Problem Statement

The data given should be taken into consideration based on indian perspective.

1.) Jamboree want's predict one's chances of admission given the rest of the variables.

2.) Jamboree want's to find what factors are important in graduate admissions and how these factors are interrelated among themselves.

In [65]:

```
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
```

In [66]:

```
jamboree=pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/00
1/839/original/Jamboree_Admission.csv")
jamboree.head()
```

Out[66]:

	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 [67]:

```
jamboree.shape
```

Out[67]:

```
(500, 9)
```

In [68]:

```
jamboree.dtypes
```

Out[68]:

```
Serial No.             int64
GRE Score              int64
TOEFL Score            int64
University Rating      int64
SOP                  float64
LOR                  float64
CGPA                 float64
Research               int64
Chance of Admit      float64
dtype: object
```

In [ ]:

```

```

Analysing basic metrics

1.)The target variable is chance of admit.

2.)Serial No is dropped since it describes nothing but a unique row number.

3.)SOP,LOR contains discrete values between 0-5([4.5, 4. , 3. , 3.5, 2. , 5. , 1.5, 1. , 2.5])

4.)University rating contains discrete values bwtween 1-5[1,2,3,4,5] and Research contains discrete values 1 and 2.

5.)There is not much difference b/w max and mean values of continuous variables hence there are less number of outliers but this will be checked with scatter and box plots.

In [69]:

```
jamboree.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         500 non-null    int64  
 1   GRE Score          500 non-null    int64  
 2   TOEFL Score        500 non-null    int64  
 3   University Rating  500 non-null    int64  
 4   SOP                500 non-null    float64
 5   LOR                500 non-null    float64
 6   CGPA               500 non-null    float64
 7   Research           500 non-null    int64  
 8   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 35.3 KB
```

In [70]:

```
jamboree.drop(["Serial No."],axis=1,inplace=True)
jamboree.head()
```

Out[70]:

	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 [71]:

```
jamboree.describe(include=[np.number])
```

Out[71]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	
count	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	5
mean	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	
std	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	
min	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	
25%	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	
50%	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	
75%	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	
max	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	

In [72]:

```
jamboree["University Rating"].unique()
```

Out[72]:

```
array([4, 3, 2, 5, 1], dtype=int64)
```

In [73]:

```
jamboree["Research"].unique()
```

Out[73]:

```
array([1, 0], dtype=int64)
```

In [74]:

```
jamboree["SOP"].unique()
```

Out[74]:

```
array([4.5, 4. , 3. , 3.5, 2. , 5. , 1.5, 1. , 2.5])
```

In [75]:

```
jamboree["LOR "].unique()
```

Out[75]:

```
array([4.5, 3.5, 2.5, 3. , 4. , 1.5, 2. , 5. , 1. ])
```

Missing value

No missing values are present in the data.

In [76]:

```
jamboree.isna().sum()
```

Out[76]:

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

Univariate Analysis

Continuos variables

1.)The distributions for variables-> GRE Score,TOEFL Score,CGPA are symmetric and are almost completely gaussian as well.

2.)There are no outliers for these continuous variables except for chance of admit variable.chance of admit variable contains some outliers

3.)From the distribution of GRE score we can understand that majority of people have GRE scores b/w 310-325.

4.)From the distribution of TOEFL Score,we can understand that majority of people have TOEFL scores b/w 103-113.

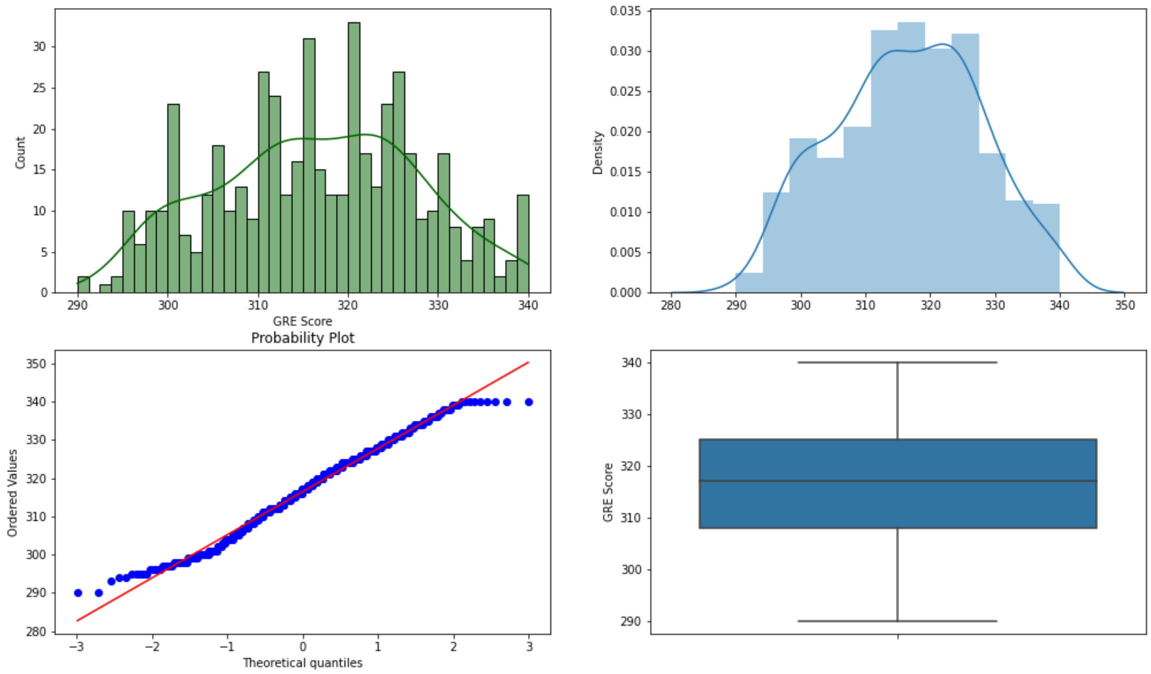
5.)From the distribution of CGPA,we can understand that majority of people have CGPA b/w 8.2 and 9.0.

In [77]:

```
plt.rcParams["figure.figsize"]=(17,10)
plt.subplot(221)
sn.histplot(data=jamboree,x="GRE Score",bins=40,kde=True,color="darkgreen")
plt.subplot(222)
sn.distplot(x=jamboree["GRE Score"])
prob=stats.probplot(jamboree["GRE Score"],dist=stats.norm,plot=plt.subplot(223))
plt.subplot(224)
sn.boxplot(data=jamboree,y="GRE Score")
```

Out[77]:

```
<AxesSubplot:ylabel='GRE Score'>
```

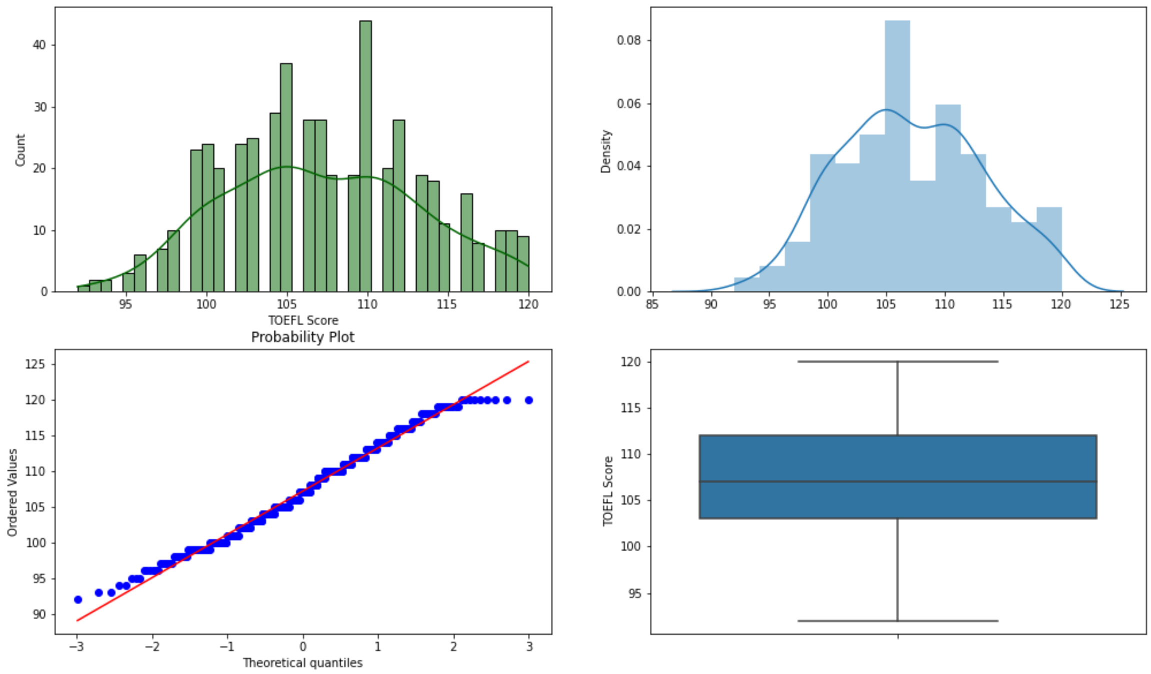


In [78]:

```
plt.rcParams["figure.figsize"]=(17,10)
plt.subplot(221)
sn.histplot(data=jamboree,x="TOEFL Score",bins=40,kde=True,color="darkgreen")
plt.subplot(222)
sn.distplot(x=jamboree["TOEFL Score"])
prob=stats.probplot(jamboree["TOEFL Score"],dist=stats.norm,plot=plt.subplot(223))
plt.subplot(224)
sn.boxplot(data=jamboree,y="TOEFL Score")
```

