```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso, 
ElasticNet
from sklearn.metrics import r2_score, mean_absolute_error, 
mean_squared_error
from scipy import stats
from statsmodels.stats.outliers_influence import 
variance_inflation_factor
import statsmodels.api as sm
import statsmodels.stats.api as sms

jb=pd.read_csv("C:/Users/shrad/Downloads/Jamboree_Admission.csv")
jb.head(10)
```

```
   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   
5           6        330          115                  5  4.5   3.0  
9.34   
6           7        321          109                  3  3.0   4.0  
8.20   
7           8        308          101                  2  3.0   4.0  
7.90   
8           9        302          102                  1  2.0   1.5  
8.00   
9          10        323          108                  3  3.5   3.0  
8.60   

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

```
4         0              0.65
5         1              0.90
6         1              0.75
7         0              0.68
8         0              0.50
9         0              0.45

jb.drop(columns="Serial No.",inplace=True)
```

Insight We can remove the uneccessary column "Serial No." from data, as it doesn't have any contribution for data visualization and operation.

```
jb.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
```

Insight There are null values in our data.

```
jb.nunique()

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

Insight Features like "Research","University Rating","SOP","LOR" are categorical features.

```
jb.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
```

```
jb.describe()
```

```
        GRE Score  TOEFL Score  University Rating         SOP
LOR   \
count  500.000000   500.000000         500.000000  500.000000
500.00000
mean   316.472000   107.192000           3.114000    3.374000
3.48400
std     11.295148     6.081868           1.143512    0.991004
0.92545
min    290.000000    92.000000           1.000000    1.000000
1.00000
25%    308.000000   103.000000           2.000000    2.500000
3.00000
50%    317.000000   107.000000           3.000000    3.500000
3.50000
75%    325.000000   112.000000           4.000000    4.000000
4.00000
max    340.000000   120.000000           5.000000    5.000000
5.00000

             CGPA    Research  Chance of Admit
count  500.000000  500.000000        500.00000
mean     8.576440    0.560000          0.72174
std      0.604813    0.496884          0.14114
min      6.800000    0.000000          0.34000
25%      8.127500    0.000000          0.63000
50%      8.560000    1.000000          0.72000
75%      9.040000    1.000000          0.82000
max      9.920000    1.000000          0.97000
```

```
mean=jb.mean(axis=0)
median=jb.median(axis=0)
mode=pd.DataFrame(jb.mode(axis=0),columns=jb.columns)
print(mean,'\n----------- mean:-----------\n',median,'\n-----------
mode:-----------\n')
mode.head()
std=jb.describe().loc['std']
Q1=jb.describe().loc['25%']
Q3=jb.describe().loc['75%']
IQR=Q3-Q1
```

```
max=Q1=jb.describe().loc['max']
min=Q1=jb.describe().loc['min']
print(std,'\n----------- IQR:-----------\n',IQR,'\n----------- max:\
n',max,'\n*********** min:*************\n'),min
skew=jb.skew(axis=1)
kurt=pd.DataFrame(jb.kurt(axis=1),columns=["kurt"])
kurt.head()
#print('\n*********** skew:*************\n',skew,'\n***********
kurt:*************\n',kurt)
```

```
GRE Score            316.47200
TOEFL Score          107.19200
University Rating      3.11400
SOP                    3.37400
LOR                    3.48400
CGPA                   8.57644
Research               0.56000
Chance of Admit        0.72174
dtype: float64
----------- mean:-----------
 GRE Score            317.00
TOEFL Score          107.00
University Rating      3.00
SOP                    3.50
LOR                    3.50
CGPA                   8.56
Research               1.00
Chance of Admit        0.72
dtype: float64
----------- mode:-----------

GRE Score            11.295148
TOEFL Score           6.081868
University Rating     1.143512
SOP                   0.991004
LOR                   0.925450
CGPA                  0.604813
Research              0.496884
Chance of Admit       0.141140
Name: std, dtype: float64
----------- IQR:-----------
 GRE Score            17.0000
TOEFL Score           9.0000
University Rating     2.0000
SOP                   1.5000
LOR                   1.0000
CGPA                  0.9125
Research              1.0000
Chance of Admit       0.1900
dtype: float64
```

```
----------- max:
 GRE Score            340.00
TOEFL Score          120.00
University Rating      5.00
SOP                    5.00
LOR                    5.00
CGPA                   9.92
Research               1.00
Chance of Admit        0.97
Name: max, dtype: float64
*********** min:*************


       kurt
0  5.342020
1  5.607376
2  5.608048
3  5.439713
4  5.605129
```

```
jb.cov()
```

```
                    GRE Score  TOEFL Score  University Rating
SOP  \
GRE Score          127.580377    56.825026           8.206605
6.867206
TOEFL Score         56.825026    36.989114           4.519150
3.883960
University Rating    8.206605     4.519150           1.307619
0.825014
SOP                  6.867206     3.883960           0.825014
0.982088
LOR                  5.484521     3.048168           0.644112
0.608701
CGPA                 5.641944     2.981607           0.487761
0.426845
Research             3.162004     1.411303           0.242645
0.200962
Chance of Admit      1.291862     0.680046           0.111384
0.095691

                        LOR      CGPA  Research  Chance of Admit
GRE Score          5.484521  5.641944  3.162004         1.291862
TOEFL Score        3.048168  2.981607  1.411303         0.680046
University Rating  0.644112  0.487761  0.242645         0.111384
SOP                0.608701  0.426845  0.200962         0.095691
LOR                0.856457  0.356807  0.171303         0.084296
CGPA               0.356807  0.365799  0.150655         0.075326
Research           0.171303  0.150655  0.246894         0.038282
Chance of Admit    0.084296  0.075326  0.038282         0.019921
```

