jamboree-education

December 4, 2023

Jamboree is India's leading institute for study abroad test prep and admission counselling. With the highest scores for GMAT and GRE in the industry and admission offers from the best universities worldwide, Jamboree has helped thousands of students get into their dream universities.

They recently launched a feature where students/learners can come to their website and check their probability of getting into the IVY league college. This feature estimates the chances of graduate admission from an Indian perspective.

0.0.1 About

0.1 Problem Statement

Analysis to help Jamboree in understanding what factors are important in graduate admissions and how these factors are interrelated among themselves. It will also help predict one's chances of admission given the rest of the variables.

0.1.1 Column Profiling:

Serial No.: This is a unique row ID corresponding to each of the student. It is of integer type.

GRE Score: Marks scored in GRE test, out of 340. It is of integer type.

TOEFL Score: Marks scored in TOEFL test, out of 120. .It is of integer type

University Rating (out of 5): Rating of the university, out of 5. It is of integer type.

Statement of Purpose and Letter of Recommendation Strength: Strength of recommendation letter or of SOP, out of 5. It is of float type.

Undergraduate GPA: Grade secured in undergraduate program, out of 10. It is of float type.

Research Experience: If the student has any research experience, (either 0 or 1). It is of integer type.

Chance of Admit: Chance of getting admission, ranging from 0 to 1. It is of float type.

0.2 Read Data

[ ]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import stat as st
import statsmodels.api as stm

import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

[ ]:

```
df= pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/
  ↪001/839/original/Jamboree_Admission.csv")
```

[ ]:

```
df.head()
```

[ ]:

```
   Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  \
0           1        337          118                  4  4.5   4.5  9.65
1           2        324          107                  4  4.0   4.5  8.87
2           3        316          104                  3  3.0   3.5  8.00
3           4        322          110                  3  3.5   2.5  8.67
4           5        314          103                  2  2.0   3.0  8.21

   Research  Chance of Admit
0         1             0.92
1         1             0.76
2         1             0.72
3         1             0.80
4         0             0.65
```

[ ]:

```
df.shape
```

[ ]:

```
(500, 9)
```

There are 500 observations and 9 features

[ ]:

```
df.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
```

We are dropping the unique row identifier, as we don't want our model to build some understanding based on row numbers.

```
[ ]: df = df.drop('Serial No.', axis = 1)
```

```
[ ]: df.shape
```

```
[ ]: (500, 8)
```

Now we have 500 observations and 8 features.

Checking for null values:

```
[ ]: df.isnull().sum()
```

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

There are no NULL values.

Checking for duplicates:

```
[ ]: df.duplicated().sum()
```

```
[ ]: 0
```

There are no duplicate values.

Number of unique values:

```
[ ]: df.nunique(dropna=False)
```

```
[ ]: GRE Score             49
     TOEFL Score           29
     University Rating      5
     SOP                    9
     LOR                    9
     CGPA                 184
     Research               2
     Chance of Admit       61
     dtype: int64
```

```
[ ]: df.describe(include = 'all').T
```

```
[ ]:                    count       mean        std     min       25%     50%  \
     GRE Score           500.0  316.47200  11.295148  290.00  308.0000  317.00
     TOEFL Score         500.0  107.19200   6.081868   92.00  103.0000  107.00
     University Rating   500.0    3.11400   1.143512    1.00    2.0000    3.00
     SOP                 500.0    3.37400   0.991004    1.00    2.5000    3.50
     LOR                 500.0    3.48400   0.925450    1.00    3.0000    3.50
     CGPA                500.0    8.57644   0.604813    6.80    8.1275    8.56
     Research            500.0    0.56000   0.496884    0.00    0.0000    1.00
     Chance of Admit     500.0    0.72174   0.141140    0.34    0.6300    0.72

                            75%     max
     GRE Score           325.00  340.00
     TOEFL Score         112.00  120.00
     University Rating     4.00    5.00
     SOP                   4.00    5.00
     LOR                   4.00    5.00
     CGPA                  9.04    9.92
     Research              1.00    1.00
     Chance of Admit       0.82    0.97
```

Observations:

1. The minimum, maximum and average GRE scores are 290, 317 and 316.47200 respectively.
2. The minimum, maximum and avarage TOEFL scores are 92, 120 and 107.19200 respectively.
3. The minimum, maximum and avarage CGPA are 6.80, 9.92 and 8.57644 respectively.
4. The minimum, maximum and avarage chance of admission are 0.34, 0.97 and 0.72174 respectively.

##**Univariate Analysis:**

```
[ ]: for i in [   'University Rating', 'SOP','LOR ', 'CGPA', 'Research']:
         print(df[i].value_counts(normalize=True)*100)
         print('_'*50)
```

```
3    32.4
2    25.2
4    21.0
5    14.6
1     6.8
Name: University Rating, dtype: float64
__________________________________________________
4.0    17.8
3.5    17.6
3.0    16.0
2.5    12.8
4.5    12.6
2.0     8.6
```

```
5.0     8.4
1.5     5.0
1.0     1.2
Name: SOP, dtype: float64
--------------------------------------------------
3.0    19.8
4.0    18.8
3.5    17.2
4.5    12.6
2.5    10.0
5.0    10.0
2.0     9.2
1.5     2.2
1.0     0.2
Name: LOR , dtype: float64
--------------------------------------------------
8.76    1.8
8.00    1.8
8.12    1.4
8.45    1.4
8.54    1.4
       …
9.92    0.2
9.35    0.2
8.71    0.2
9.32    0.2
7.69    0.2
Name: CGPA, Length: 184, dtype: float64
--------------------------------------------------
1    56.0
0    44.0
Name: Research, dtype: float64
--------------------------------------------------
```

[ ]:

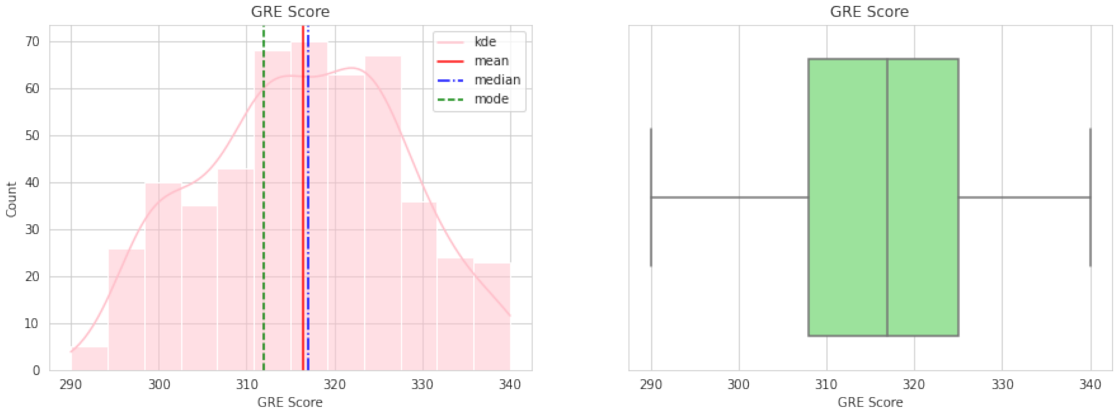
```python
X = df.drop(columns='Chance of Admit ')
Y = df['Chance of Admit ']
```

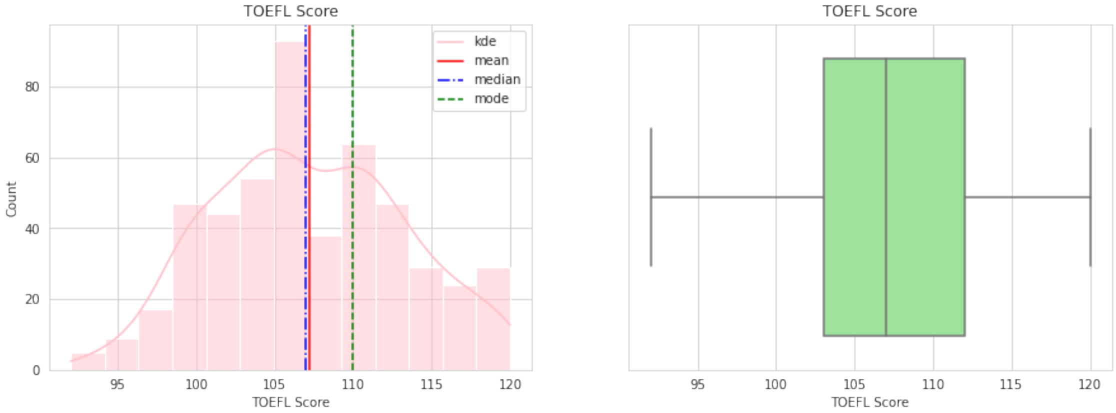
[ ]:

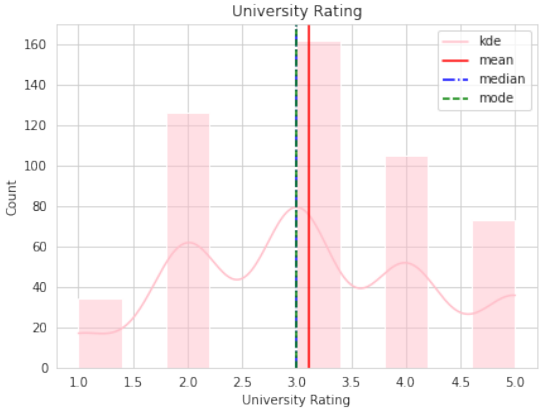
```python
def univariate_plot(x,y):
  fig = plt.figure(figsize=(15,5))
  ax = fig.add_subplot(121)
  sns.histplot(x,kde=True,ax=ax, color='pink')
  ax.axvline(x.mean(), color='red', linestyle='-',linewidth=1.5)
  ax.axvline(x.median(), color='blue', linestyle='-.',linewidth=1.5)
  ax.axvline(x.mode()[0], color='green', linestyle='--',linewidth=1.5)
  ax.legend(labels=['kde','mean','median','mode'])
  ax.set_title(str(y))
  ax2 = fig.add_subplot(122)
  sns.boxplot(x,ax=ax2,color='lightgreen')
  ax2.set_title(str(y))

  plt.show()
```

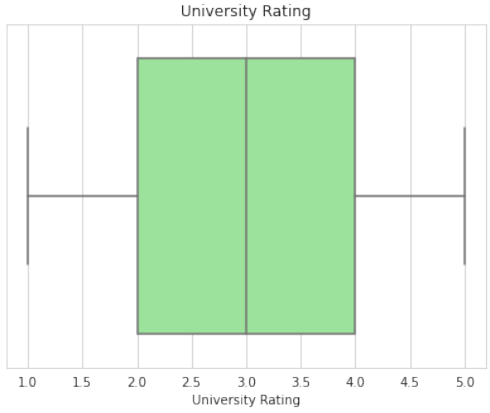
[ ]:

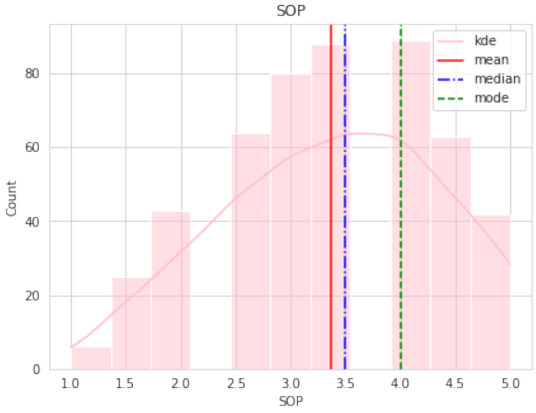
```
for i in list(df.select_dtypes(include=["number"]).columns):
  univariate_plot(df[i],i)
  plt.show()
```

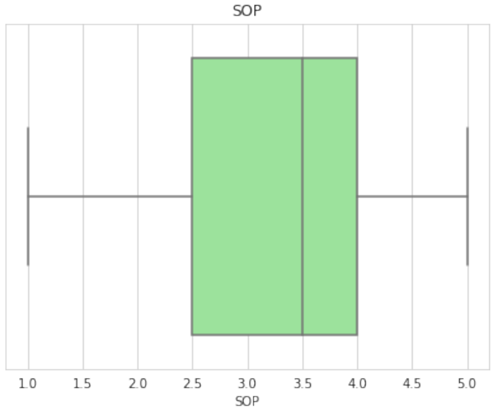


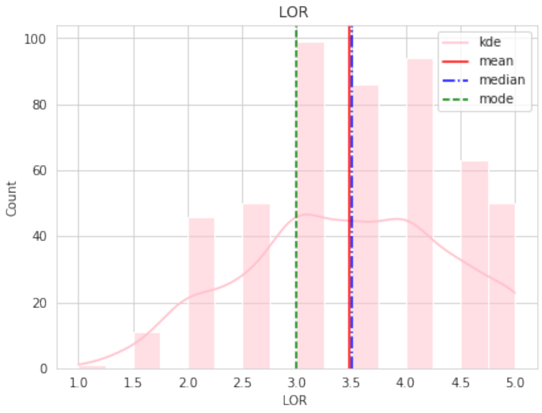


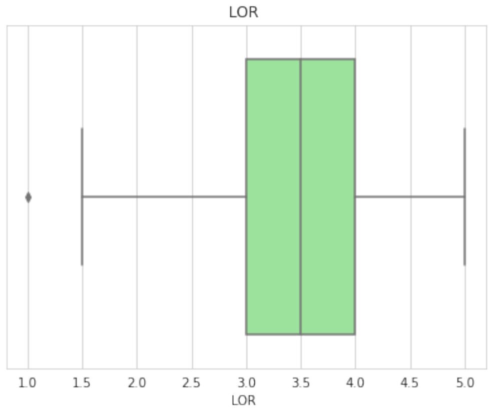

University Rating
kde
mean
median
mode
Count
0
20
40
60
80
100
120
140
160
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
University Rating

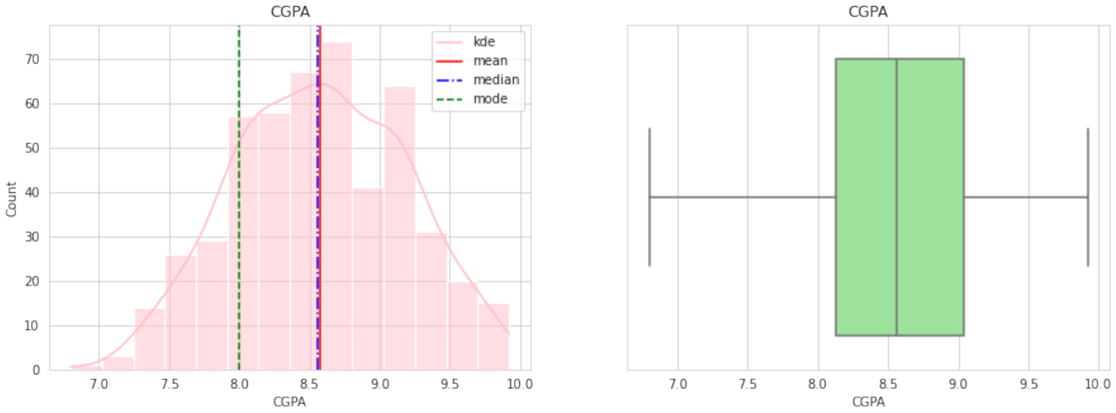

University Rating
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
University Rating

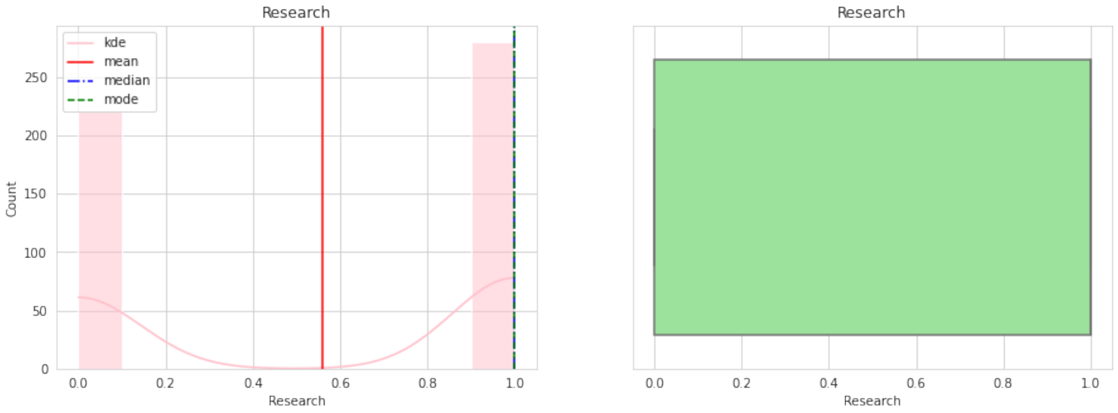

SOP
kde
mean
median
mode
Count
0
20
40
60
80
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
SOP

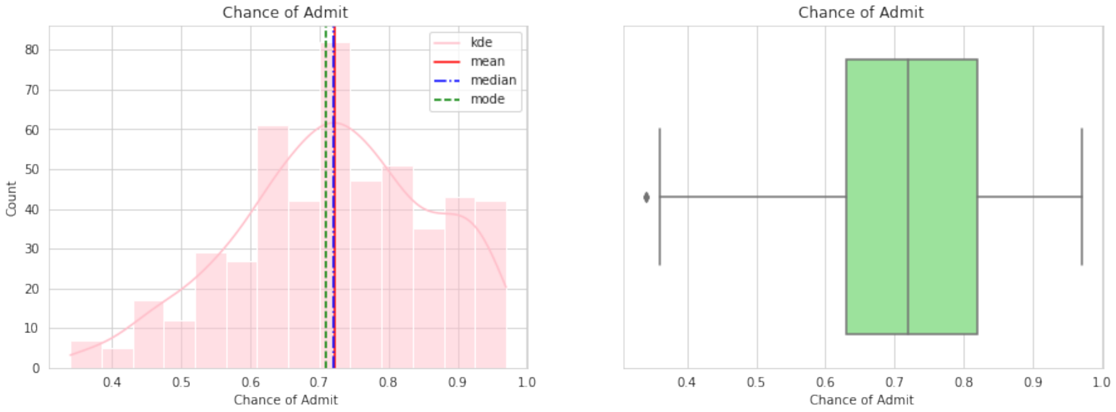

SOP
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
SOP


LOR
kde
mean
median
mode
Count
0
20
40
60
80
100
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
LOR