Out[78]:

```
<AxesSubplot:ylabel='TOEFL Score'>
```

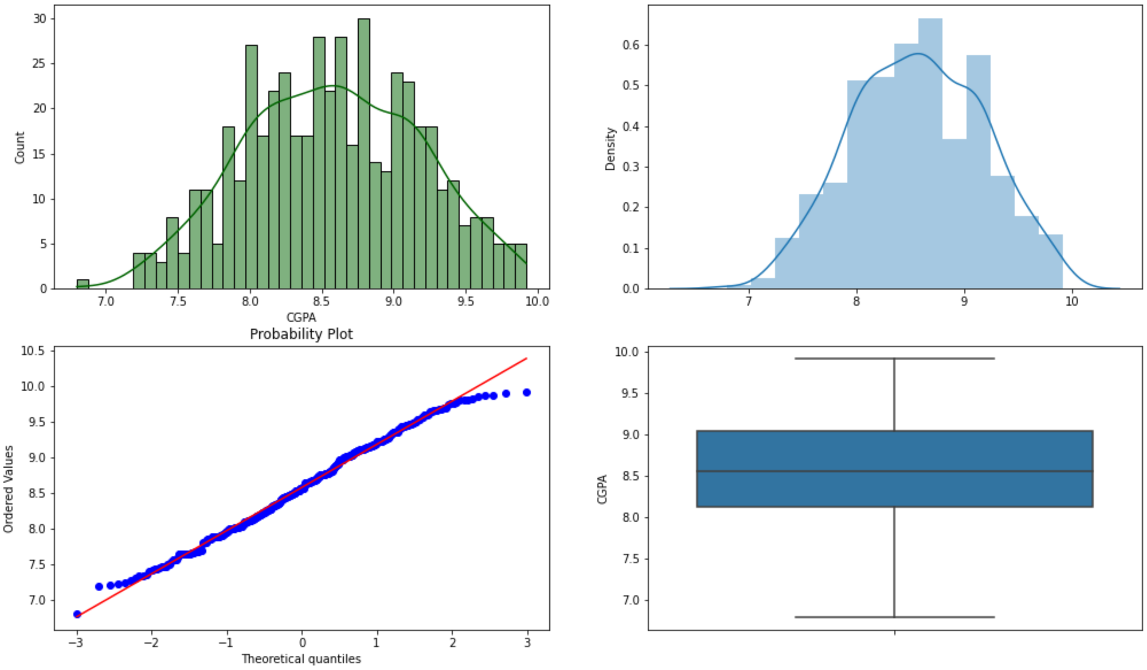


In [79]:

```
plt.rcParams["figure.figsize"]=(17,10)
plt.subplot(221)
sn.histplot(data=jamboree,x="CGPA",bins=40,kde=True,color="darkgreen")
plt.subplot(222)
sn.distplot(x=jamboree["CGPA"])
prob=stats.probplot(jamboree["CGPA"],dist=stats.norm,plot=plt.subplot(223))
plt.subplot(224)
sn.boxplot(data=jamboree,y="CGPA")
```

Out[79]:

```
<AxesSubplot:ylabel='CGPA'>
```

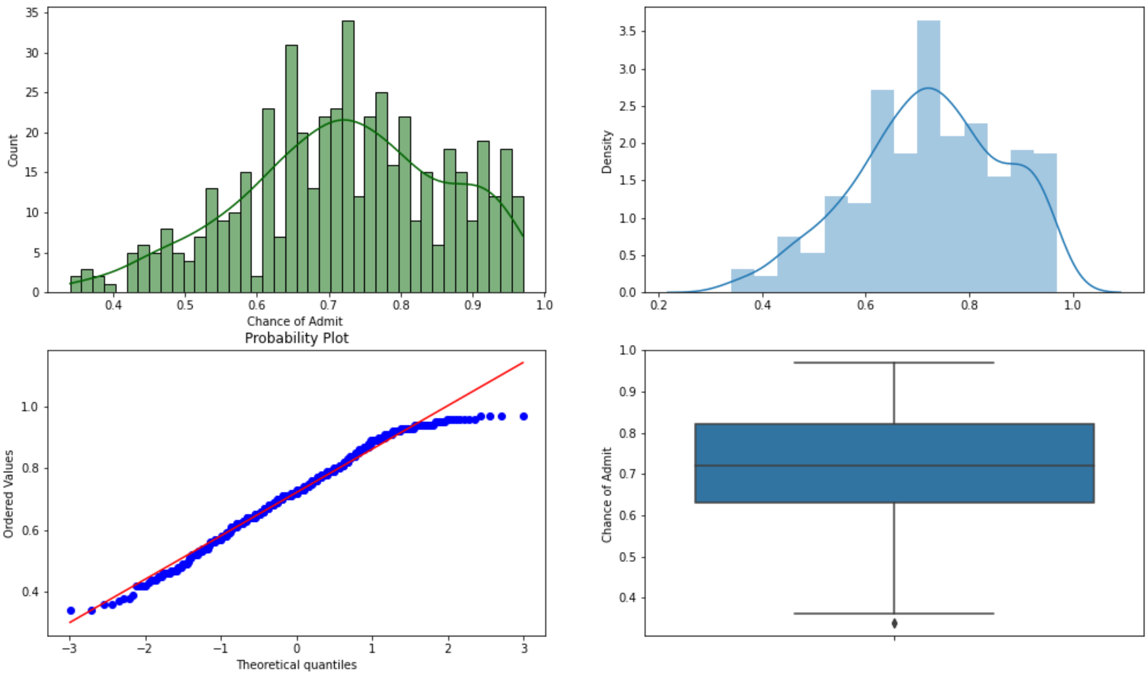


In [80]:

```
plt.rcParams["figure.figsize"]=(17,10)
plt.subplot(221)
sn.histplot(data=jamboree,x="Chance of Admit ",bins=40,kde=True,color="darkgreen")
plt.subplot(222)
sn.distplot(x=jamboree["Chance of Admit "])
prob=stats.probplot(jamboree["Chance of Admit "],dist=stats.norm,plot=plt.subplot(223))
plt.subplot(224)
sn.boxplot(data=jamboree,y="Chance of Admit ")
```

Out[80]:

```
<AxesSubplot:ylabel='Chance of Admit '>
```



Categorical variables

1.) From the distribution of Research,we can interpet that 56./. of people have already done research.

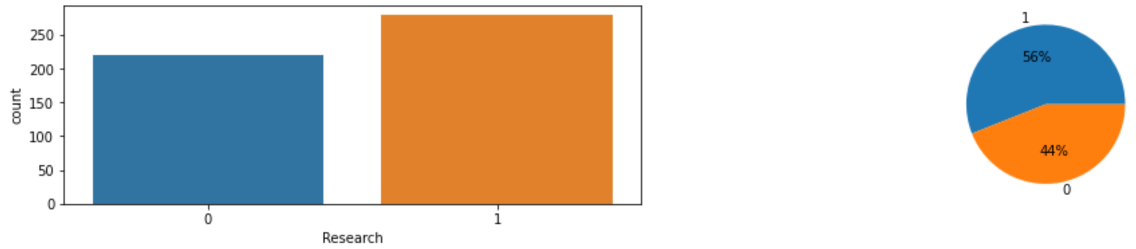
2.) From the distribution of SOP,we can interpet that majority of people have SOP strength 3.0,3.5,4.0. Very less number of people with SOP strength 1.0 have applied.So we can consider these as outliers.

3.)From the distribution of University Rating,we can interpet that very less number of people(7./.) have applied for universities with rating 1.

4.)From the distribution of LOR,we can interpet that majority of people have LOR strength 3.0,3.5,4.0.Very less number of people with LOR strength 1.0 have applied.So we can consider these as outliers.

