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

```
!gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/839/original/Jamboree_Admission.csv
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/839/original/Jamboree_Admission.csv
To: /content/Jamboree_Admission.csv
100% 16.2k/16.2k [00:00<00:00, 27.5MB/s]
```

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

	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

```
# Serial No. (Unique row ID)
# GRE Scores (out of 340)
# TOEFL Scores (out of 120)
# University Rating (out of 5)
# Statement of Purpose and Letter of Recommendation Strength (out of 5)
# Undergraduate GPA (out of 10)
# Research Experience (either 0 or 1)
# Chance of Admit (ranging from 0 to 1)
```

1. Define Problem Statement and perform Exploratory Data Analysis

Problem Statment: To predict the chances of graduate admission based on the given features.

Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required) , missing value detection, statistical summary

```
df.shape
```

```
(500, 9)
```

```
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
```

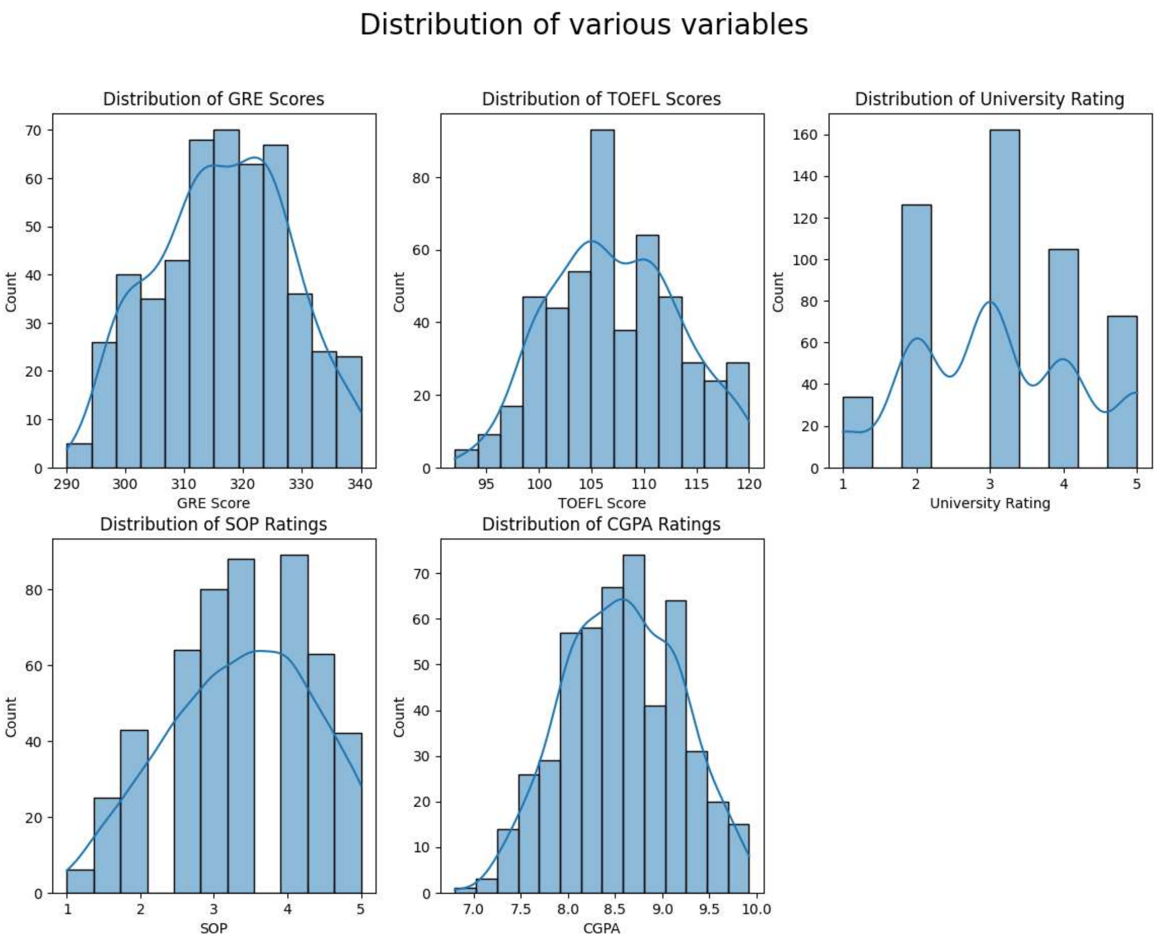
```
df.describe(include='all')
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Cl of /
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.(
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.7
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.1
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.3
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.6
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.7
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.8
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.9

Univariate Analysis

```
plt.figure(figsize=(14,10)).suptitle("Distribution of various variables",fontsize=20)

plt.subplot(2,3, 1)
fig=sns.histplot(df['GRE Score'],kde=True)
plt.title("Distribution of GRE Scores")


plt.subplot(2,3, 2)
fig=sns.histplot(df['TOEFL Score'],kde=True)
plt.title("Distribution of TOEFL Scores")
```