LOR
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
LOR







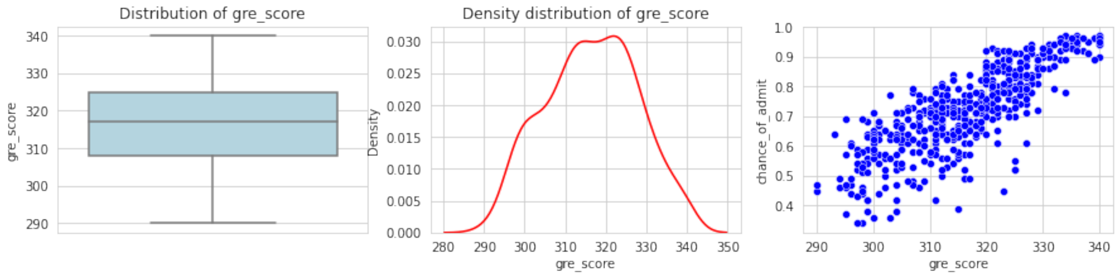
[ ]:
```
df.columns = df.columns.str.replace(' ', '_')
df.columns = df.columns.str.strip('_')
df.columns = df.columns.str.lower()
```

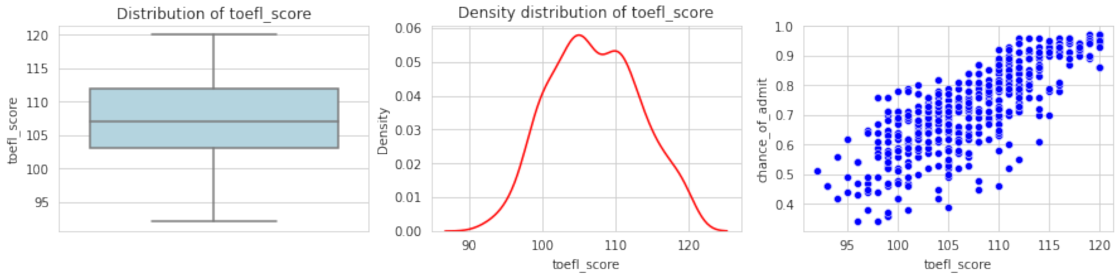
[ ]:

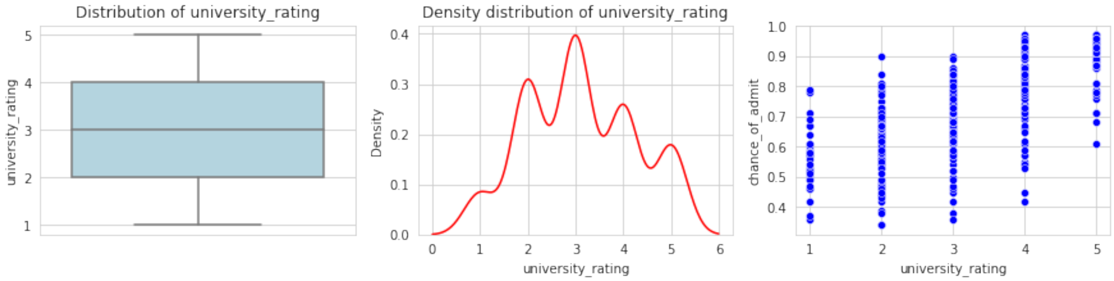
```
columns = [
    'gre_score', 'toefl_score', 'university_rating', 'sop',
    'lor', 'cgpa', 'research', 'chance_of_admit'
]
```

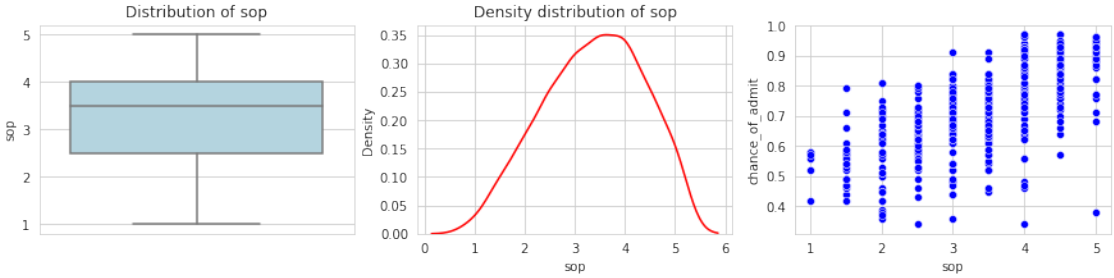
[ ]:

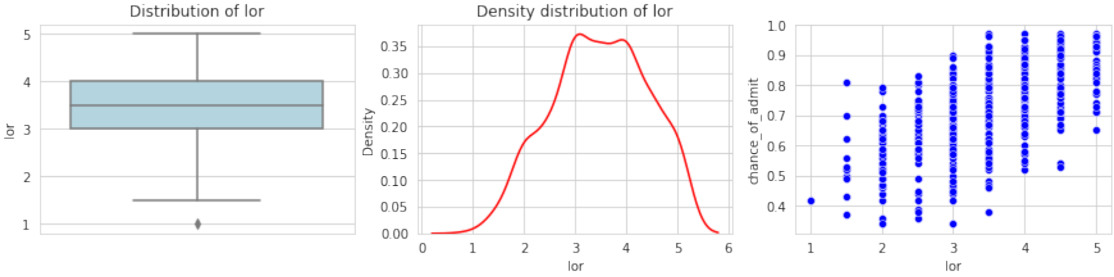
```
for i in columns:
    plt.figure(figsize=(15,3))
    plt.subplot(131)
    sns.boxplot(y=df[i], color='lightblue');
    plt.title(f"Distribution of {i}");

    plt.subplot(132)
    sns.kdeplot(x=df[i],color='red');
    plt.title(f"Density distribution of {i}");

    plt.subplot(133)
    sns.scatterplot(df[i],df['chance_of_admit'],color = 'blue')
    plt.show()
```

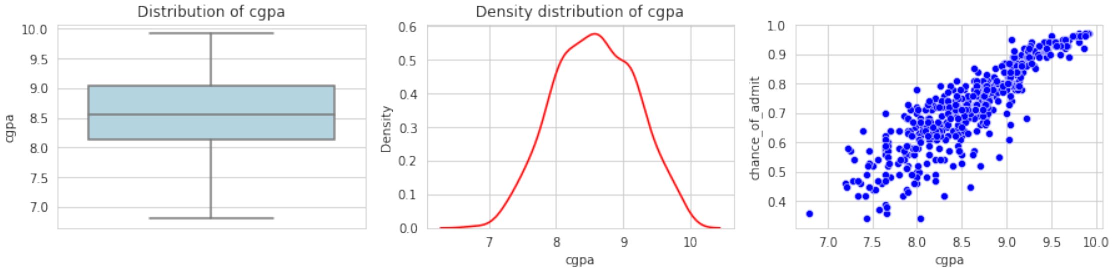


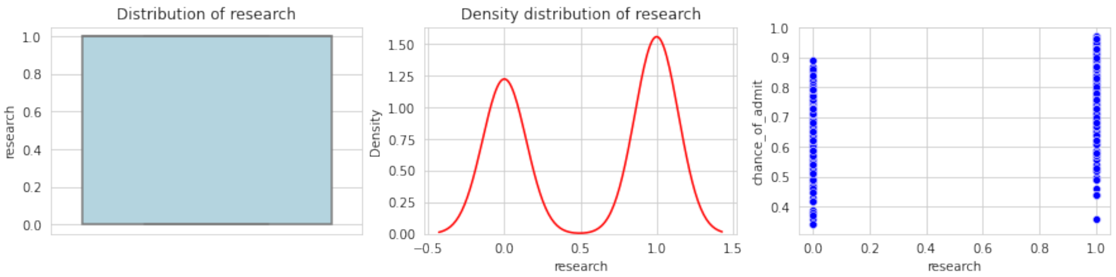


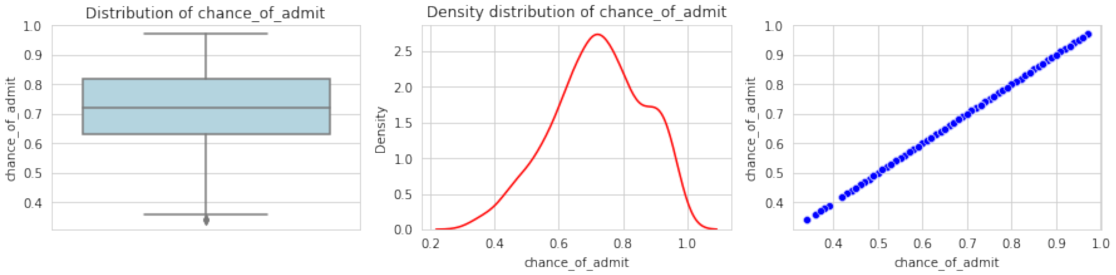

Distribution of university_rating
university_rating
1
2
3
4
5
Density distribution of university_rating
Density
0.0
0.1
0.2
0.3
0.4
0
1
2
3
4
5
6
university_rating
chance_of_admit
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1
2
3
4
5
university_rating


Distribution of sop
sop
1
2
3
4
5
Density distribution of sop
Density
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0
1
2
3
4
5
6
sop
chance_of_admit
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1
2
3
4
5
sop