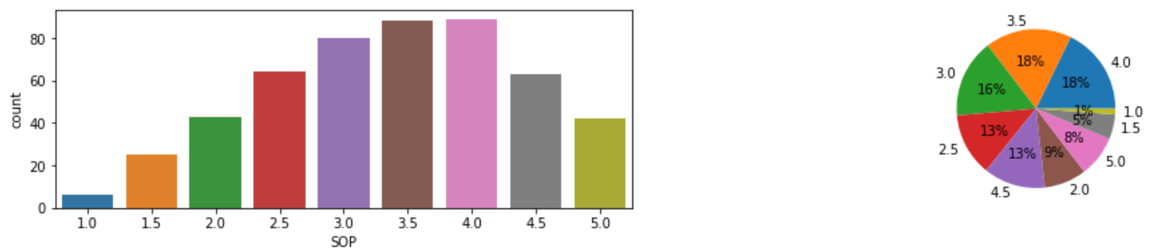
In [81]:

```
plt.rcParams["figure.figsize"]=(17,6)
plt.subplot(221)
sn.countplot(data=jamboree,x="Research")
plt.subplot(222)
# Finding the distribution of categorical variable
palette_color = sn.color_palette('dark')
k=jamboree["Research"].value_counts().reset_index()
plt.pie(k["Research"],labels=k["index"],autopct='%.0f%%')
plt.show()
```



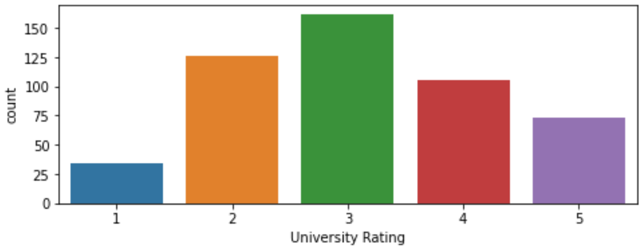
In [82]:

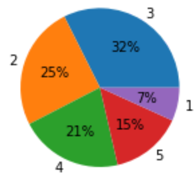
```
plt.rcParams["figure.figsize"]=(17,6)
plt.subplot(221)
sn.countplot(data=jamboree,x="SOP")
plt.subplot(222)
# Finding the distribution of categorical variable
palette_color = sn.color_palette('dark')
k=jamboree["SOP"].value_counts().reset_index()
plt.pie(k["SOP"],labels=k["index"],autopct='%.0f%%')
plt.show()
```



In [83]:

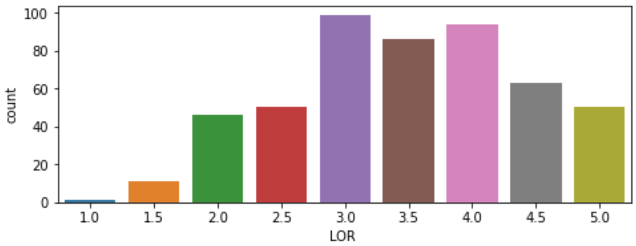
```
plt.rcParams["figure.figsize"]=(17,6)
plt.subplot(221)
sn.countplot(data=jamboree,x="University Rating")
plt.subplot(222)
# Finding the distribution of categorical variable
palette_color = sn.color_palette('dark')
k=jamboree["University Rating"].value_counts().reset_index()
plt.pie(k["University Rating"],labels=k["index"],autopct='%.0f%%')
plt.show()
```

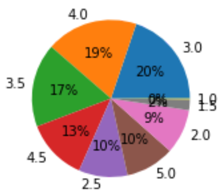




In [84]:

```
plt.rcParams["figure.figsize"]=(17,6)
plt.subplot(221)
sn.countplot(data=jamboree,x="LOR ")
plt.subplot(222)
# Finding the distribution of categorical variable
palette_color = sn.color_palette('dark')
k=jamboree["LOR "].value_counts().reset_index()
plt.pie(k["LOR "],labels=k["index"],autopct='%.0f%%')
plt.show()
```





outlier Detection

Removing outliers present in Chance of Admit variable

In [85]:

```
q3=np.percentile(jamboree["Chance of Admit "],75)
q1=np.percentile(jamboree["Chance of Admit "],25)
IQR=q3-q1
p100=np.percentile(jamboree["Chance of Admit "],100)
p0=np.percentile(jamboree["Chance of Admit "],0)
upper_whisker=min(q3+(1.5*IQR),p100)
lower_whisker=max(q1-(1.5*IQR),p0)
print(upper_whisker,lower_whisker)
jamboree[(jamboree["Chance of Admit "]>upper_whisker)].shape
```

```
0.97 0.3450000000000001
```

Out[85]:

```
(0, 8)
```

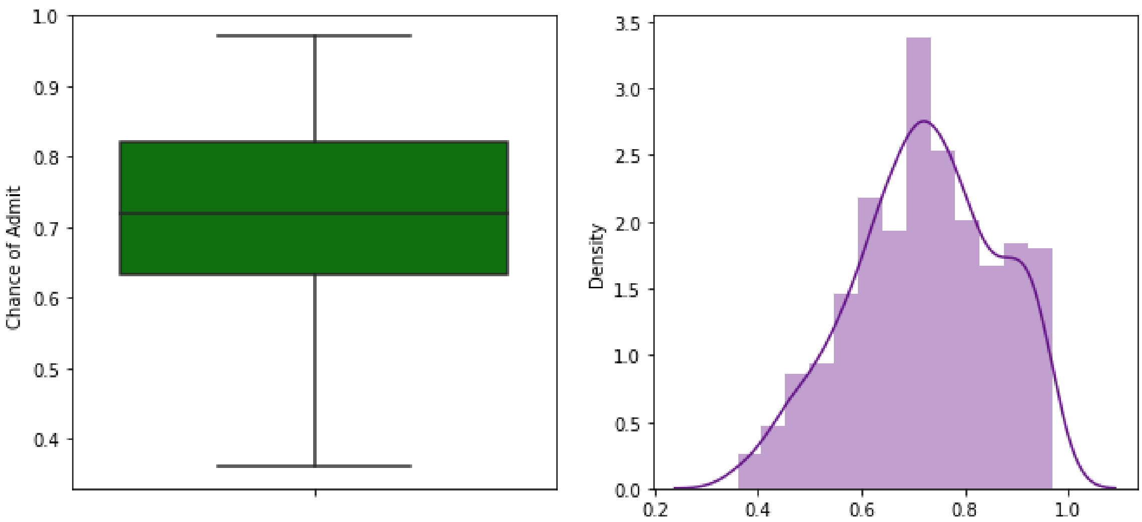
In [86]:

```
jamboree=jamboree[(jamboree["Chance of Admit "]>lower_whisker)]
print(jamboree.shape)
#for casual counts
plt.rcParams["figure.figsize"]=(17,5)
plt.subplot(131)
sn.boxplot(data=jamboree,y="Chance of Admit ",color="green")
plt.subplot(132)
sn.distplot(x=jamboree["Chance of Admit "],color="#641387")
```

```
(498, 8)
```

Out[86]:

```
<AxesSubplot:ylabel='Density'>
```



Removing outliers present in SOP,LOR variable

In [87]:

```
jamboree=jamboree[(jamboree["SOP"]!=1.0)&(jamboree["LOR "]!=1.0)]
jamboree["SOP"].unique()
```

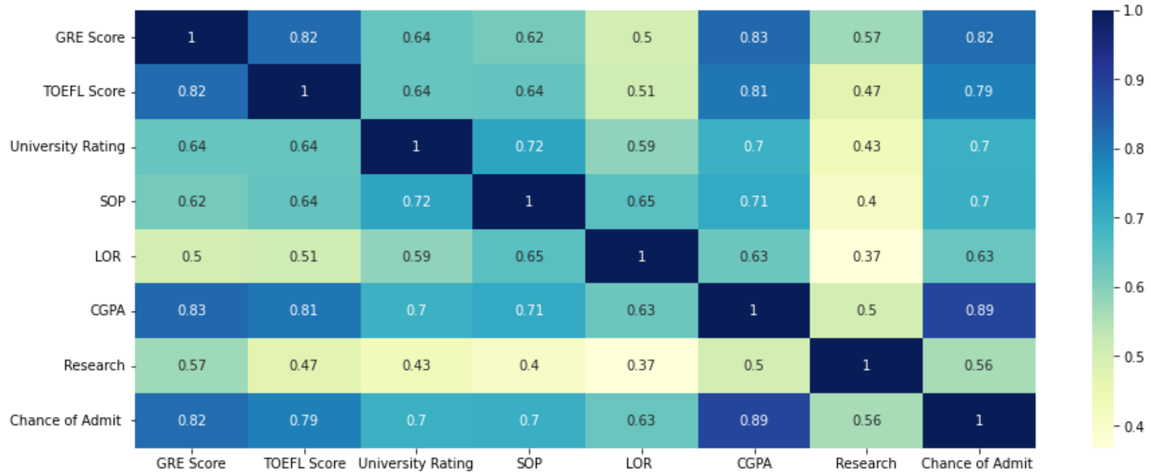
Out[87]:

```
array([4.5, 4. , 3. , 3.5, 2. , 5. , 1.5, 2.5])
```