```
plt.subplot(2,3, 3)
fig=sns.histplot(df['University Rating'],kde=True)
plt.title("Distribution of University Rating")

plt.subplot(2,3,4)
fig=sns.histplot(df['SOP'],kde=True)
plt.title("Distribution of SOP Ratings")

plt.subplot(2,3, 5)
fig=sns.histplot(df['CGPA'],kde=True)
plt.title("Distribution of CGPA Ratings")


plt.show()
```

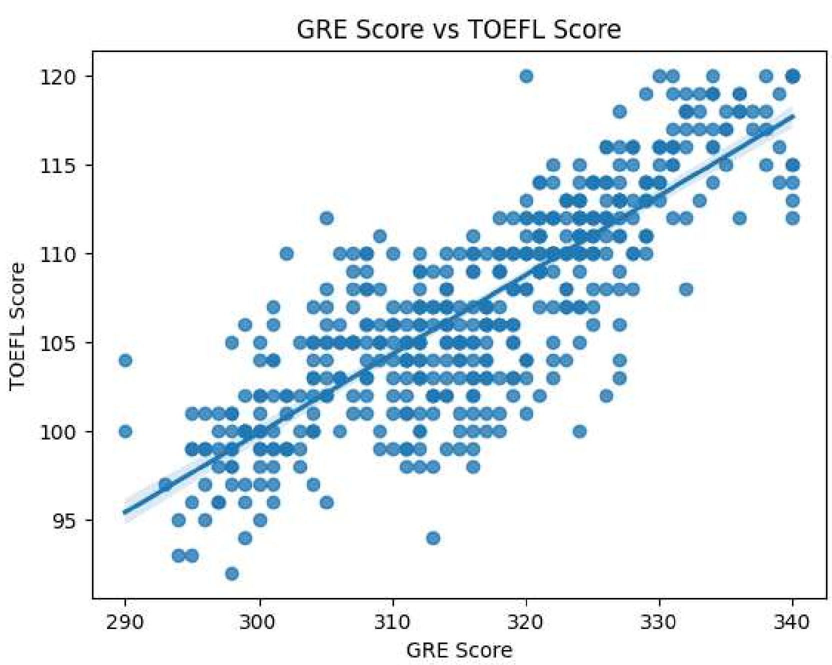


It is clear from the distributions, students with varied merit apply for the university.

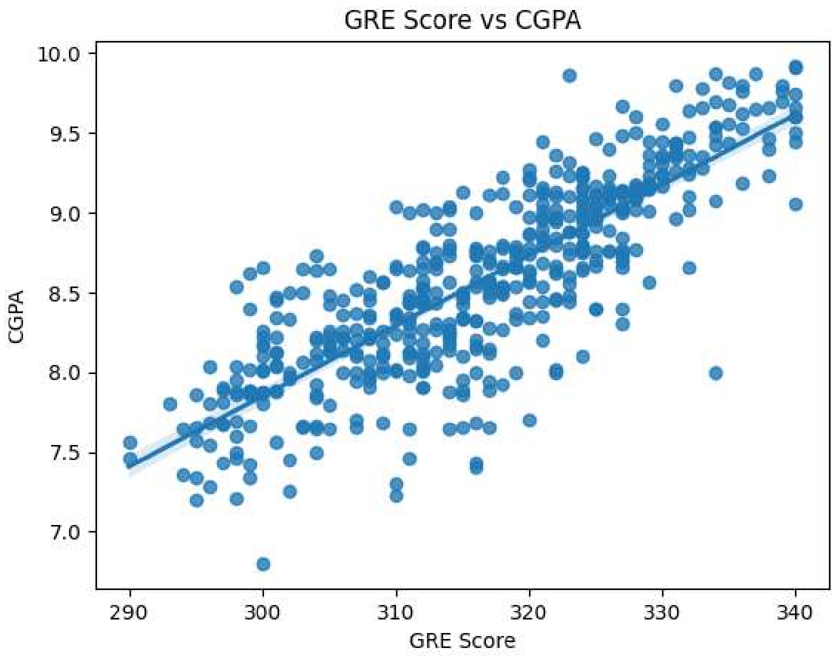
Bivariate Analysis

Relation between different factors responsible for graduate admissions

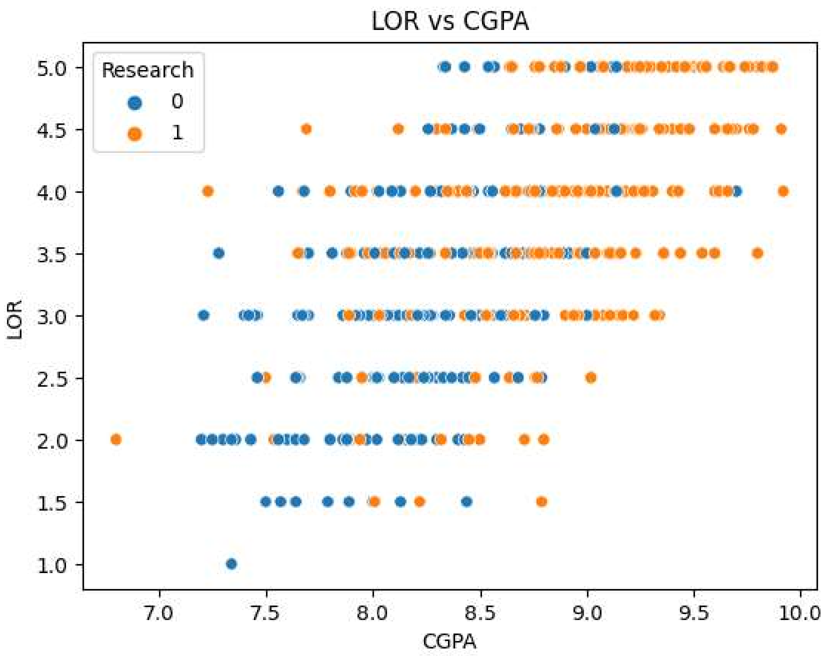
```
fig = sns.regplot(x="GRE Score",y="TOEFL Score",data=df)
plt.title("GRE Score vs TOEFL Score")
plt.show()
```



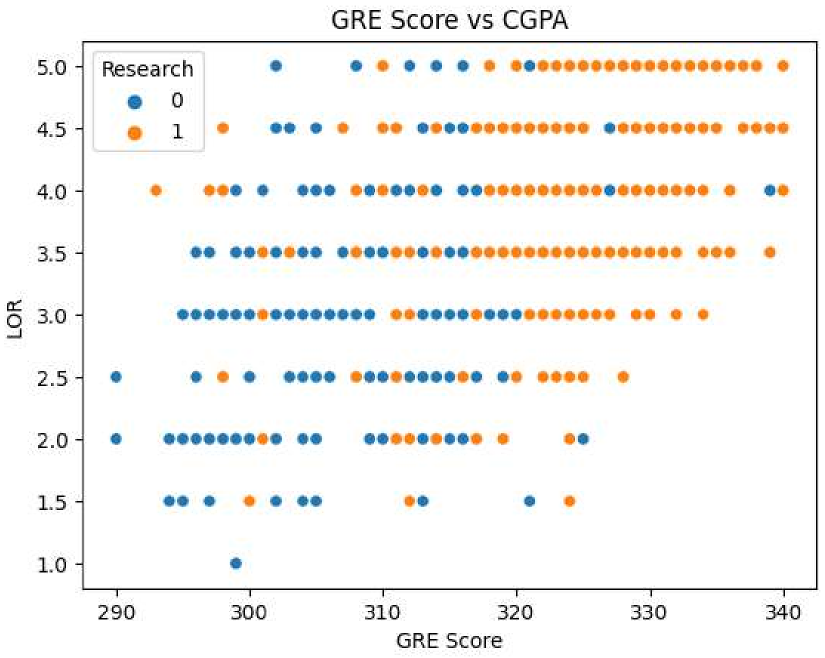