Distribution of lor
lor
1
2
3
4
5
Density distribution of lor
Density
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0
1
2
3
4
5
6
lor
chance_of_admit
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1
2
3
4
5
lor


Distribution of cgpa
cgpa
7.0
7.5
8.0
8.5
9.0
9.5
10.0
Density distribution of cgpa
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
7
8
9
10
cgpa
chance_of_admit
0.4
0.5
0.6
0.7
0.8
0.9
1.0
7.0
7.5
8.0
8.5
9.0
9.5
10.0
cgpa



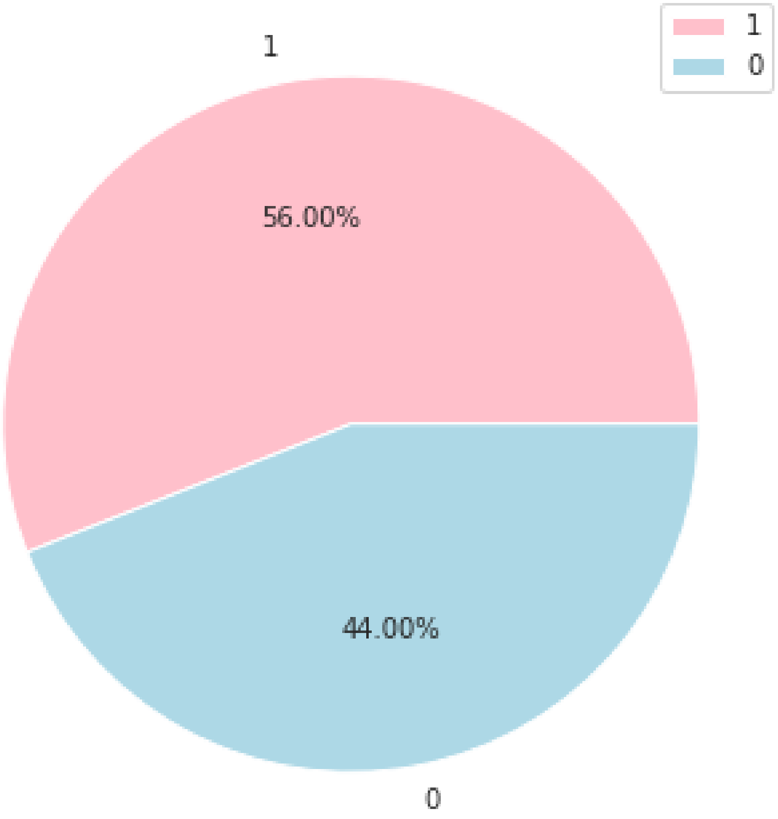


Observations from Boxplot:

- Only **CGPA**, **TOEFL** Score ,**GRE** Score hold the assumption of Linear Regression, linearity of variable.
- Distribution of **LOR** and **Chance of admit** have small range outliers.

[ ]:

```
plt.figure(figsize=(8,6))
plt.pie(
    df['research'].value_counts(dropna=False),
    autopct='%1.2f%%',
    labels=df.research.value_counts().index,
    colors = ("pink", "lightblue")
)
plt.legend()
plt.show()
```

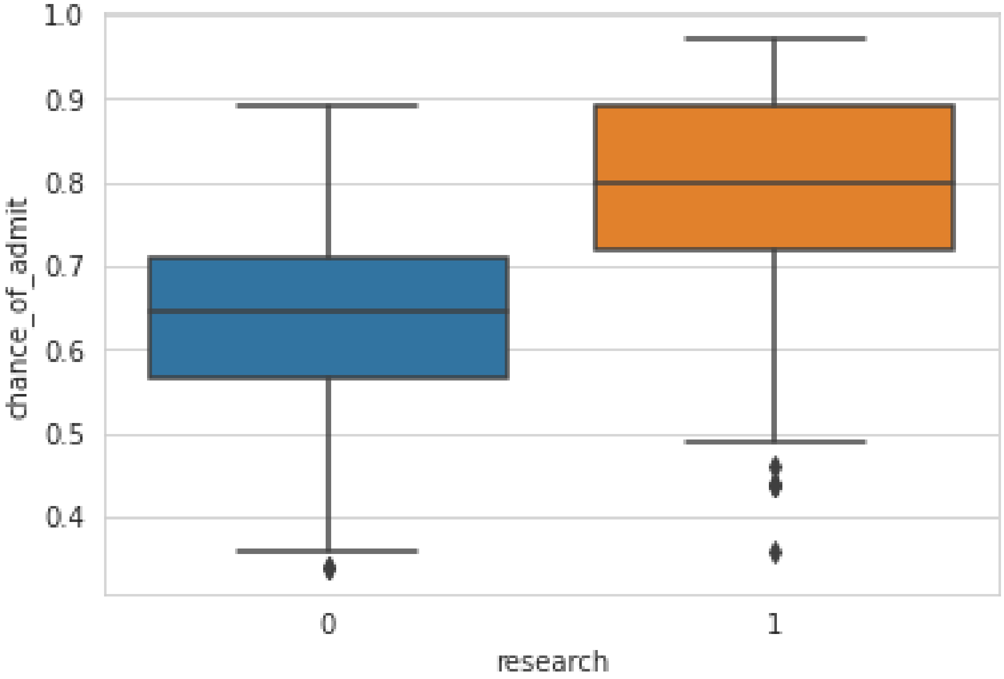


Observation from the pieplot:

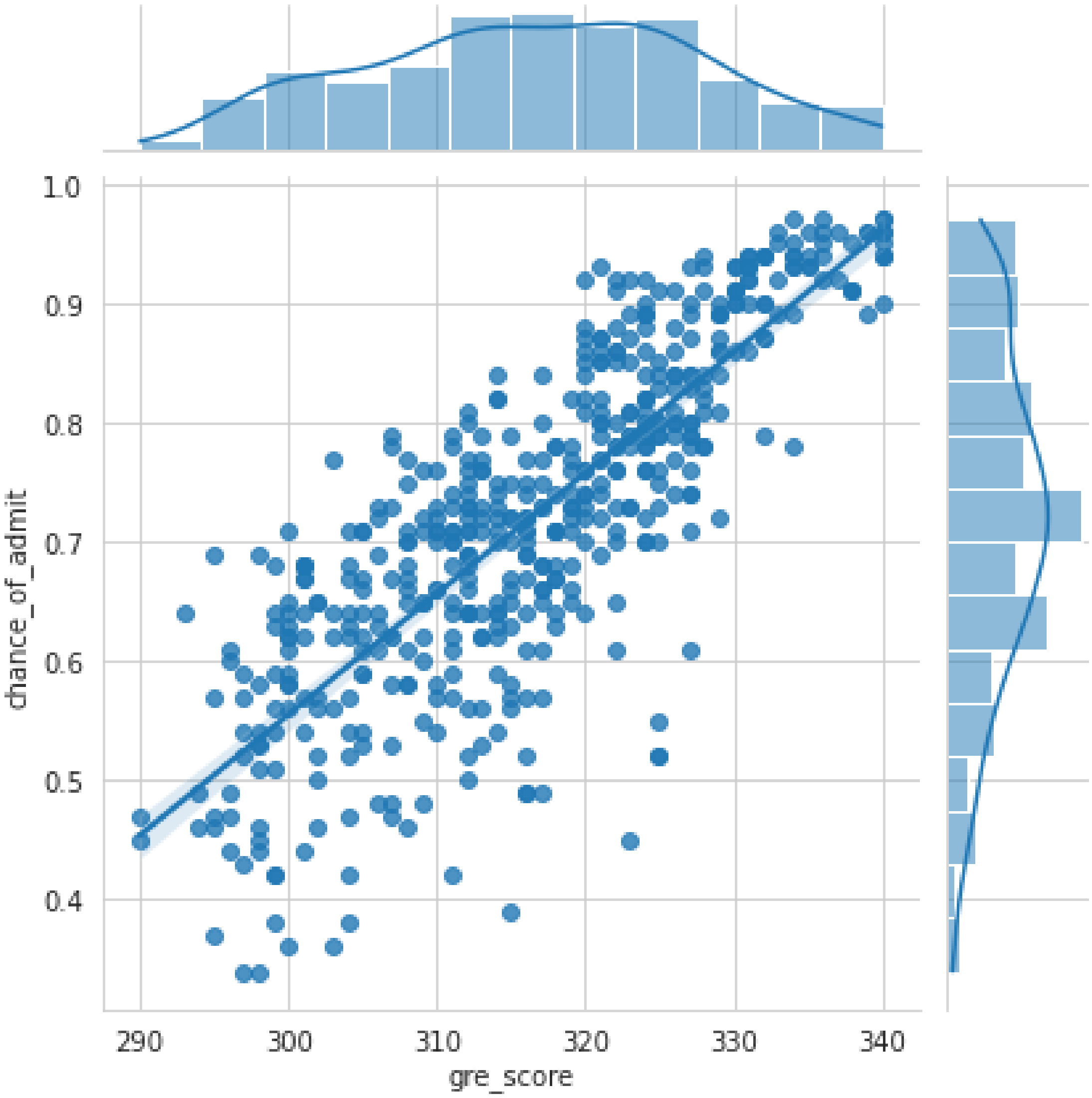
The students with research experience have more chance of getting the admission