Bivariate Analysis

In [88]:

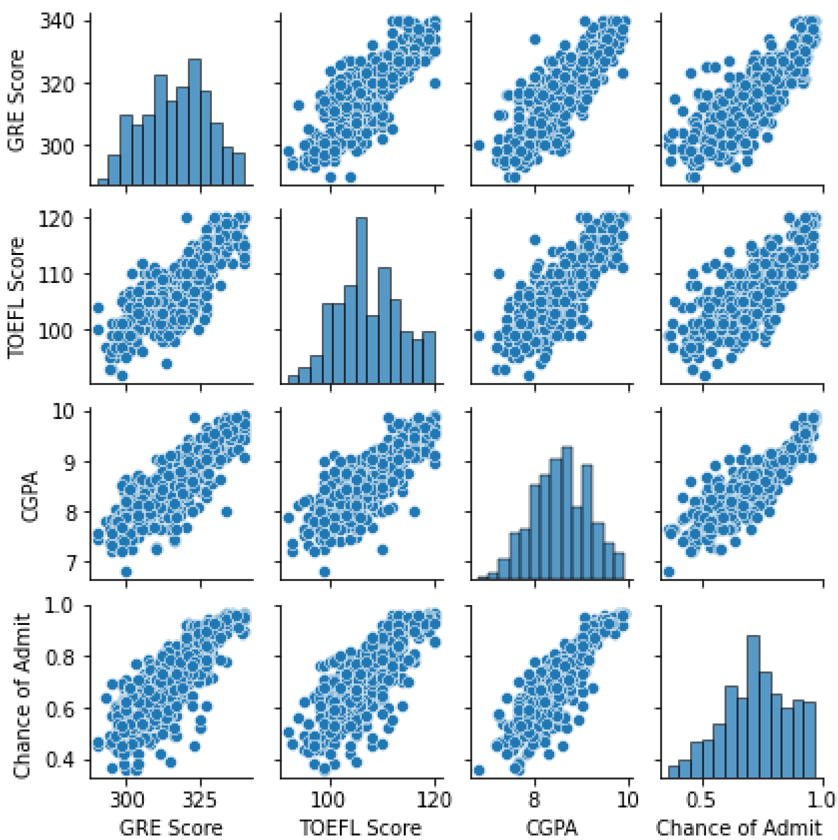
```
plt.figure(figsize=(15,6))
sn.heatmap(jamboree.corr(method="spearman"), cmap="YlGnBu", annot=True)
plt.show()
```



All independent variables have a positive correlation with target variable Chance of Admit

In [89]:

```
sn.pairplot(jamboree[["GRE Score","TOEFL Score","CGPA","Chance of Admit "]],height=1.5)
plt.show()
```



1.) If GRE score is more,the more is the chance of Admit.

2.) If TOEFL score is more,the more is the chance of Admit.

3.) If CGPA score is more,the more is the chance of Admit.

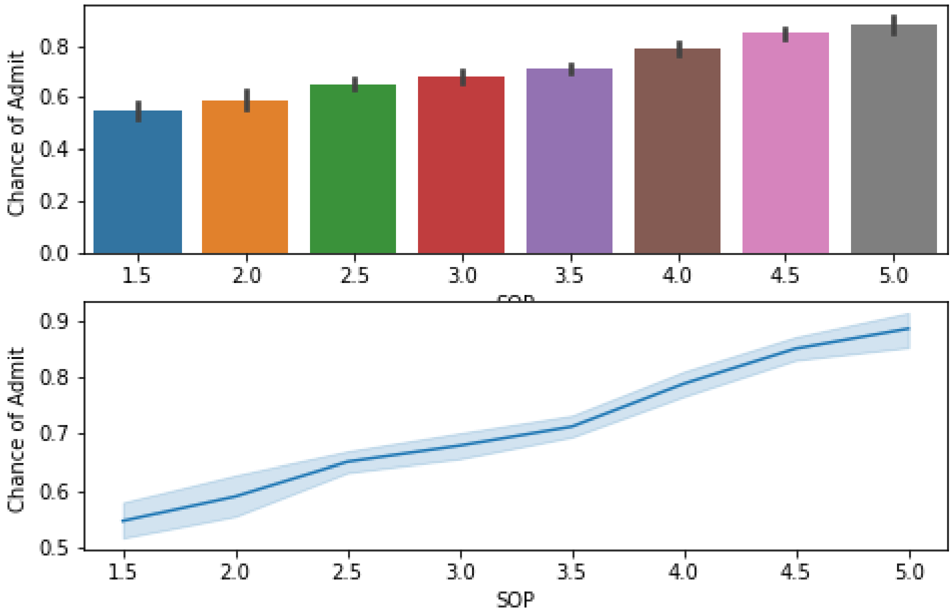
4.) If CGPA is more,the GRE and TOEFL score is more.

In [90]:

```
plt.subplot(221)
sn.barplot(data=jamboree,x="SOP",y="Chance of Admit ")
plt.subplot(223)
sn.lineplot(data=jamboree,x="SOP",y="Chance of Admit ")
```

Out[90]:

```
<AxesSubplot:xlabel='SOP', ylabel='Chance of Admit '>
```



In [ ]:

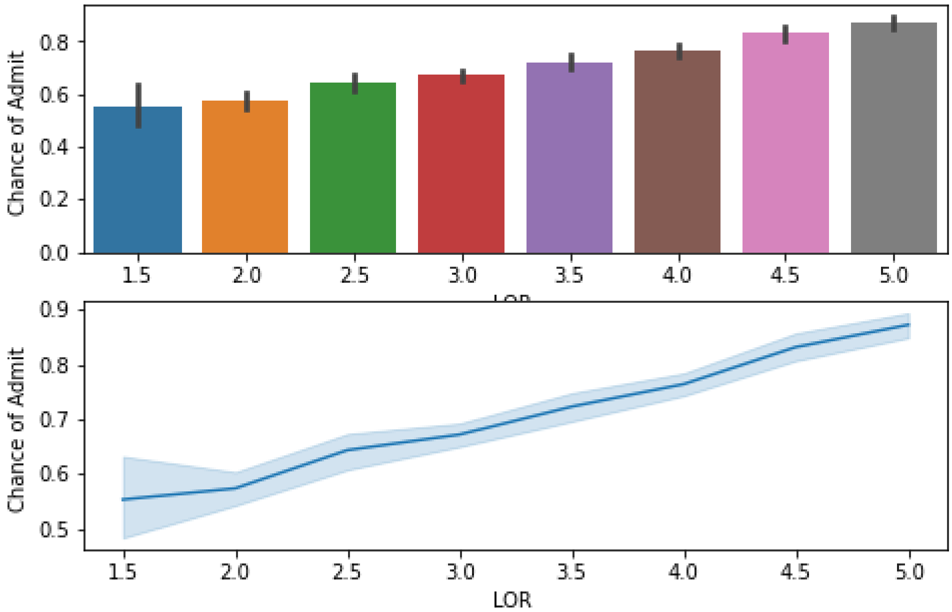
AS SOP strength increases, Chance of Admit increases

In [91]:

```
plt.subplot(221)
sn.barplot(data=jamboree,x="LOR ",y="Chance of Admit ")
plt.subplot(223)
sn.lineplot(data=jamboree,x="LOR ",y="Chance of Admit ")
```

Out[91]:

<AxesSubplot:xlabel='LOR ', ylabel='Chance of Admit '>



In [ ]:

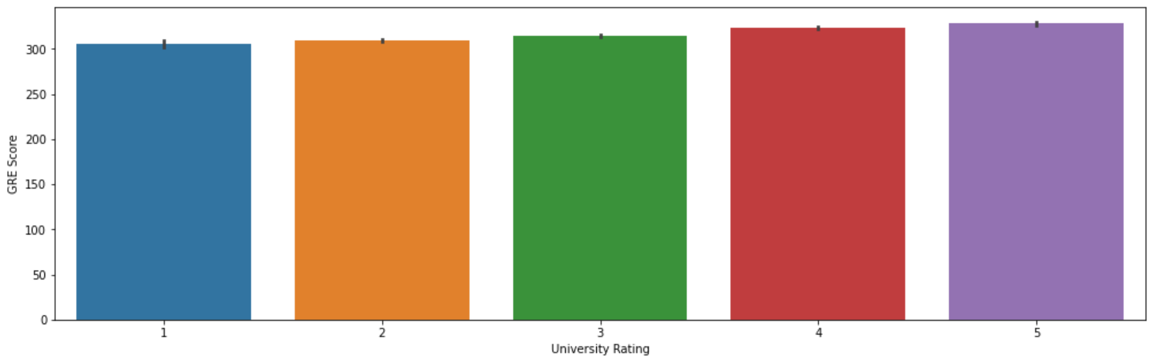
AS LOR strength increases, Chance of Admit increases

