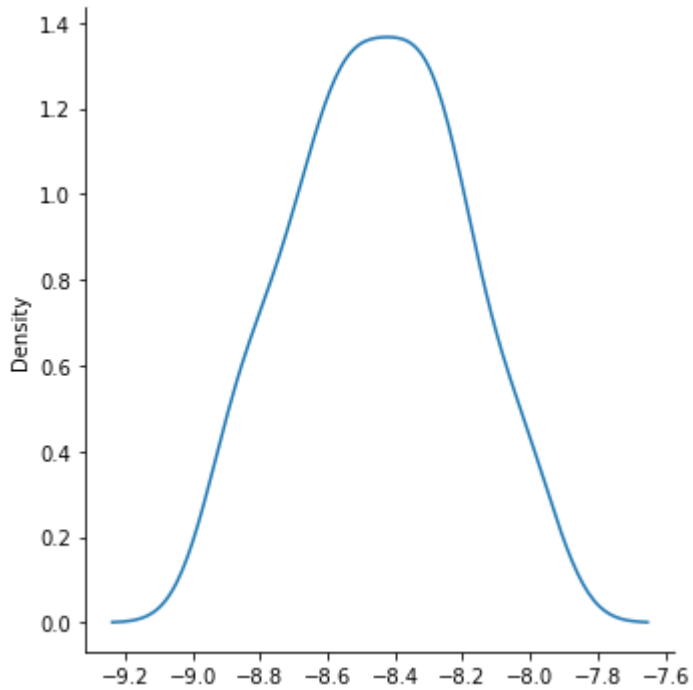
Out[48]:

```
-8.450281063224677
```

the meaan of the residuals is around -8.45

In [69]:

```
sns.displot(residual,kind='kde')
plt.show()
```



the distribution of the residuals is almost bell shaped curve but the mean is not centered at 0

In [50]:

```
from sklearn.metrics import median_absolute_error,mean_squared_error,r2_score
```

In [51]:

```
median_absolute_error(y_test,y_pred)
```

Out[51]:

```
8.447343787546373
```

median absolute error of the model is 8.447

In [52]:

```
mean_squared_error(y_test,y_pred)
```

Out[52]:

```
71.46708365232038
```

mean sqquared error is 71.46

In [53]:

```
r2_score(y_test,y_pred)
```

Out[53]:

```
-3134.071225316739
```

the r squared value is -3134 whis is too bad for a model as it is performing very bad than an mean model

In [55]:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

def calc_vif(X):
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

    return(vif)
```

function to calculate the variance inflation factor of the data

In [56]:

```
calc_vif(X)
```

Out[56]:

	variables	VIF
0	GRE Score	1308.061089
1	TOEFL Score	1215.951898
2	University Rating	20.933361
3	SOP	35.265006
4	LOR	30.911476
5	CGPA	950.817985
6	Research	2.869493

as we can see that the VIF of GRE score ,toefl score, CGPA are too large it means that we can predict the values of this featuires using the remaining features

In [57]:

```
calc_vif(X.drop(columns='GRE Score'))
```

Out[57]:

	variables	VIF
0	TOEFL Score	639.741892
1	University Rating	19.884298
2	SOP	33.733613
3	LOR	30.631503
4	CGPA	728.778312
5	Research	2.863301

dropping the GRE score and observing the VIF score

In [58]:

```
calc_vif(X.drop(columns=['GRE Score','TOEFL Score']))
```

Out[58]:

	variables	VIF
0	University Rating	19.777410
1	SOP	33.625178
2	LOR	30.356252
3	CGPA	25.101796
4	Research	2.842227

In [59]:

```
calc_vif(X.drop(columns=['GRE Score','TOEFL Score','SOP']))
```

Out[59]:

	variables	VIF
0	University Rating	15.140770
1	LOR	26.918495
2	CGPA	22.369655
3	Research	2.819171

In [61]:

```
calc_vif(X.drop(columns=['GRE Score','TOEFL Score','SOP','LOR ']))
```

Out[61]:

	variables	VIF
0	University Rating	12.498400
1	CGPA	11.040746
2	Research	2.783179

In [65]:

```
calc_vif(X.drop(columns=['GRE Score','TOEFL Score','SOP','LOR ','University Rating']))
```

Out[65]:

	variables	VIF
0	CGPA	2.455008
1	Research	2.455008

finally we can see after droping GRE Score TOEFL Score SOP LOR University Rating we can observe the data is having less VIF score

In [ ]:

```

```

Actionable Insights & Recommendations

As their is more gap between the tarin and the test score (which specifies that teh model is opverfitting to the traain data) we have to do hyperparameter tuning(on lambda) with regulizer as for small value of lambda tends to overfit the data and large values of lambda underfits the data.

we have to do crossvalidation on the data so that the effect of the outlyers is minimised

While doing hyperparameter tuning we have to be causies that the model should be low bias and low varience model because both are important for the model to be best model.

In [ ]:

```

```