```
fig = sns.regplot(x="GRE Score",y="CGPA",data=df)
plt.title("GRE Score vs CGPA")
plt.show()
```



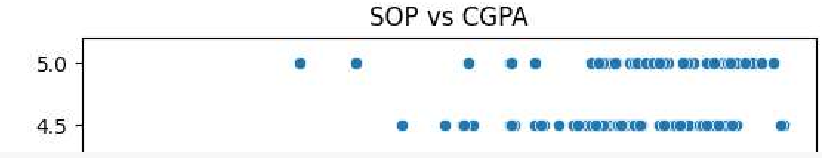

```
fig = sns.scatterplot(x="CGPA", y="LOR ", data=df, hue="Research")
plt.title("LOR vs CGPA")
plt.show()
```



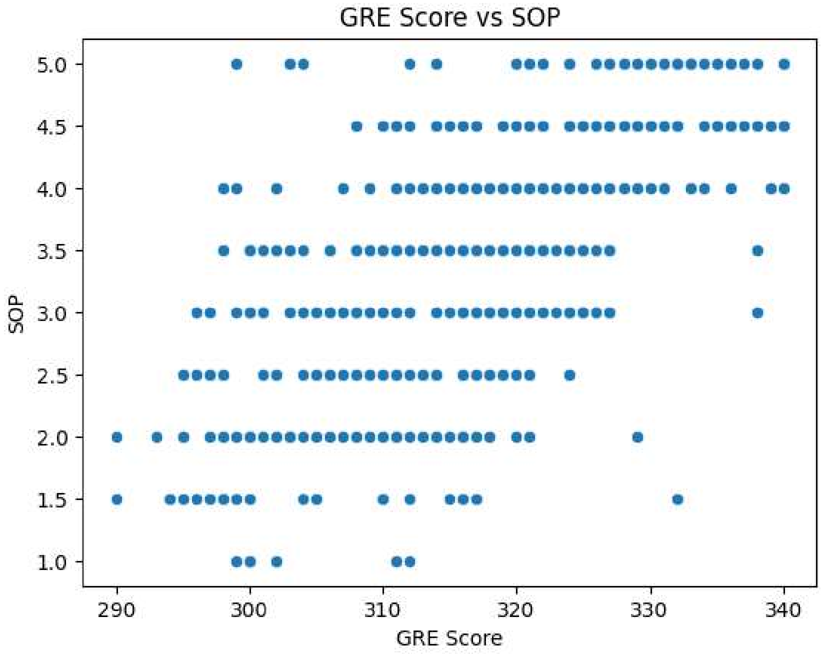