In [92]:

```
sn.barplot(data=jamboree,x="University Rating",y="GRE Score")
```

Out[92]:

<AxesSubplot:xlabel='University Rating', ylabel='GRE Score'>

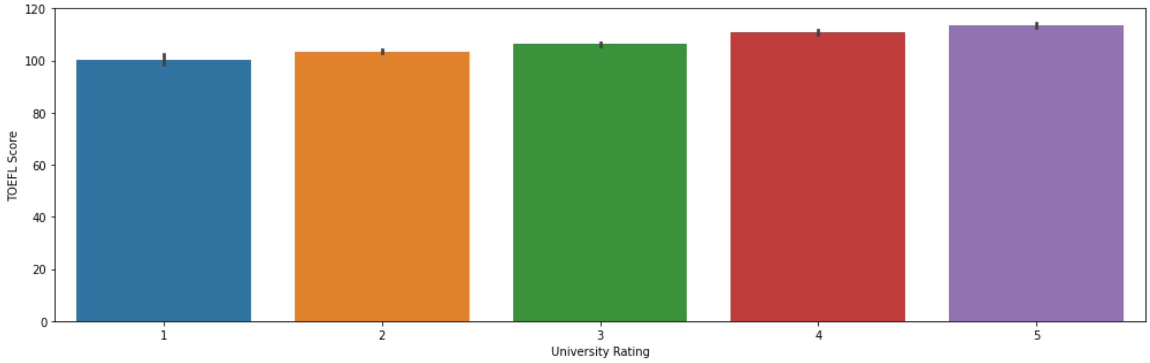


In [93]:

```
sn.barplot(data=jamboree,x="University Rating",y="TOEFL Score")
```

Out[93]:

```
<AxesSubplot:xlabel='University Rating', ylabel='TOEFL Score'>
```

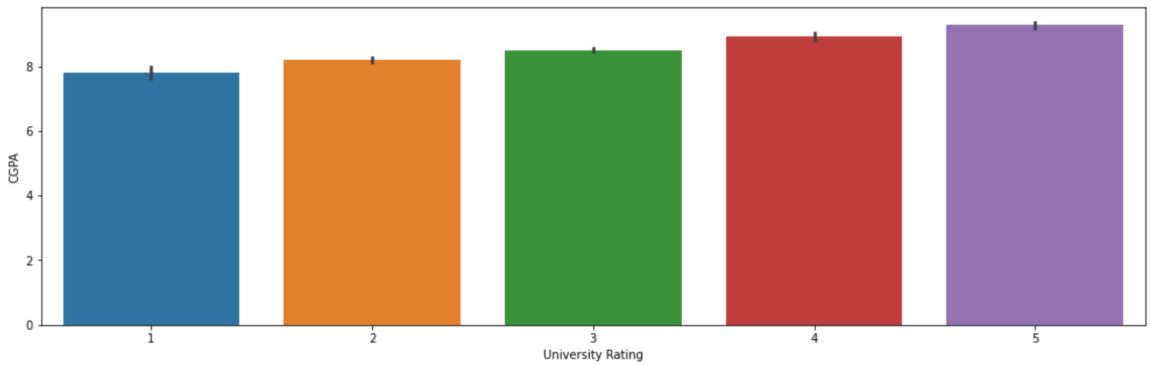


In [94]:

```
sn.barplot(data=jamboree,x="University Rating",y="CGPA")
```

Out[94]:

```
<AxesSubplot:xlabel='University Rating', ylabel='CGPA'>
```



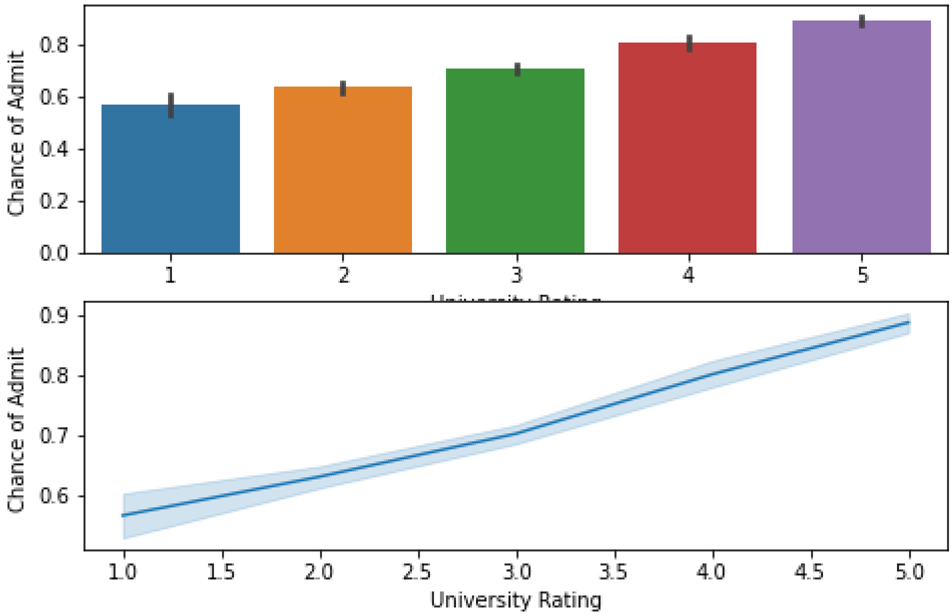
Higher university rating universitites have higher average GRE,TOEFL,CGPA scores.

In [95]:

```
plt.subplot(221)
sn.barplot(data=jamboree,x="University Rating",y="Chance of Admit ")
plt.subplot(223)
sn.lineplot(data=jamboree,x="University Rating",y="Chance of Admit ")
```

Out[95]:

```
<AxesSubplot:xlabel='University Rating', ylabel='Chance of Admit '>
```



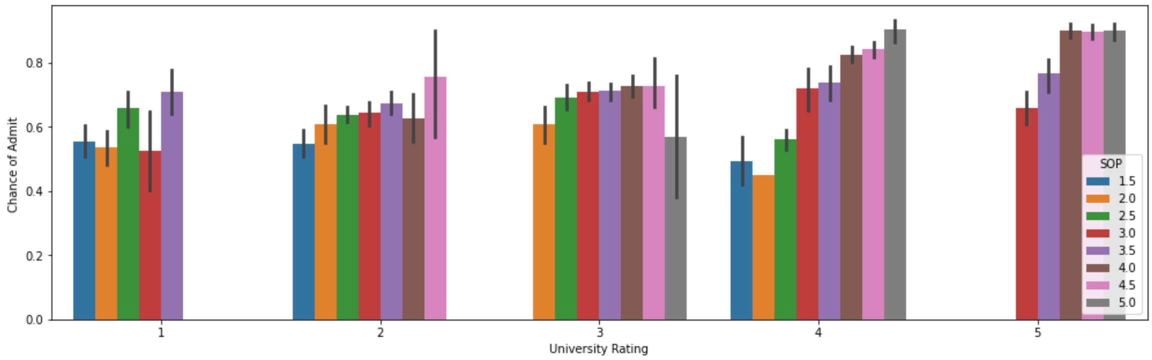
Higher university ratings,have higher chance of admission on an average.

In [96]:

```
sn.barplot(data=jamboree,x="University Rating",y="Chance of Admit ",hue="SOP")
```

Out[96]:

```
<AxesSubplot:xlabel='University Rating', ylabel='Chance of Admit '>
```



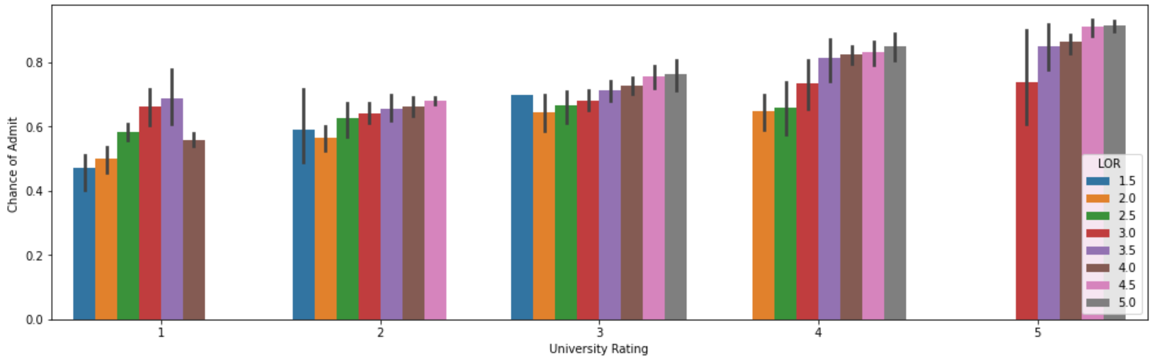
For university rating 4 and 5 ,students with SOP rating >=3 are only considered for admitting.