0.3 Bivariate Analysis

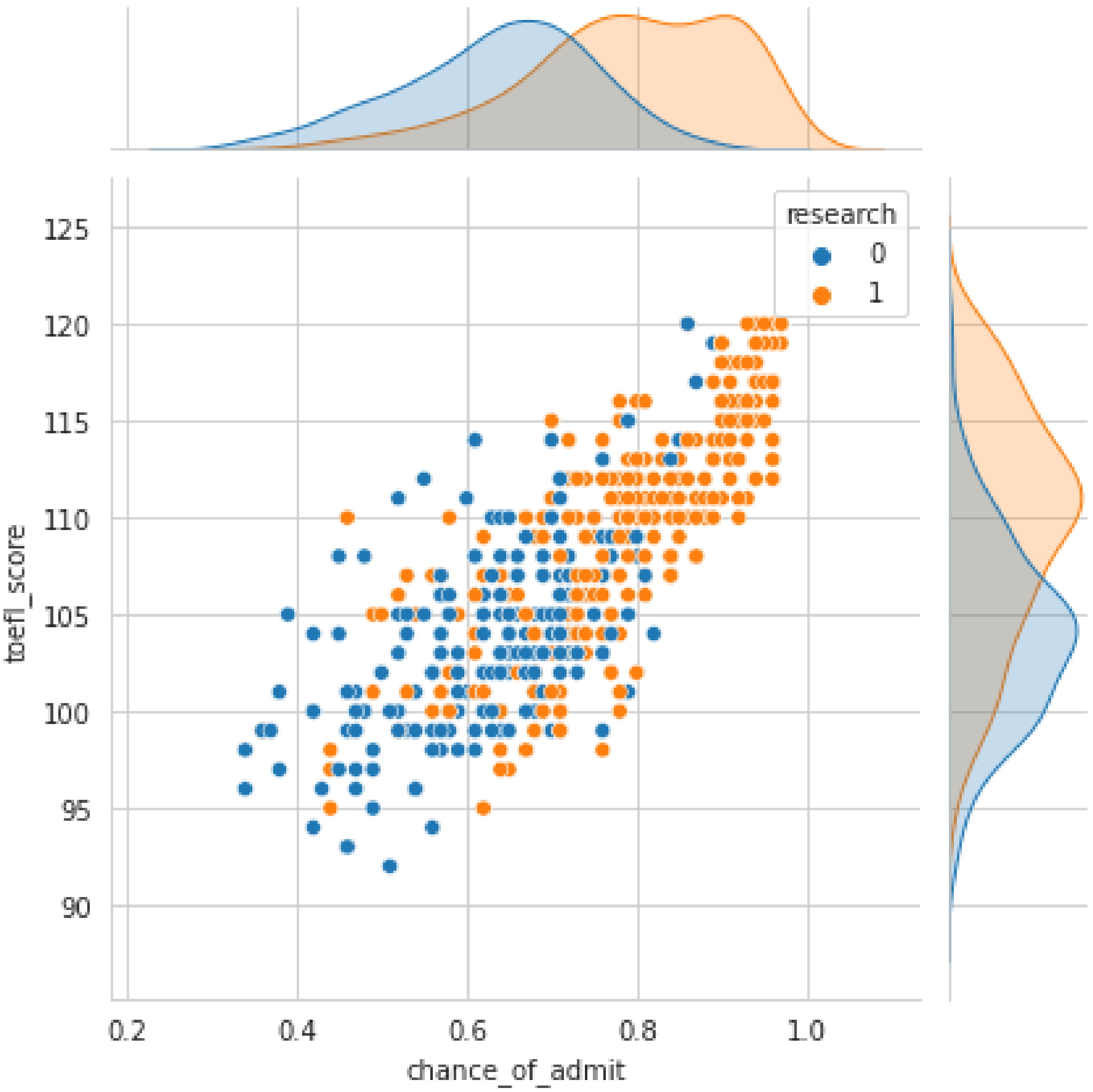
```
[ ]: sns.boxplot(data=df,x='research',y='chance_of_admit')
     plt.show()
```



```
[ ]: sns.jointplot( data=df, kind='reg', x='gre_score',y='chance_of_admit')
     plt.show()
```

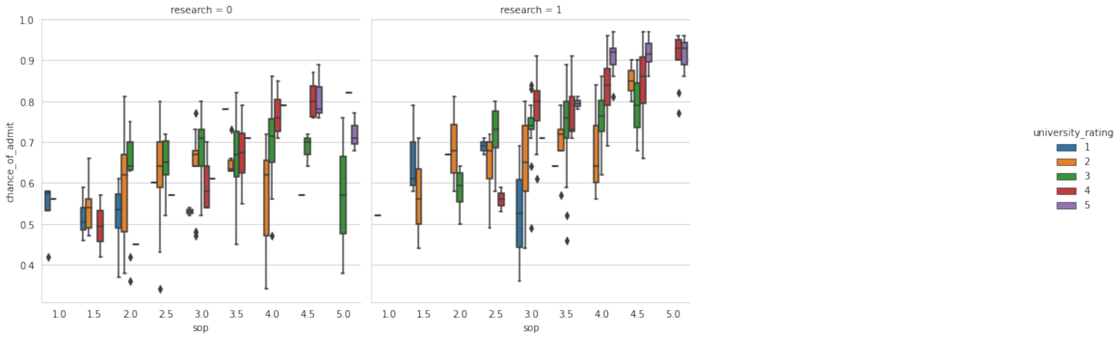


[ ]:
```
sns.jointplot(data=df,x='chance_of_admit',y='toefl_score',kind="scatter",␣
  ↪hue="research")
plt.show()
```



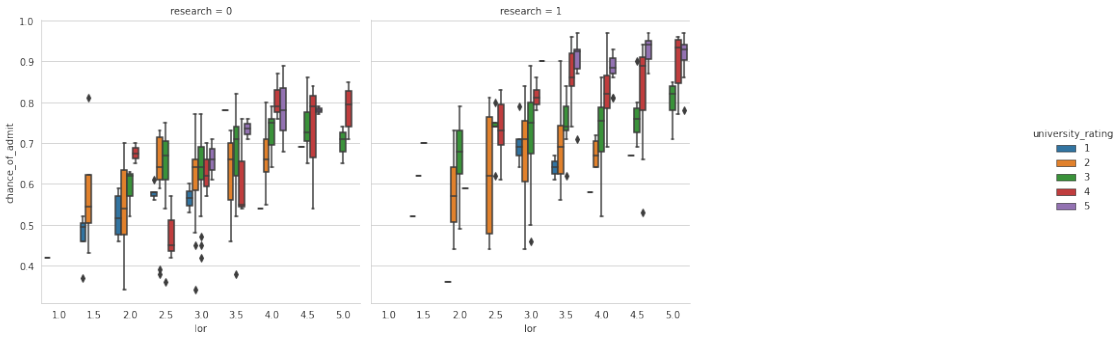
[ ]:

```
sns.catplot(data =␣
 ↪df,y='chance_of_admit',hue='university_rating',x='sop',kind='box',col='research',col_wrap=3
plt.show()
```



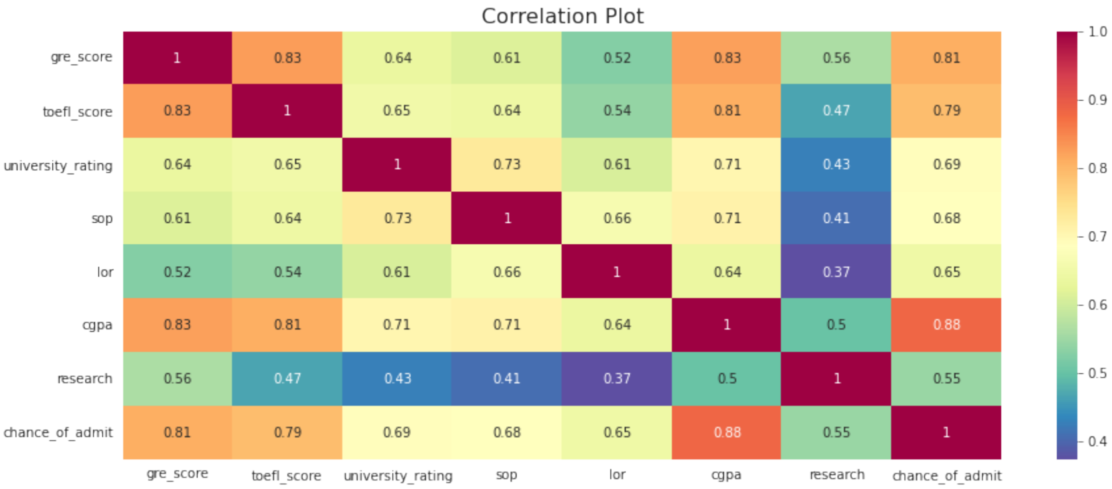
From the above plot, we can clearly see that the students with **SOP=5** and **Research=0** have a very **low** chance of admit.

```
[ ]: sns.catplot(data=df,␣
       ↪y='chance_of_admit',hue='university_rating',x='lor',kind='box',col='research',col_wrap=3)
     plt.show()
```