```
fig = sns.scatterplot(x="GRE Score", y="LOR ", data=df, hue="Research")
plt.title("GRE Score vs CGPA")
plt.show()
```



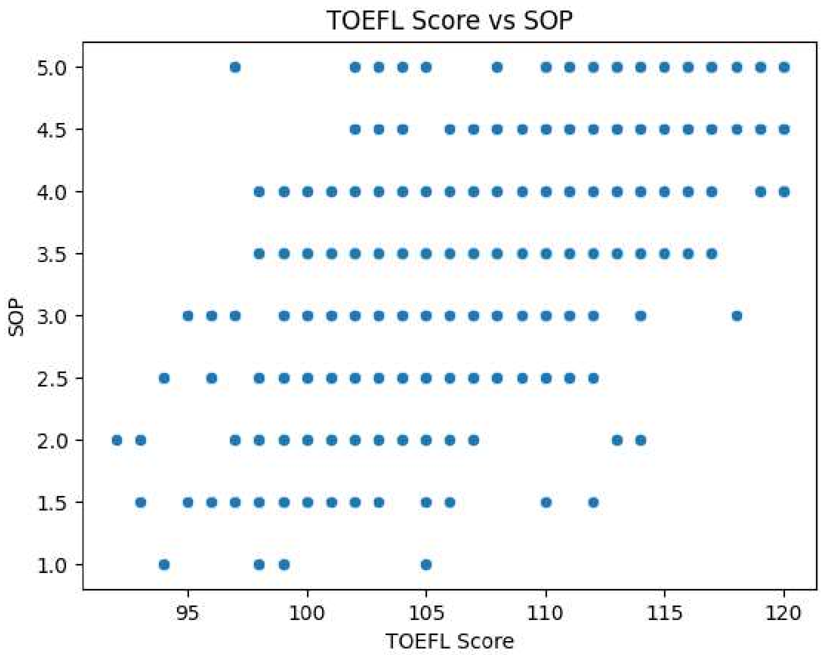

```
fig = sns.scatterplot(x="CGPA", y="SOP", data=df)
plt.title("SOP vs CGPA")
plt.show()
```



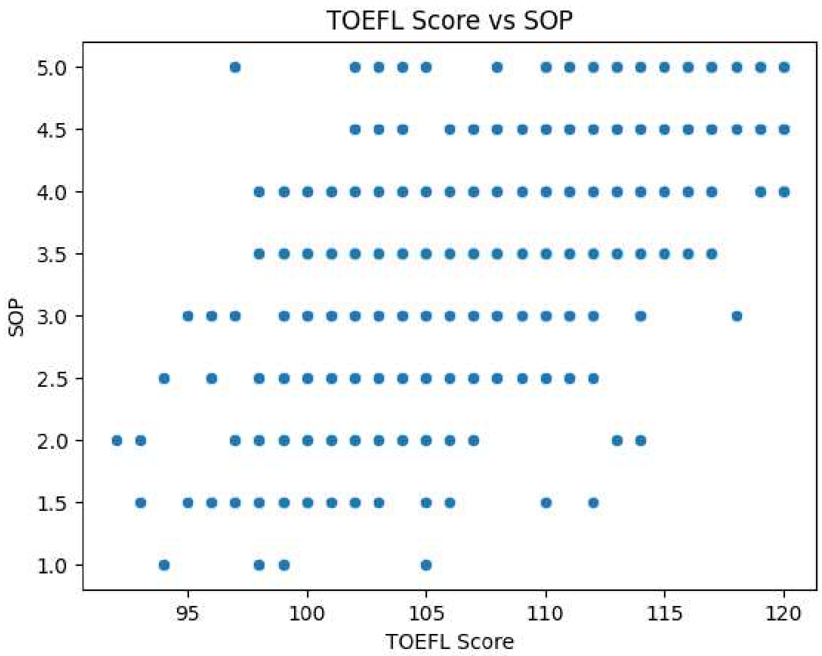

```
fig = sns.scatterplot(x="GRE Score", y="SOP", data=df)
plt.title("GRE Score vs SOP")
plt.show()
```




```
fig = sns.scatterplot(x="TOEFL Score", y="SOP", data=df)
plt.title("TOEFL Score vs SOP")
plt.show()
```

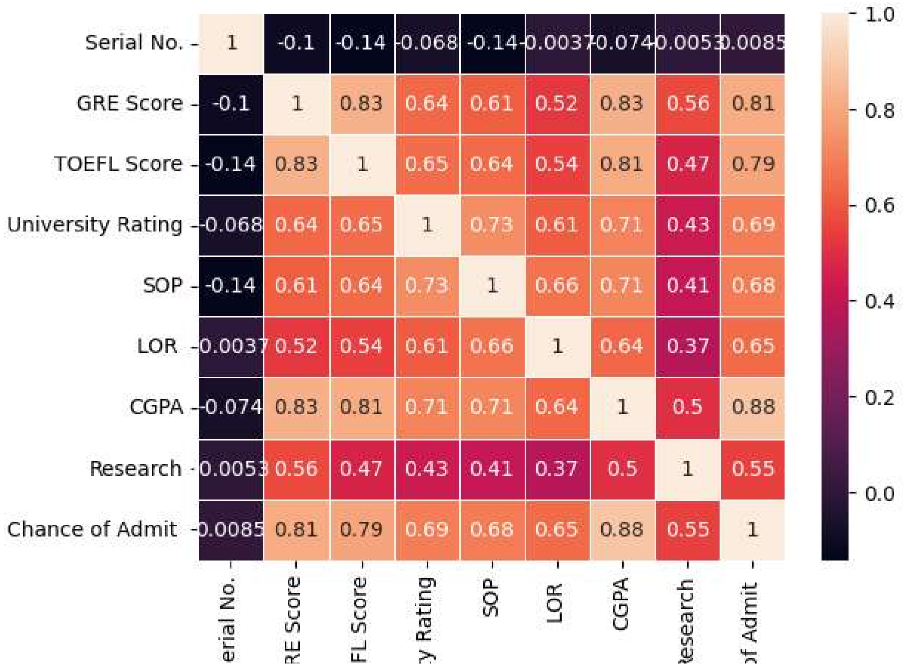