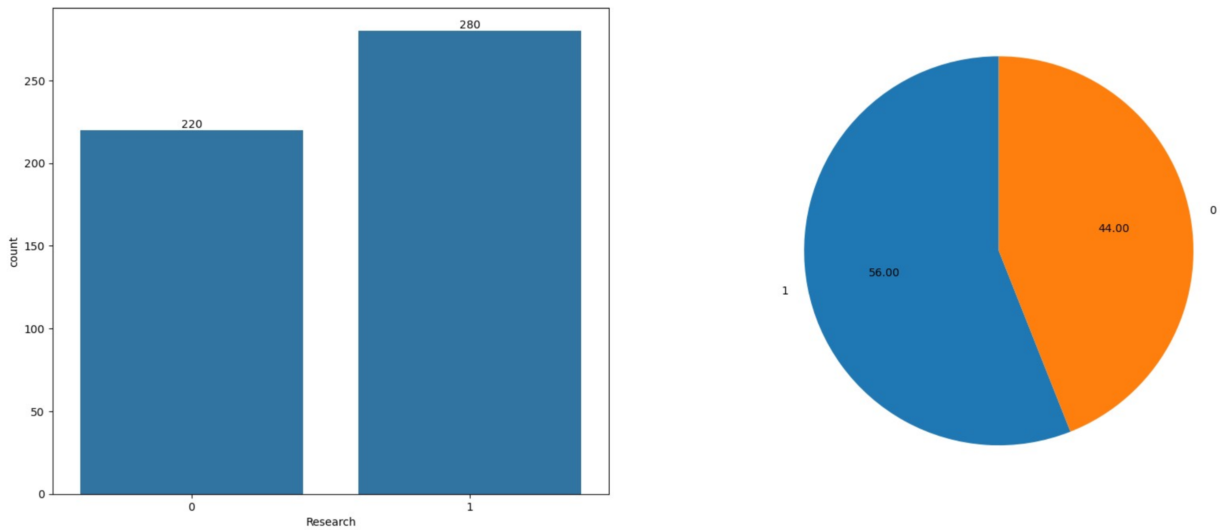
```
jb.corr()
```

	GRE Score	TOEFL Score	University Rating	SOP
GRE Score	1.000000	0.827200	0.635376	0.613498
TOEFL Score	0.827200	1.000000	0.649799	0.644410
University Rating	0.635376	0.649799	1.000000	0.728024
SOP	0.613498	0.644410	0.728024	1.000000
LOR	0.524679	0.541563	0.608651	0.663707
CGPA	0.825878	0.810574	0.705254	0.712154
Research	0.563398	0.467012	0.427047	0.408116
Chance of Admit	0.810351	0.792228	0.690132	0.684137

	LOR	CGPA	Research	Chance of Admit
GRE Score	0.524679	0.825878	0.563398	0.810351
TOEFL Score	0.541563	0.810574	0.467012	0.792228
University Rating	0.608651	0.705254	0.427047	0.690132
SOP	0.663707	0.712154	0.408116	0.684137
LOR	1.000000	0.637469	0.372526	0.645365
CGPA	0.637469	1.000000	0.501311	0.882413
Research	0.372526	0.501311	1.000000	0.545871
Chance of Admit	0.645365	0.882413	0.545871	1.000000