In [97]:

```
sn.barplot(data=jamboree,x="University Rating",y="Chance of Admit ",hue="LOR ")
```

Out[97]:

```
<AxesSubplot:xlabel='University Rating', ylabel='Chance of Admit '>
```



In Universities whith rating 4 and 5,LOR and SOP strength plays a major role in chance of admit i.e; higher the LOR and SOP strength higher is the chance of getting admit.In remaining universities the difference is very less.

Candidates with LOR Strength =5 have applied only to universities having rating >3 and they've higher chance of getting admitted.

For university rating 5,LOR strength below 3 are not considered.

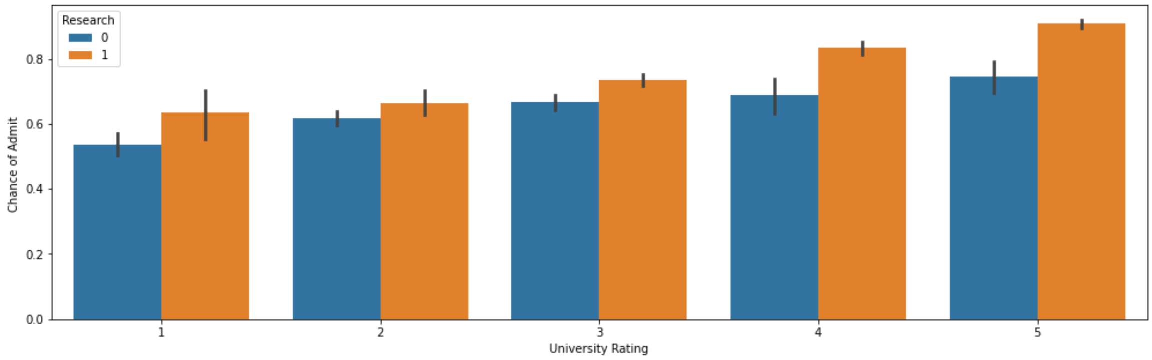
Students with LOR rating <3 have higher chance of admitting in university ratings<=3

In [98]:

```
sn.barplot(data=jamboree,x="University Rating",y="Chance of Admit ",hue="Research")
```

Out[98]:

```
<AxesSubplot:xlabel='University Rating', ylabel='Chance of Admit '>
```



In across all universities with different university ratings,people who did research have higher chance of admit.

In universities of rating >3,student's who have done research previously has a higher chance of getting admitted.

Data Preparation For Model

In [99]:

```
from sklearn.model_selection import train_test_split

X = jamboree.iloc[:, :-1]
y = jamboree.iloc[:, -1]
X.head()
```

Out[99]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	337	118	4	4.5	4.5	9.65	1
1	324	107	4	4.0	4.5	8.87	1
2	316	104	3	3.0	3.5	8.00	1
3	322	110	3	3.5	2.5	8.67	1
4	314	103	2	2.0	3.0	8.21	0

In [100]:

```
# split the dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=0)
```

In [101]:

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()

scaler = scaler.fit(X_train.to_numpy())
X_trainscaled=scaler.transform(X_train)
X_testscaled=scaler.transform(X_test)

X_trainscaled1=pd.DataFrame(X_trainscaled,columns=X_train.columns)
X_trainscaled1
```

Out[101]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	-1.290518	-0.881057	-1.865230	-1.413148	-2.202275	-0.965115	-1.151279
1	1.841288	1.653756	1.637926	1.686576	1.686337	2.185949	0.868599
2	-0.932597	-0.712070	-0.113652	0.136714	-0.535727	-0.611253	-1.151279
3	-0.932597	0.470843	-0.989441	0.136714	0.575305	-0.206838	-1.151279
4	-1.737919	-1.219033	-1.865230	-1.929769	-1.646759	-1.133622	-1.151279
...	...	...	...	...	...	...	...
388	0.678046	0.808818	0.762137	0.653335	0.019789	0.332381	0.868599
389	-0.037796	0.301856	-0.113652	0.136714	-0.535727	0.315531	-1.151279
390	0.946486	-0.543082	1.637926	-0.379907	0.019789	0.450336	0.868599
391	0.409605	0.470843	-0.113652	0.136714	1.686337	0.467186	0.868599
392	0.409605	0.639831	0.762137	0.653335	0.575305	0.669394	0.868599

393 rows × 7 columns

In [102]:

```
col=['const']+list(X.columns)
```

Model Building

In [103]:

```
import statsmodels.api as sm
X_sm = sm.add_constant(X_trainscaled)#Statmodels default is without intercept, to add i
ntercept we need to add constant
sm_model = sm.OLS(y_train, X_sm).fit()
print(X_sm)
```

```
[[ 1.         -1.29051813 -0.88105736 ... -2.20227468 -0.96511547
  -1.1512792 ]
 [ 1.          1.84128795  1.65375628 ...  1.68633742  2.1859486
   0.86859904]
 [ 1.         -0.93259744 -0.71206978 ... -0.53572664 -0.61125266
  -1.1512792 ]
 ...
 [ 1.          0.94648622 -0.54308221 ...  0.01978937  0.45033577
   0.86859904]
 [ 1.          0.40960517  0.47084325 ...  1.68633742  0.46718638
   0.86859904]
 [ 1.          0.40960517  0.63983082 ...  0.57530539  0.6693937
   0.86859904]]
```

In [104]:

```
print(sm_model.summary(xname=['const']+list(X.columns)))
```

```
                            OLS Regression Results
==========================================================================
====
Dep. Variable:      Chance of Admit   R-squared:
0.820
Model:                          OLS   Adj. R-squared:
0.817
Method:               Least Squares   F-statistic:                       2
50.6
Date:              Mon, 15 Aug 2022   Prob (F-statistic):            4.19e
-139
Time:                      22:55:45   Log-Likelihood:                   55
3.77
No. Observations:               393   AIC:                              -1
092.
Df Residuals:                   385   BIC:                              -1
060.
Df Model:                         7
Covariance Type:          nonrobust
==========================================================================
===========
                       coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------
-----------
const                0.7215      0.003    239.401      0.000       0.716
0.727
GRE Score            0.0226      0.006      3.690      0.000       0.011
0.035
TOEFL Score          0.0116      0.006      2.009      0.045       0.000
0.023
University Rating    0.0065      0.005      1.364      0.173      -0.003
0.016
SOP                  0.0085      0.005      1.689      0.092      -0.001
0.018
LOR                  0.0137      0.004      3.276      0.001       0.005
0.022
CGPA                 0.0691      0.006     10.736      0.000       0.056
0.082
Research             0.0135      0.004      3.738      0.000       0.006
0.021
==========================================================================
====
Omnibus:                     85.668   Durbin-Watson:
2.118
Prob(Omnibus):                0.000   Jarque-Bera (JB):                18
6.720
Skew:                        -1.122   Prob(JB):                       2.85
e-41
Kurtosis:                     5.524   Cond. No.
5.44
==========================================================================
====

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

In [105]:

```
from sklearn.linear_model import LinearRegression
model=LinearRegression().fit(X_trainscaled,y_train)
print(model.score(X_testscaled,y_test))
```

```
0.8169706085273892
```

The training and testing scores are similar ,hence the model is good fit.But we need to check whether it's performance will increase using regularization or using higher degrees of the polynomial.