```
fig = sns.scatterplot(x="TOEFL Score", y="SOP", data=df)
plt.title("TOEFL Score vs SOP")
plt.show()
```

Correlation among variables

```
corr = df.corr()
sns.heatmap(corr, linewidths=.5, annot=True)
plt.show()
```



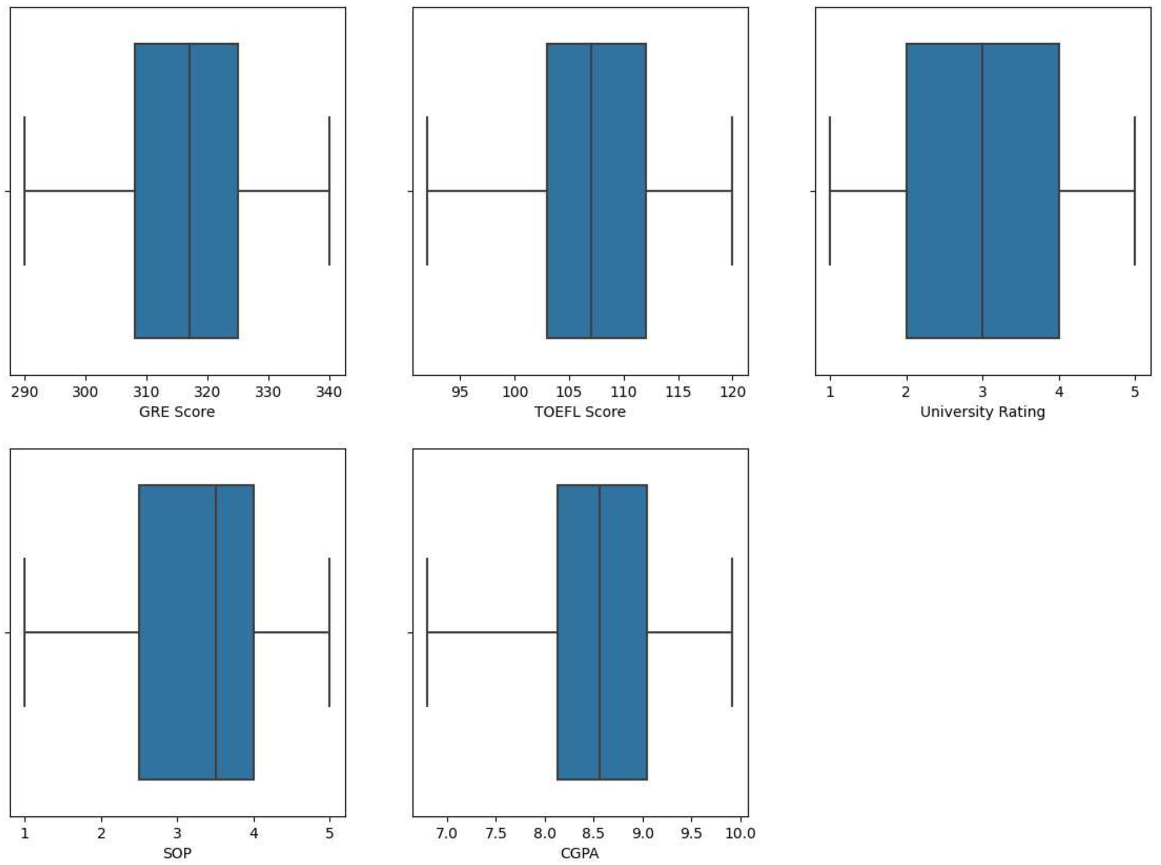
Data Preprocessing

```
# drop Serial NO. column
df = df.drop(columns=['Serial No.'], axis=1)
```

```
# check for duplicates
df.duplicated().sum()
```

```
0
```

```
plt.figure(figsize=(14,10))

plt.subplot(2,3, 1)
fig=sns.boxplot(x=df['GRE Score'],data=df)

plt.subplot(2,3, 2)
fig=sns.boxplot(x=df['TOEFL Score'],data=df)


plt.subplot(2,3, 3)
fig=sns.boxplot(x=df['University Rating'],data=df)

plt.subplot(2,3,4)
fig=sns.boxplot(x=df['SOP'],data=df)

plt.subplot(2,3, 5)
fig=sns.boxplot(x=df['CGPA'],data=df)
```



There are no outliers present in the dataset

Data preparation for model building by splitting the dataset with training and testing set