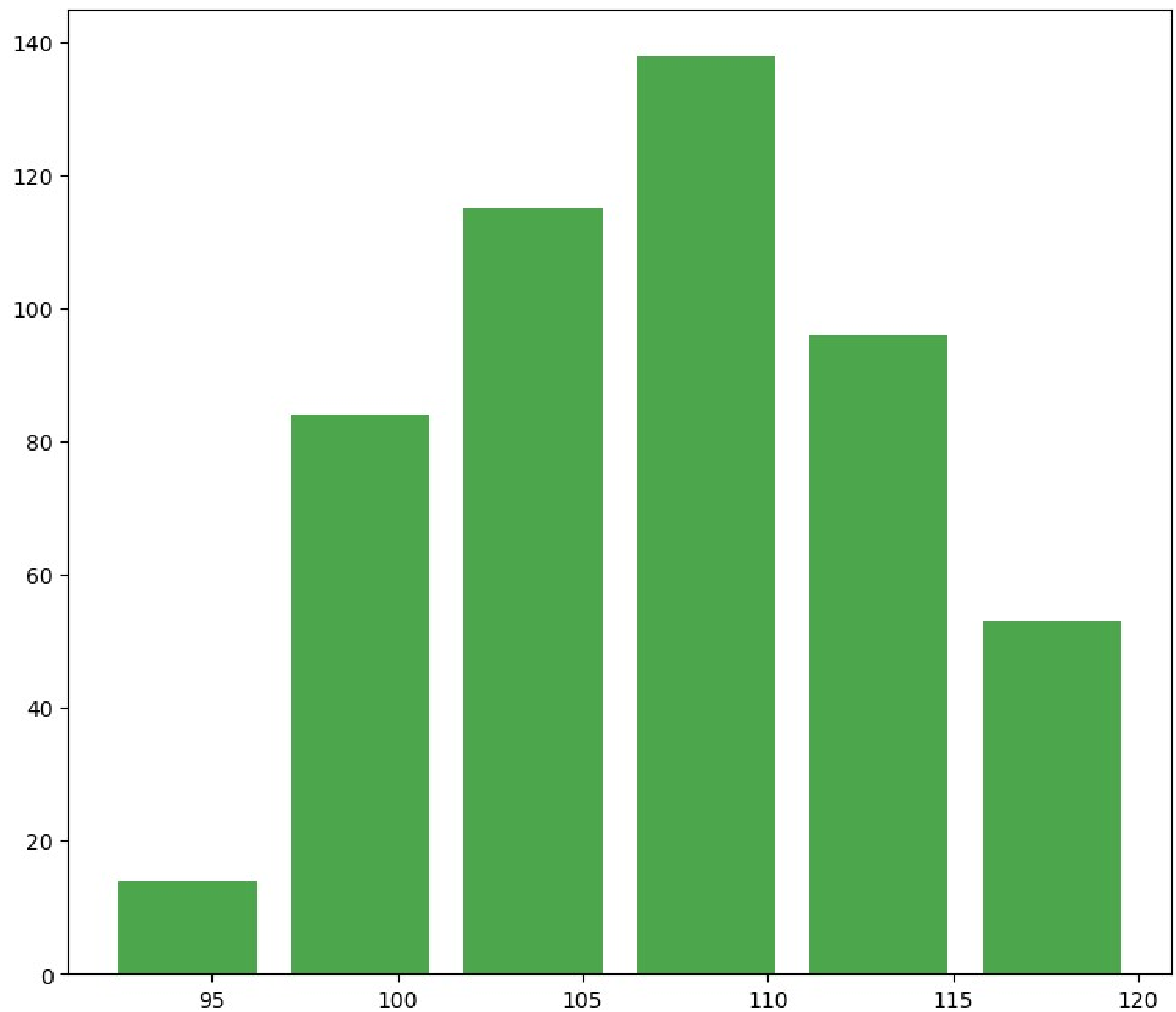
Univariate Analysis

```
plt.figure(figsize=(20,8))
plt.subplot(1,2,1)
researchcount=jb["Research"].value_counts()
ax=sns.countplot(data=jb,x="Research")
for bars in ax.containers:
   ax.bar_label(bars)
plt.subplot(1,2,2)
plt.pie(
researchcount,
labels=researchcount.index,
startangle=90,
autopct="%.2f")
plt.show()
```