0.4 Multivariate Analysis

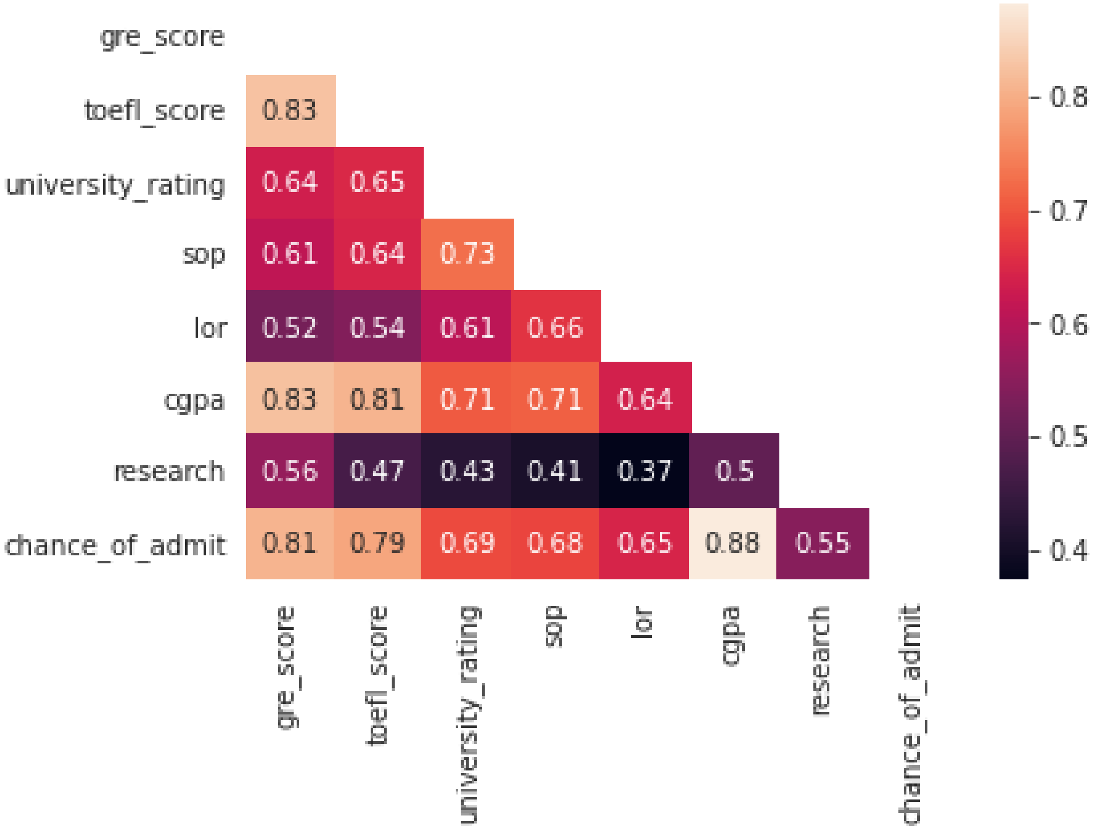
```
[ ]: plt.figure(figsize=(15,6))
     corr = df.corr()
     sns.heatmap(corr,cmap='Spectral_r',annot=True)
     plt.title(
     'Correlation Plot',
     fontsize=16,
     )
     plt.show()
```



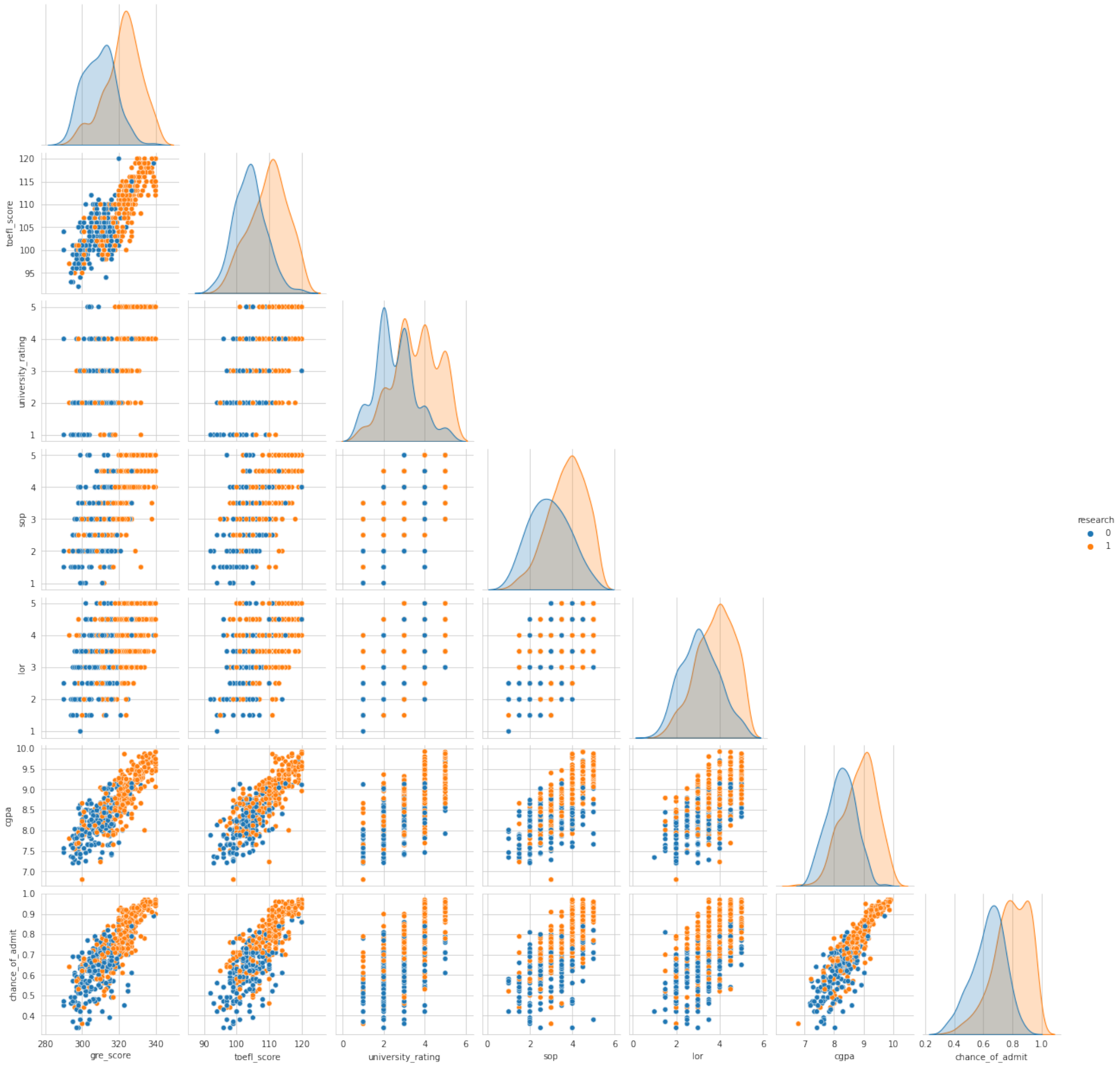
Observations from Correlation Heatmap:

There is a high correlation between: 1. TOEFL score and GRE score. 2. CGPA and GRE score. 3. CGPA and university rating. 4. CGPA and LOR. 5. CGPA and SOP. 6. CGPA and GRE score. 7. TOEFL score and CGPA. 8. Chance of admit and CGPA. 9. Chance of admit and GRE score. 10. Chance of admit and TOEFL score. 11. Chance of admit and TOEFL score.

[ ]:
```
corr = df.corr()
_ = np.triu(corr)
sns.heatmap(data=corr,annot=True,mask=_)
plt.show()
```



[ ]:
```
sns.pairplot(data=df,corner=True,hue='research')
plt.show()
```



0.5 Model building:

[ ]:

```
from sklearn.model_selection import train_test_split
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from sklearn.metrics import mean_absolute_error, mean_squared_error,␣
  ↪mean_absolute_percentage_error, r2_score
```

0.5.1 Outlier Treatment:

[ ]:

```
for col in ['gre_score', 'toefl_score', 'university_rating', 'sop', 'lor', 'cgpa']:
  q3 = df[col].quantile(0.75)
  q1 = df[col].quantile(0.25)
  iqr = q3-q1
  upper_limit = q3 + (1.5*iqr)
  lower_limit = q1 - (1.5*iqr)
  df = df.loc[((df[col]>=lower_limit) & (df[col]<=upper_limit))]
X = df.drop('chance_of_admit',axis=1)
Y = df['chance_of_admit']
# Treat Research as category for Interpreatation and do One-Hot-Encodiing
X = pd.get_dummies(X,drop_first=True)
```

0.5.2 Split Data: Train and Test Data

[ ]:

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
statsmodel_x_train = x_train.copy(deep=True)
statsmodel_x_test = x_test.copy(deep=True)
x_train.shape, x_test.shape
```

[ ]:

```
((399, 7), (100, 7))
```

0.5.3 Standardizing the data:

[ ]:

```
x_train_mean_std = pd.DataFrame(np.mean(x_train)).rename(columns={0:"mean"})
x_train_mean_std['std']= np.std(x_train)
x_train_mean_std
```

[ ]:

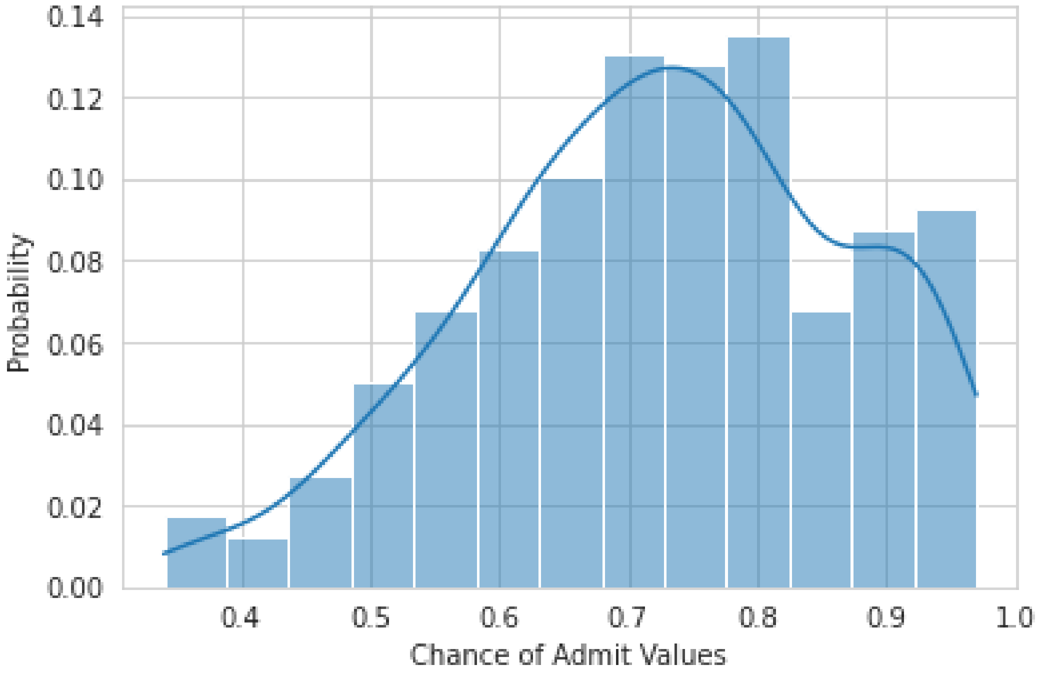
```
                         mean        std
gre_score          316.892231  11.199083
toefl_score        107.363409   6.129003
university_rating    3.117794   1.145399
sop                  3.385965   0.992530
lor                  3.494987   0.923095
cgpa                 8.585890   0.608468
research             0.551378   0.497353
```

[ ]:

```
sc = StandardScaler()
values = sc.fit_transform(x_train)
x_train = pd.DataFrame(data=values,columns=['gre_score', 'toefl_score', 'university_rating', 'sop', 'lor', 'cgpa','research_1']
)
```

[ ]:

```
sns.histplot(data=y_train.values,kde=True,stat='probability')
plt.xlabel("Chance of Admit Values")
plt.show()
```



From the above plot we can say that y_train in not perfect.

0.5.4 Variance_Inflation_Factor(VIF) or checking multicollinearity

[ ]:

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

[ ]:

```
            Features   VIF
5               cgpa  4.80
0          gre_score  4.57
1        toefl_score  3.68
3                sop  2.90
2  university_rating  2.72
4                lor  2.08
6         research_1  1.57
```

As VIF Score is below 5 for every column, we can assume that there is no multicollinearity between multiple columns.

0.5.5 Linear Regression

[ ]:

```
model = LinearRegression()
model.fit(x_train,y_train)
LinearRegression()
# Results from using sklearn package
print("intercept: ", model.intercept_)
print(pd.DataFrame(data=model.
 ↪coef_,index=['gre_score','toefl_score','university_rating','sop','lor','cgpa','research_1']
 ↪rename(columns={0:"coef"},errors='raise'
))
y_train_predict = model.predict(x_train)
print(f"R^2 score on train data: {r2_score(y_train,y_train_predict)}")
```

```
intercept:  0.7236090225563909
                       coef
gre_score          0.025445
toefl_score        0.020628
university_rating  0.002809
sop                0.001135
lor                0.017351
cgpa               0.069341
research_1         0.010076
R^2 score on train data: 0.8268135842932021
```

We got the R2 Score for train data as 0.8268135842932021

[ ]:

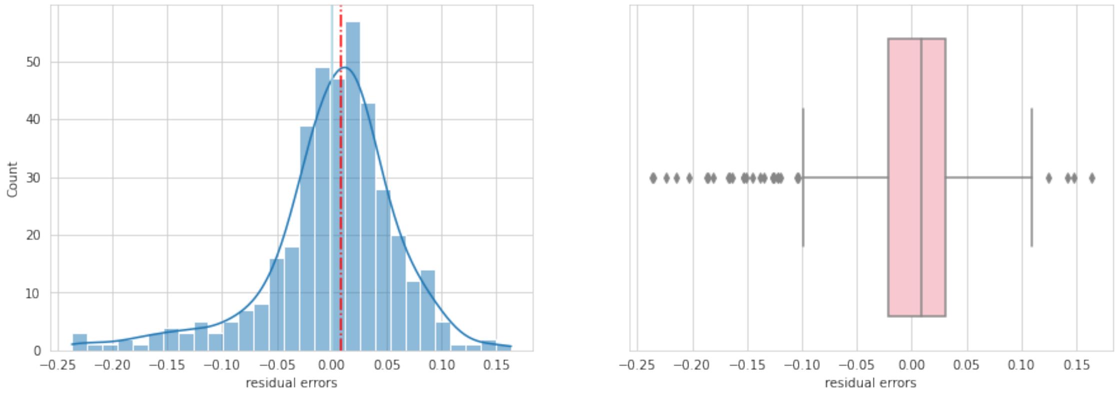
```
residuals_error = y_train-y_train_predict
residuals_error = pd.Series(data=residuals_error,index=y_train.index)
print(f"mean {residuals_error.mean()}")
print(f"median {residuals_error.median()}")
print(f"variance {residuals_error.var()}")
```

```
mean -3.756393692340755e-17
median 0.008724903876585977
variance 0.0035274343747741176
```

[ ]:

```
fig = plt.figure(figsize=(15,5))
ax1 = fig.add_subplot(121)
sns.histplot(residuals_error,kde=True,stat='count',ax=ax1)
ax1.axvline(residuals_error.mean(), color='lightblue',␣
 ↪linestyle='-',linewidth=1.5)
ax1.axvline(residuals_error.median(), color='red', linestyle='-.',linewidth=1.5)
ax1.set_xlabel('residual errors')
ax = fig.add_subplot(122)
```

```
sns.boxplot(x=residuals_error,ax=ax, color = 'pink')
ax.set_xlabel('residual errors')
plt.show()
```



```
[ ]: x_train_dummy = x_train.copy(deep=True)
     x_train_dummy['y_train'] = y_train
     x_train_dummy['y_predict'] = y_train_predict
```

```
[ ]: index_ = x_train_dummy.
      ↪loc[((x_train_dummy['y_train']-x_train_dummy['y_predict']<-0.10)|␣
      ↪(x_train_dummy['y_train']-x_train_dummy['y_predict']>0.10))].index
     X.loc[index_]
```

```
[ ]:      gre_score  toefl_score  university_rating  sop  lor  cgpa  research
     5          330          115                  5  4.5  3.0  9.34         1
     6          321          109                  3  3.0  4.0  8.20         1
     7          308          101                  2  3.0  4.0  7.90         0
     8          302          102                  1  2.0  1.5  8.00         0
     17         319          106                  3  4.0  3.0  8.00         1
     ..         …            …                 …  …  …    …      …
     390        314          102                  2  2.0  2.5  8.24         0
     393        317          104                  2  3.0  3.0  8.76         0
     395        324          110                  3  3.5  3.5  9.04         1
     396        325          107                  3  3.0  3.5  9.11         1
     397        330          116                  4  5.0  4.5  9.45         1

     [195 rows x 7 columns]
```

```
[ ]: for col in x_test.columns:
       x_test[col] = (x_test[col] - x_train_mean_std['mean'][col])/
      ↪x_train_mean_std['std'][col]
     y_test_predict = model.predict(x_test)
     print(f"mean absoulte error: {mean_absolute_error(y_test,y_test_predict)}")
     print(f"mean squared error: {mean_squared_error(y_test,y_test_predict)}")
     print(f"root mean squared error: {np.
      ↪sqrt(mean_squared_error(y_test,y_test_predict))}")
     print(f"mean absoulte percentage error:␣
      ↪{mean_absolute_percentage_error(y_test,y_test_predict)}")
     print(f"r^2 score on Test Data: {r2_score(y_test,y_test_predict)}")
```