```
from sklearn.model_selection import train_test_split

X = df.drop(['Chance of Admit '], axis=1)
y = df['Chance of Admit ']
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,shuffle=True)
X_train
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
215	330	116	5	5.0	4.5	9.36	1
298	330	114	3	4.5	4.5	9.24	1
15	314	105	3	3.5	2.5	8.30	0
338	323	108	5	4.0	4.0	8.74	1
4	314	103	2	2.0	3.0	8.21	0
...	...	...	...	...	...	...	...
493	300	95	2	3.0	1.5	8.22	1
487	327	115	4	3.5	4.0	9.14	0
276	329	113	5	5.0	4.5	9.45	1
475	300	101	3	3.5	2.5	7.88	0
483	304	103	5	5.0	3.0	7.92	0

400 rows × 7 columns

```
y_train
```

```
215    0.93
298    0.90
15     0.54
338    0.81
4      0.65
       ...
493    0.62
487    0.79
276    0.89
475    0.59
483    0.71
Name: Chance of Admit , Length: 400, dtype: float64
```

Standarization

```
from sklearn.preprocessing import StandardScaler
X_train_columns=X_train.columns
std=StandardScaler()
X_train_std=std.fit_transform(X_train)
X_train_std
```

```
array([[ 1.21640273,  1.46429159,  1.69299887, ...,  1.11424987,
         1.32711786,  0.89091075],
       [ 1.21640273,  1.13597509, -0.0591345 , ...,  1.11424987,
         1.1271001 ,  0.89091075],
       [-0.19494199, -0.34144916, -0.0591345 , ..., -1.0208397 ,
        -0.43970572, -1.12244688],
       ...,
       [ 1.12819369,  0.97181684,  1.69299887, ...,  1.11424987,
         1.47713118,  0.89091075],
       [-1.42986862, -0.99808216, -0.0591345 , ..., -1.0208397 ,
        -1.13976789, -1.12244688],
       [-1.07703244, -0.66976566,  1.69299887, ..., -0.48706731,
        -1.0730953 , -1.12244688]])
```

```
X_train=pd.DataFrame(X_train_std,columns=X_train_columns)
X_train
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	1.216403	1.464292	1.692999	1.696882	1.114250	1.327118	0.890911
1	1.216403	1.135975	-0.059135	1.190350	1.114250	1.127100	0.890911
2	-0.194942	-0.341449	-0.059135	0.177286	-1.020840	-0.439706	-1.122447
3	0.598939	0.151026	1.692999	0.683818	0.580477	0.293693	0.890911
4	-0.194942	-0.669766	-0.935201	-1.342310	-0.487067	-0.589719	-1.122447
...	...	...	...	...	...	...	...
395	-1.429869	-1.983032	-0.935201	-0.329246	-2.088384	-0.573051	0.890911
396	0.951776	1.300133	0.816932	0.177286	0.580477	0.960419	-1.122447
397	1.128194	0.971817	1.692999	1.696882	1.114250	1.477131	0.890911
398	-1.429869	-0.998082	-0.059135	0.177286	-1.020840	-1.139768	-1.122447
399	-1.077032	-0.669766	1.692999	1.696882	-0.487067	-1.073095	-1.122447

400 rows × 7 columns

Model building

```
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso,Ridge,LinearRegression
from sklearn.metrics import mean_squared_error
```

```
models = [
           ['Linear Regression :', LinearRegression()],

          ['Lasso Regression :', Lasso(alpha=0.1)], #try with different alpha values
          ['Ridge Regression :', Ridge(alpha=1.0)] #try with different alpha values
          ]

print("Results without removing features with multicollinearity ...")


for name,model in models:
    model.fit(X_train, y_train.values)
    predictions = model.predict(std.transform(X_test))
    print(name, (np.sqrt(mean_squared_error(y_test, predictions))))
```

```
Results without removing features with multicollinearity ...
Linear Regression : 0.054912287744188125
Lasso Regression : 0.12771769035305203
Ridge Regression : 0.054969829195955215
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Lasso was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Ridge was fitted with feature names
  warnings.warn(
```

Linear Regression using Statsmodel library

```
import statsmodels.api as sm
X_train=sm.add_constant(X_train)
model=sm.OLS(y_train.values,X_train).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.809
Model:                            OLS   Adj. R-squared:                  0.805
Method:                 Least Squares   F-statistic:                     236.9
Date:                Mon, 18 Sep 2023   Prob (F-statistic):          1.42e-136
Time:                        17:25:55   Log-Likelihood:                 551.91
No. Observations:                 400   AIC:                            -1088.
Df Residuals:                     392   BIC:                            -1056.
Df Model:                           7
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                 0.7203      0.003    234.211      0.000       0.714       0.726
GRE Score             0.0229      0.006      3.608      0.000       0.010       0.035
TOEFL Score           0.0201      0.006      3.387      0.001       0.008       0.032
University Rating     0.0116      0.005      2.318      0.021       0.002       0.021
SOP                  -0.0017      0.005     -0.332      0.740      -0.012       0.009
LOR                   0.0155      0.004      3.525      0.000       0.007       0.024
CGPA                  0.0649      0.007      9.862      0.000       0.052       0.078
Research              0.0096      0.004      2.533      0.012       0.002       0.017
==============================================================================
Omnibus:                       97.584   Durbin-Watson:                   1.942
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              232.662
Skew:                          -1.213   Prob(JB):                     3.01e-51
Kurtosis:                       5.841   Cond. No.                         5.54
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```
X_train_new=X_train.drop(columns='SOP')
```