Regularisation Using Lasso and Ridge Regression

Using Cross Validation

In [106]:

```
from sklearn.model_selection import train_test_split
#0.6, 0.2, 0.2 split
X_tr_cv, X_test, y_tr_cv, y_test = train_test_split(X, y, test_size=0.2, random_state=1
)
X_train, X_val, y_train, y_val = train_test_split(X_tr_cv, y_tr_cv, test_size=0.25,rand
om_state=1)
```

Fixing the degree of polynomial

In [107]:

```
# Train and Validatation without hyper param tuning. Just by controlling the degree
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline
max_degree = 11 # max polynomial degree
train_scores = []
val_scores = []
scaler = StandardScaler()
for degree in range(1, max_degree):
    polyreg_scaled = make_pipeline(PolynomialFeatures(degree), scaler, LinearRegression
())
    polyreg_scaled.fit(X_train, y_train)
    train_score = polyreg_scaled.score(X_train, y_train)
    val_score = polyreg_scaled.score(X_val, y_val)
    train_scores.append(train_score)
    val_scores.append(val_score)
```

In [108]:

```
print(train_scores)
print(val_scores)
```

```
[0.812029789336635, 0.8359680013829242, 0.867035686548884, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0]
[0.8295010004161876, 0.7915391460548464, 0.6018025621131762, -1159.2420392
17745, -82.06884555017689, -61.31307500717507, -57.65394601129486, -61.630
23398563677, -68.18075191705691, -74.99004632648047]
```

The best degree that we can use is degree=1 which is not but linear equation

Ridge Regression

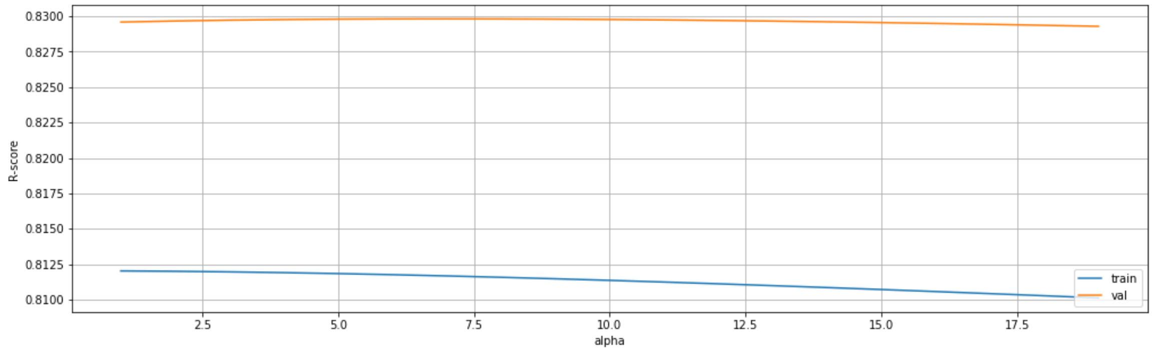
In [109]:

```
from sklearn.linear_model import Ridge
train_scores = []
val_scores = []
scaler = StandardScaler()
for alpha in range(1,20):
    scaler = scaler.fit(X_train.to_numpy())
    X_trainscaled=scaler.transform(X_train)
    X_valscaled=scaler.transform(X_val)
    polyreg_scaledridge =  Ridge(alpha)
    polyreg_scaledridge.fit(X_trainscaled, y_train)
    train_score = polyreg_scaledridge.score(X_trainscaled, y_train)
    val_score = polyreg_scaledridge.score(X_valscaled, y_val)
    train_scores.append(train_score)
    val_scores.append(val_score)
print(val_scores)
```

```
[0.8295955068707989, 0.8296709920767912, 0.8297293206431087, 0.82977216214
63559, 0.8298010136996306, 0.8298172196188813, 0.8298219885989411, 0.82981
640874732, 0.8298014607705805, 0.8297780295637117, 0.8297469144157702, 0.8
297088380138902, 0.8296644544015503, 0.8296143560248572, 0.829559079981898
2, 0.8294991135743388, 0.8294348992469532, 0.8293668389892641, 0.829295298
2636534]
```

In [110]:

```
plt.figure()
plt.plot(list(range(1, 20)), train_scores, label="train")
plt.plot(list(range(1, 20)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("alpha")
plt.ylabel("R-score")
plt.grid()
plt.show()
```



In [111]:

```
np.argmax(val_scores)
```

Out[111]:

6

so we fix the max value of alpha to be 6

Lasso Regression

In [112]:

```
from sklearn.linear_model import Lasso
train_scores = []
val_scores = []
scaler = StandardScaler()
for alpha in range(1,20):
    scaler = scaler.fit(X_train.to_numpy())
    X_trainscaled=scaler.transform(X_train)
    X_valscaled=scaler.transform(X_val)
    polyreg_scaledlasso =  Lasso(alpha)

    polyreg_scaledlasso.fit(X_trainscaled, y_train)
    train_score = polyreg_scaledlasso.score(X_trainscaled, y_train)
    val_score = polyreg_scaledlasso.score(X_valscaled, y_val)
    train_scores.append(train_score)
    val_scores.append(val_score)
print(val_scores)
```

```
[-0.0069937666198001125, -0.0069937666198001125, -0.0069937666198001125, -
0.0069937666198001125, -0.0069937666198001125, -0.0069937666198001125, -0.
0069937666198001125, -0.0069937666198001125, -0.0069937666198001125, -0.00
69937666198001125, -0.0069937666198001125, -0.0069937666198001125, -0.0069
937666198001125, -0.0069937666198001125, -0.0069937666198001125, -0.006993
7666198001125, -0.0069937666198001125, -0.0069937666198001125, -0.00699376
66198001125]
```

From the above ,we can clearly observe that lasso is not working here.So we'll fix with ridge

In [113]:

```
scaler = scaler.fit(X_train.to_numpy())
X_trainscaled=scaler.transform(X_train)
X_valscaled=scaler.transform(X_val)
polyreg_scaledridge =  Ridge(alpha=6)
polyreg_scaledridge.fit(X_trainscaled, y_train)
train_score = polyreg_scaledridge.score(X_trainscaled, y_train)
val_score = polyreg_scaledridge.score(X_valscaled, y_val)
print("Train score::",train_score)
print("validation score::",val_score)
```

```
Train score:: 0.8117555919849093
validation score:: 0.8298172196188813
```

In [114]:

```
X_testscaled=scaler.transform(X_test)
polyreg_scaledridge.score(X_testscaled, y_test)
```

Out[114]:

```
0.8290955509310067
```

AS you can see,the performance of unseen data,has increased incomparison to normal linear regression.So we'll use this model

In [115]:

```
model=polyreg_scaledridge
y_hattrain = model.predict(X_trainscaled)

y_hattest= model.predict(X_testscaled)
residuals=(y_train-y_hattrain)
```

Displaying Model Coefficients

In [116]:

```
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error

X_train,X_test=X_trainscaled,X_testscaled
cols=np.array(list(X.columns))
coef=model.coef_
df=pd.DataFrame({"predictor":cols.reshape(7,),"weights":coef})
df
```

Out[116]:

	predictor	weights
0	GRE Score	0.021728
1	TOEFL Score	0.014096
2	University Rating	0.006583
3	SOP	0.012110
4	LOR	0.011828
5	CGPA	0.068192
6	Research	0.008567

Model Performance Evaluation

The Evaluation metrices for both training and test data are similar,hence the model is a good fit

In [117]:

```
print("R2-squared")
print("------------")
print("Training score")
print("------------")
print(model.score(X_train,y_train))
print("------------")
print("Test score")
print("------------")
print(model.score(X_test,y_test))
```

```
R2-squared
------------
Training score
------------
0.8117555919849093
------------
Test score
------------
0.8290955509310067
```

In [118]:

```
print("MAE")
print("------------")
print("Training")
print("------------")
print(mean_absolute_error(y_train, y_hattrain))
print("------------")
print("Test")
print("------------")
print(mean_absolute_error(y_test, y_hattest))
```

```
MAE
------------
Training
------------
0.043589746973142295
------------
Test
------------
0.04076335810725591
```

In [119]:

```
import math
print("RMSE")
print("------------")
print("Training")
print("------------")
print(math.sqrt(mean_squared_error(y_train, y_hattrain)))
print("------------")
print("Test")
print("------------")
print(math.sqrt(mean_squared_error(y_test, y_hattest)))
```

```
RMSE
------------
Training
------------
0.06104743334215431
------------
Test
------------
0.057124631280847764
```

In [120]:

```
n=X_test.shape[0]
p=X_test.shape[1]
print("Adj R2-squared")
print("------------")
print("Training score")
rt=model.score(X_train,y_train)
print("------------")
adjr2_sq = (n-1)/(n-p-1)
adjr2_sq=1-(adjr2_sq*(1-rt))
print(adjr2_sq)
print("------------")
print("Test score")
print("------------")
rt=model.score(X_test,y_test)
print("------------")
adjr2_sq = (n-1)/(n-p-1)
adjr2_sq=1-(adjr2_sq*(1-rt))
print(adjr2_sq)
```

```
Adj R2-squared
------------
Training score
------------
0.7972752529068254
------------
Test score
------------
------------
0.8159490548487764
```

Testing the assumptions of the linear regression model

Multicollinearity check by VIF score (variables are dropped one-by-one till none has VIF>5)

In [121]:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [122]:

```
vif = pd.DataFrame()
X_t = X_trainscaled1
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[122]:

	Features	VIF
5	CGPA	4.56
0	GRE Score	4.14
1	TOEFL Score	3.68
3	SOP	2.80
2	University Rating	2.51
4	LOR	1.93
6	Research	1.43

None of the variables have VIF >5,so there is no multicollinearity with any variables.

Mean of residuals should be close to zero

In [123]:

```
round(np.mean(residuals),2)
```

Out[123]:

```
0.0
```

Mean of residuals is close to zero

Linearity of variables

From the EDA and correlation's of heatmap , we can observe that all independent variables have linear relation with dependent variable

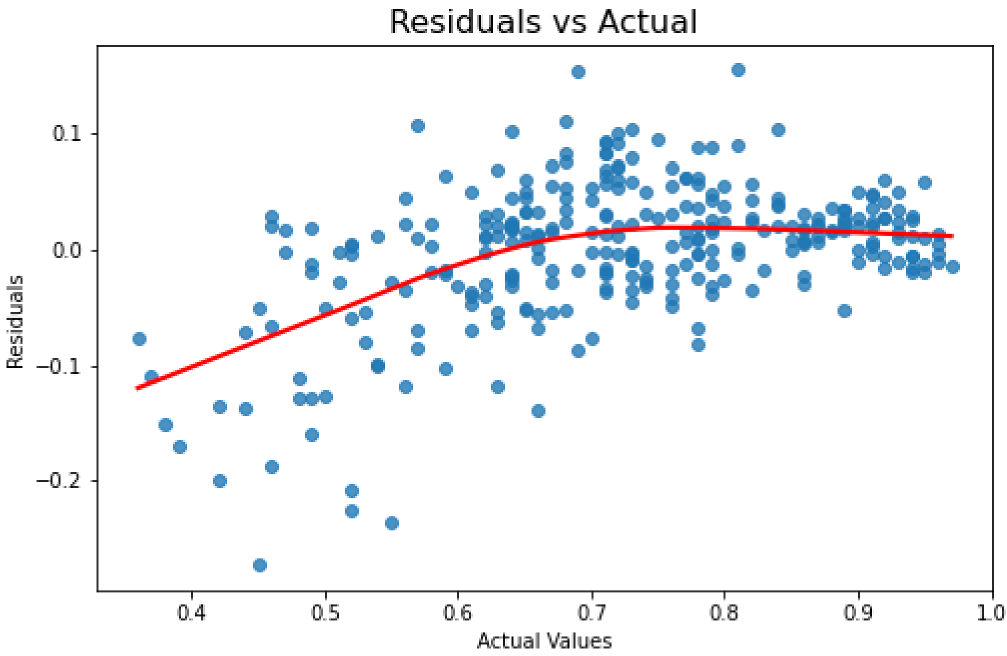
Test for Homoscedasticity

In [124]:

```
plt.figure(figsize=(8,5))
sn.regplot(x=y_train, y=residuals, lowess=True, line_kws={'color': 'red'})
plt.title('Residuals vs Actual', fontsize=16)
plt.xlabel('Actual Values')
plt.ylabel('Residuals')
```

Out[124]:

```
Text(0, 0.5, 'Residuals')
```

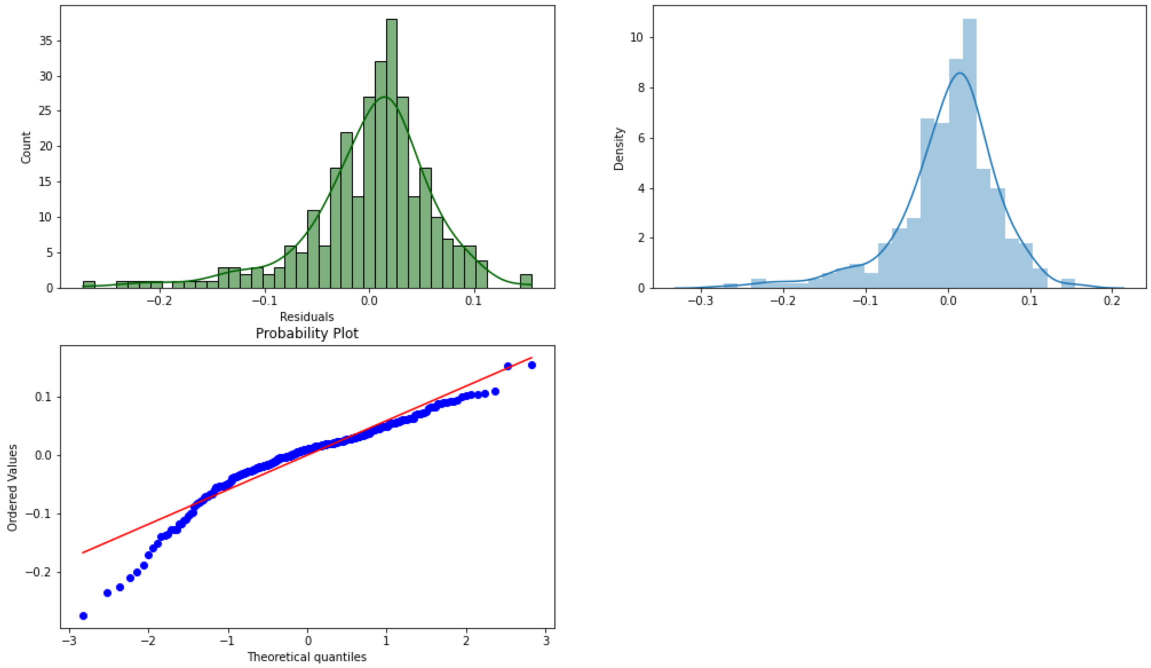


Residuals are homoscedastic in nature

Normality of residuals (almost bell-shaped curve in residuals distribution, points in QQ plot are almost all on the line)

In [125]:

```
plt.rcParams["figure.figsize"]=(17,10)
plt.subplot(221)
sn.histplot(x=residuals,bins=40,kde=True,color="darkgreen")
plt.xlabel("Residuals")
plt.subplot(222)
sn.distplot(x=residuals)
prob=stats.probplot(residuals,dist=stats.norm,plot=plt.subplot(223))
```



Distribution of residuals is close to normal

All of the assumptions are true,hence we can use linear Regression for this problem

Comments on significance of predictor variables

In [126]:

```
df.sort_values(by="weights")
```

Out[126]:

	predictor	weights
2	University Rating	0.006583
6	Research	0.008567
4	LOR	0.011828
3	SOP	0.012110
1	TOEFL Score	0.014096
0	GRE Score	0.021728
5	CGPA	0.068192

Based on the above,information we can say that more the magnitude of weights,more is the signifance of the predictor

1.) CGPA has most significance among all the variables. This might be the important factor for getting admitted in universities.

2.) University Rating doesn't impact the chance of admission,this is the least significant factor in getting admitted in universities.

3.)order of magnitude of weights is CGPA>GRE score>TOEFL score>SOP>LOR>Research.

4.)signifance of predictors is CGPA>GRE score>TOEFL score>SOP>LOR>Research.

5.) All independent variables have a positive correlation with target variable Chance of Admit.

6.) If GRE score is more,the more is the chance of Admit.

7.) If TOEFL score is more,the more is the chance of Admit.

8.) If CGPA score is more,the more is the chance of Admit.

9.) If CGPA is more,the GRE and TOEFL score is more.

10.) AS SOP,LOR strength increases, Chance of Admit increases

11.) Higher university rating universitites have higher average GRE,TOEFL,CGPA scores.

12.) Higher university ratings,have higher chance of admission on an average.

13.) For university rating 4 and 5 ,students with SOP rating >=3 are only considered for admitting.

14.) In Universities whith rating 4 and 5,LOR and SOP strength plays a major role in chance of admit i.e; higher the LOR and SOP strength higher is the chance of getting admit.In remaining universities the difference is very less.

15.) Candidates with LOR Strength =5 have applied only to universities having rating >3 and they've higher chance of getting admitted.

16.) For university rating 5,LOR strength below 3 are not considered.

17.) Students with LOR rating <3 have higher chance of admitting in university ratings<=3.

18.) In across all universities with different university ratings,people who did research have higher chance of admit.

19.) In universities of rating >3,student's who have done research previously has a higher chance of getting admitted.

In [ ]:

```

```

Comments on additional data sources for model improvement, model implementation in real world, potential business benefits from improving the model

1.) Course applied by the candidate can be added, chance of getting admitted can also be depend on type of course enrolled.

2.) Number of work experience years can be added,people having relavant work experience for a course will have higher chance of admission.

3.) Some univeristies, even might consider candidates based on work experience.

4.) State in which university is located can be added, chance of getting can differ based from state to state.

5.) IELTS score should be added, now days most of universities are considering this as well.

6.) These additional data will influence the chance of admission .

7.) This will improve performance of model by giving more accurate predictions because more the data more is the accuracy of prediction and can give more insights from the data as well.

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```