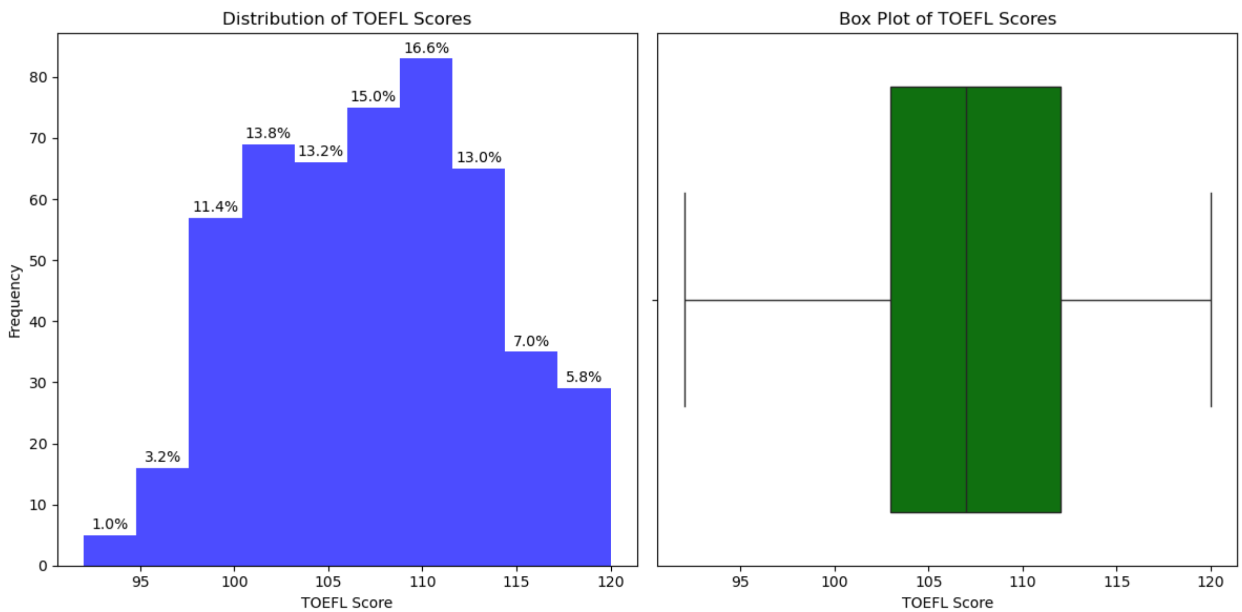
Insight More than half of the have research which is 56%.

```
plt.figure(figsize=(20,8))
plt.subplot(1,2,1)
counts, bins, patches = plt.hist(jb['TOEFL Score'], bins=6, alpha=0.7,
color='green', rwidth=0.8)
# Calculate and annotate percentages
total = len(jb)
```




```
plt.figure(figsize=(12, 6))

# 1. Create a histogram for TOEFL scores
plt.subplot(1, 2, 1)
counts, bins, patches = plt.hist(jb['TOEFL Score'], bins=10,
color='blue', alpha=0.7)
total = sum(counts)

# Add percentage labels on top of the bars
for count, patch in zip(counts, patches):
    height = patch.get_height()
    plt.text(patch.get_x() + patch.get_width() / 2, height + 1,
             '{:.1f}%'.format(100 * count / total), ha='center')

# Set labels and title for the histogram
plt.xlabel('TOEFL Score')
plt.ylabel('Frequency')
```

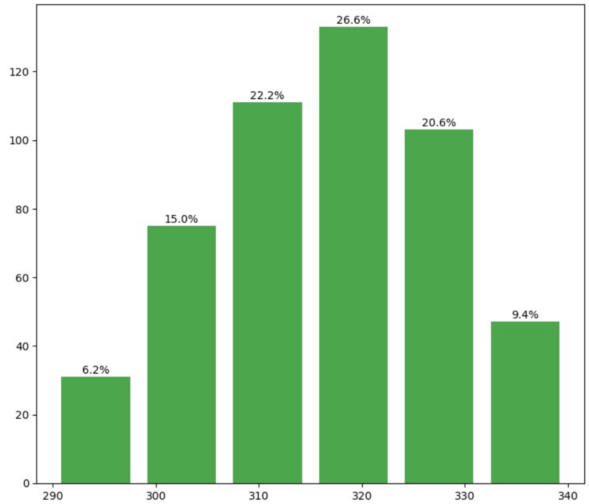
```
plt.title('Distribution of TOEFL Scores')

# 2. Create a box plot for TOEFL scores
plt.subplot(1, 2, 2)
sns.boxplot(data=jb, x="TOEFL Score", color="green")
plt.title('Box Plot of TOEFL Scores')

# Show the plots
plt.tight_layout()
plt.show()
```