```
model1=sm.OLS(y_train.values,X_train_new).fit()
print(model1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.809
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     277.0
Date:                Mon, 18 Sep 2023   Prob (F-statistic):          8.64e-138
Time:                        17:25:55   Log-Likelihood:                 551.85
No. Observations:                 400   AIC:                            -1090.
Df Residuals:                     393   BIC:                            -1062.
Df Model:                           6
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                 0.7203      0.003    234.476      0.000       0.714       0.726
GRE Score             0.0230      0.006      3.631      0.000       0.011       0.036
TOEFL Score           0.0200      0.006      3.375      0.001       0.008       0.032
University Rating     0.0110      0.005      2.370      0.018       0.002       0.020
LOR                   0.0150      0.004      3.624      0.000       0.007       0.023
CGPA                  0.0645      0.006     10.002      0.000       0.052       0.077
Research              0.0095      0.004      2.525      0.012       0.002       0.017
==============================================================================
Omnibus:                       98.918   Durbin-Watson:                   1.940
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              237.520
Skew:                          -1.227   Prob(JB):                     2.65e-52
Kurtosis:                       5.869   Cond. No.                         5.10
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

VIF(Variance Inflation Factor)

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

def calculate_vif(dataset,col):
    dataset=dataset.drop(columns=col,axis=1)
```

```
    vif=pd.DataFrame()
    vif['features']=dataset.columns
    vif['VIF_Value']=[variance_inflation_factor(dataset.values,i) for i in range(dataset.shape[1])]
    return vif
```

```
calculate_vif(X_train_new,[])
```

	features	VIF_Value
0	const	1.000000
1	GRE Score	4.266587
2	TOEFL Score	3.715528
3	University Rating	2.275559
4	LOR	1.826483
5	CGPA	4.403837
6	Research	1.501664

VIF looks fine and hence, we can go ahead with the predictions

```
X_test_std=std.transform(X_test)
```

```
X_test=pd.DataFrame(X_test_std,columns=X_train_columns)
```

```
X_test=sm.add_constant(X_test)
```

```
X_test_del=list(set(X_test.columns).difference(set(X_train_new.columns)))
```

```
print(f'Dropping {X_test_del} from test set')
```

```
Dropping ['SOP'] from test set
```

```
X_test_new=X_test.drop(columns=X_test_del)
```

```
#Prediction from the clean model
pred=model1.predict(X_test_new)

from sklearn.metrics import mean_squared_error,r2_score,mean_absolute_error

print('Mean Absolute Error ',mean_absolute_error(y_test.values,pred))
print('Root Mean Square Error ',np.sqrt(mean_squared_error(y_test.values,pred)))
```

```
Mean Absolute Error  0.03793499840457928
Root Mean Square Error  0.05475155790111222
```

Mean of Residuals

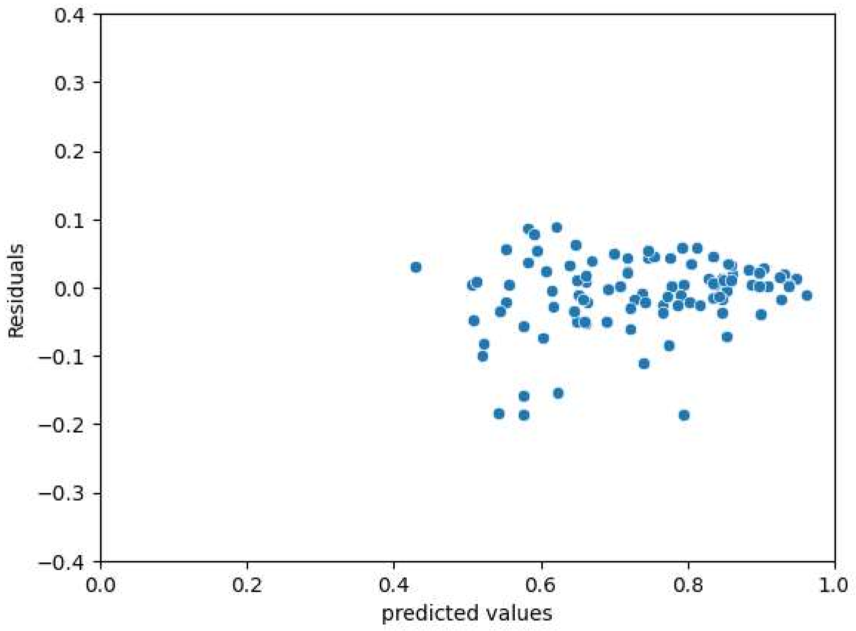
```
residuals = y_test.values-pred
mean_residuals = np.mean(residuals)
print("Mean of Residuals {}".format(mean_residuals))
```

```
Mean of Residuals -0.008670993045012405
```

Test for Hommoscedasticity

```
import seaborn as sns
p = sns.scatterplot(x=pred,y=residuals)
plt.xlabel('predicted values')
plt.ylabel('Residuals')
plt.ylim(-0.4,0.4)
plt.xlim(0,1)
#p = sns.lineplot([0,26],[0,0],color='blue')
#p = plt.title('Residuals vs fitted values plot for homoscedasticity check')
```