```
mean absoulte error: 0.046685249302270186
mean squared error: 0.0037763480500829063
root mean squared error: 0.061451997934020876
mean absoulte percentage error: 0.07139773117809865
r^2 score on Test Data: 0.7829149801828781
```

We got the R2 Score for test data as 0.7829149801828781

Even though we didn't remove features internally, sklearn assign coef_ is close to zero.

0.5.6 Stats Model Package

[ ]:
```
x_sm = sm.add_constant(statsmodel_x_train)
sm_model = sm.OLS(y_train.values,x_sm).fit()
sm_model.summary()
```

[ ]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.827
Model:                            OLS   Adj. R-squared:                  0.824
Method:                 Least Squares   F-statistic:                     266.7
Date:                Sun, 30 Oct 2022   Prob (F-statistic):          1.38e-144
Time:                        17:34:23   Log-Likelihood:                 560.96
No. Observations:                 399   AIC:                            -1106.
Df Residuals:                     391   BIC:                            -1074.
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
const                -1.4246      0.122    -11.695      0.000      -1.664
-1.185
gre_score             0.0023      0.001      3.968      0.000       0.001
0.003
toefl_score           0.0034      0.001      3.584      0.000       0.002
0.005
```

```
university_rating      0.0025      0.004      0.568      0.570      -0.006
0.011
sop                    0.0011      0.005      0.222      0.824      -0.009
0.011
lor                    0.0188      0.005      4.008      0.000       0.010
0.028
cgpa                   0.1140      0.011     10.554      0.000       0.093
0.135
research               0.0203      0.008      2.682      0.008       0.005
0.035
==============================================================================
Omnibus:                       87.719   Durbin-Watson:                   2.094
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              198.165
Skew:                          -1.117   Prob(JB):                     9.31e-44
Kurtosis:                       5.632   Cond. No.                     1.36e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.36e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

In P>|t|,

H0: coef is zero(Null hypothesis)

Ha: coef is not zero(alternate hypothesis)

all we get P-value of SOP is less than 0.824, fail to reject the null hypothesis

Drop SOP

[ ]:
```
statsmodel_x_train.drop(['sop','university_rating'],axis=1,inplace=True)
```

[ ]:
```
x_sm_1 = sm.add_constant(statsmodel_x_train)
sm_model_1 = sm.OLS(y_train.values,x_sm_1).fit()
sm_model_1.summary()
```

[ ]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.827
Model:                            OLS   Adj. R-squared:                  0.824
Method:                 Least Squares   F-statistic:                     374.6
Date:                Sun, 30 Oct 2022   Prob (F-statistic):          4.78e-147
Time:                        17:34:23   Log-Likelihood:                 560.69
```

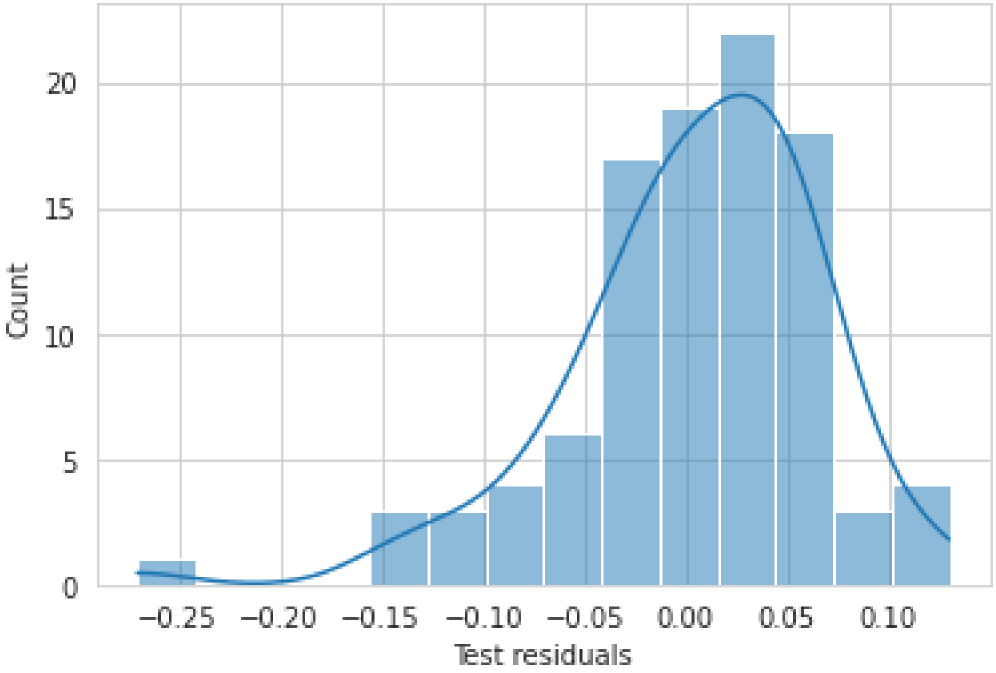
```
No. Observations:                 399   AIC:                            -1109.
Df Residuals:                     393   BIC:                            -1085.
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          -1.4563      0.113    -12.839      0.000      -1.679      -1.233
gre_score       0.0023      0.001      4.059      0.000       0.001       0.003
toefl_score     0.0035      0.001      3.752      0.000       0.002       0.005
lor             0.0200      0.004      4.718      0.000       0.012       0.028
cgpa            0.1159      0.010     11.162      0.000       0.095       0.136
research        0.0207      0.008      2.762      0.006       0.006       0.035
==============================================================================
Omnibus:                       86.572   Durbin-Watson:                   2.092
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              194.520
Skew:                          -1.105   Prob(JB):                     5.76e-43
Kurtosis:                       5.611   Cond. No.                     1.27e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.27e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

Even after removing two features we got the similar R2 score. As per occam's razor principle we can go with 5 features insteadof 7

```
[ ]: statsmodel_x_test.drop(['sop','university_rating'],axis=1,inplace=True)
     statsmodel_x_test = sm.add_constant(statsmodel_x_test)
     y_test_predict = sm_model_1.predict(statsmodel_x_test)
     print(f"mean absoulte error: {mean_absolute_error(y_test,y_test_predict)}")
     print(f"mean squared error: {mean_squared_error(y_test,y_test_predict)}")
     print(f"root mean squared error: {np.
       ↪sqrt(mean_squared_error(y_test,y_test_predict))}")
     print(f"mean absoulte percentage error:␣
       ↪{mean_absolute_percentage_error(y_test,y_test_predict)}")
     print(f"r^2 score: {r2_score(y_test,y_test_predict)}")
```

```
mean absoulte error: 0.047159021440168127
mean squared error: 0.003846388492181463
root mean squared error: 0.062019259042506006
mean absoulte percentage error: 0.07187868127107999
r^2 score: 0.7788886747260408
```

[ ]:
```
sns.histplot(data=(y_test-y_test_predict),kde=True,stat='count')
plt.xlabel('Test residuals')
plt.show()
```



The residuals follows the normal distributions here.

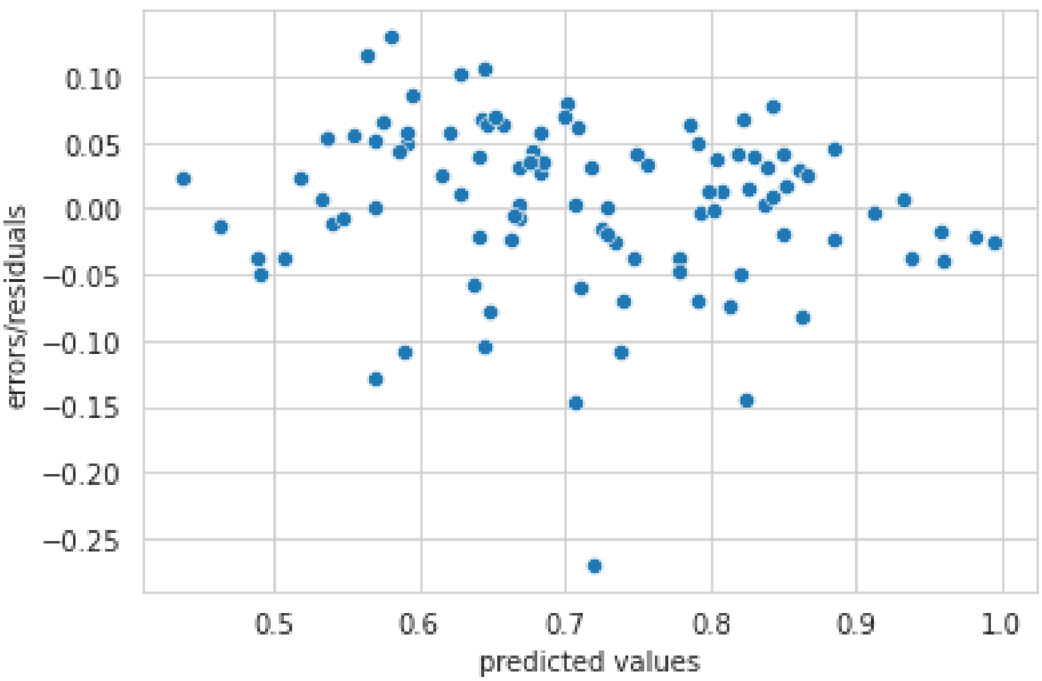
Now we can see that:

Train Data model performs R2 score = 0.827

On Test Data model performs R2 score = 0.78

0.5.7 Check for Homoscedasticity

[ ]:
```
sns.set_style('whitegrid')
sns.scatterplot(x=y_test_predict,y=(y_test-y_test_predict))
plt.xlabel('predicted values')
plt.ylabel('errors/residuals')
plt.show()
```



From the above plot it is clear that there is no pattern in variance, so it is homoscedasticity

0.6 Recommendations:

1. The chance of getting admission in top rated universities mostly depends on CGPA, as verified by correlation heatmap and other techniques, thus focusing on CGPA will highly increase the chance of getting into top rated universities.
2. After CGPA , students with high TOEFL Score have the high chance of getting into high rated universities, thus focusing on TOEFL Score might increase chance of getting into high rated universities.
3. After CGPA and high TOEFL Score , students with high Gre Score have high probability of getting into high rated universities, thus focusing on Gre Score might increase a little chance of getting into high rated universities.
4. All the research oriented activities are mostly performed in high rated universities. Research experience will also add to the chance of getting admission into the top rated universities, it is verified by pie-plot.