```
(0.0, 1.0)
```



Here null hypothesis is - error terms are homoscedastic and since p-values >0.05, we fail to reject the null hypothesis

```
import statsmodels.stats.api as sms
```

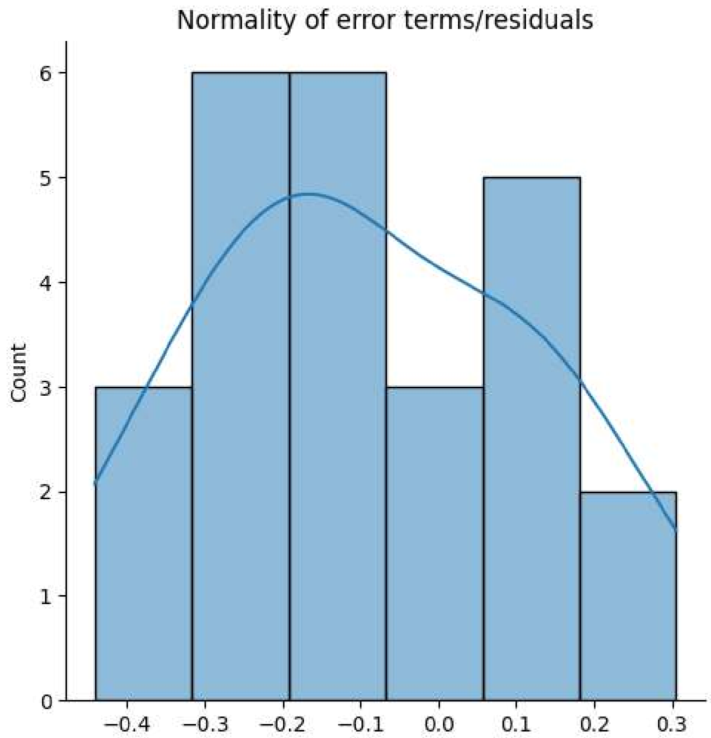
```
import statsmodels.stats.api as sms
from statsmodels.compat import lzip
name=['F statistics','p-value']
test=sms.het_goldfeldquandt(residuals,X_test)
lzip(name,test)
```

```
[('F statistics', 1.1483370469532888), ('p-value', 0.3280314712521504)]
```

Normality of residuals

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.displot(x=residuals,kde=True)
plt.title('Normality of error terms/residuals')
```

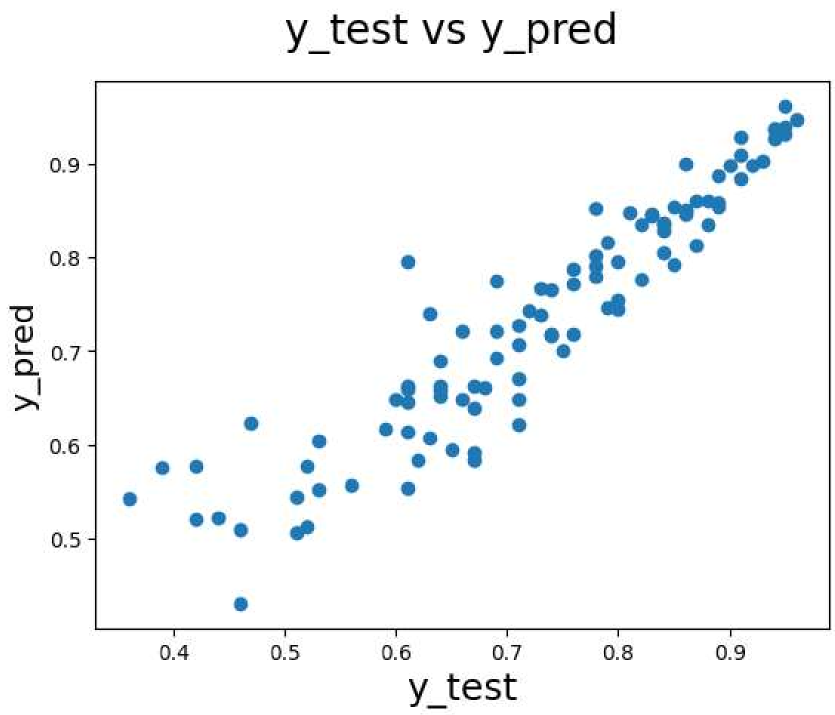
```
Text(0.5, 1.0, 'Normality of error terms/residuals')
```



Model performance evaluation

```
# Plotting y_test and y_pred to understand the spread.
fig = plt.figure()
plt.scatter(y_test.values, pred)
fig.suptitle('y_test vs y_pred', fontsize=20)              # Plot heading
plt.xlabel('y_test', fontsize=18)                          # X-label
plt.ylabel('y_pred', fontsize=16)
```

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



Insights

1. Multicollinearity is present in the data.
2. After removing collinear features there are only two variables which are important in making predictions for the target variables.
3. Indepedent variables are linearly correlated with dependent variables.

Recommendations

1. CGPA and Research are the only two variables which are important in making the prediction for Chance of Admit.
2. CGPA is the most important varibale in making the prediction for the Chance of Admit.
3. Following are the final model results on the test data:
   - RMSE: 0.07
   - MAE: 0.05

- R2_score: 0.81
- Adjusted_R2: 0.81