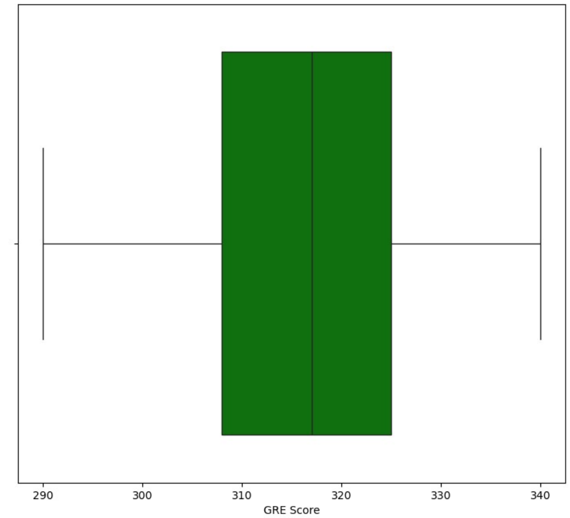


Insight

1. About 50% students score about 100-110.
2. There is no outliers in TOEFL Score.

```
plt.figure(figsize=(20,8))
plt.subplot(1,2,1)
#jb['GRE Score'].plot(kind='hist', bins=6, alpha=0.7, color='green',
figsize=(10, 6), width=2.5)
counts, bins, patches = plt.hist(jb['GRE Score'], bins=6, alpha=0.7,
color='green', rwidth=0.8)
# Calculate and annotate percentages
total = len(jb)
for count, patch in zip(counts, patches):
   height = patch.get_height()
   plt.text(patch.get_x() + patch.get_width() / 2, height + 1, '{:.1f}
%'.format(100 * count / total), ha='center')
plt.subplot(1,2,2)
sns.boxplot(data=jb,x="GRE Score",color="green")
plt.show()
```

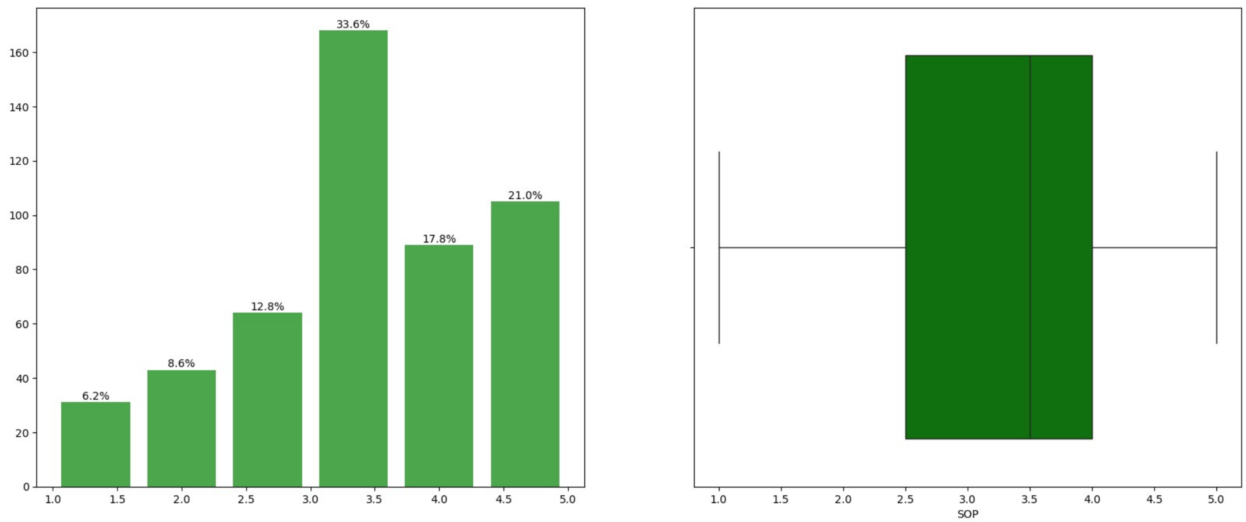




Insight

1. About 49% of students scored 308-322 in GRE.
2. 9.4% students scored 332-339 and 6.2% scored 291-228.
3. There are no outliers in GRE Score.

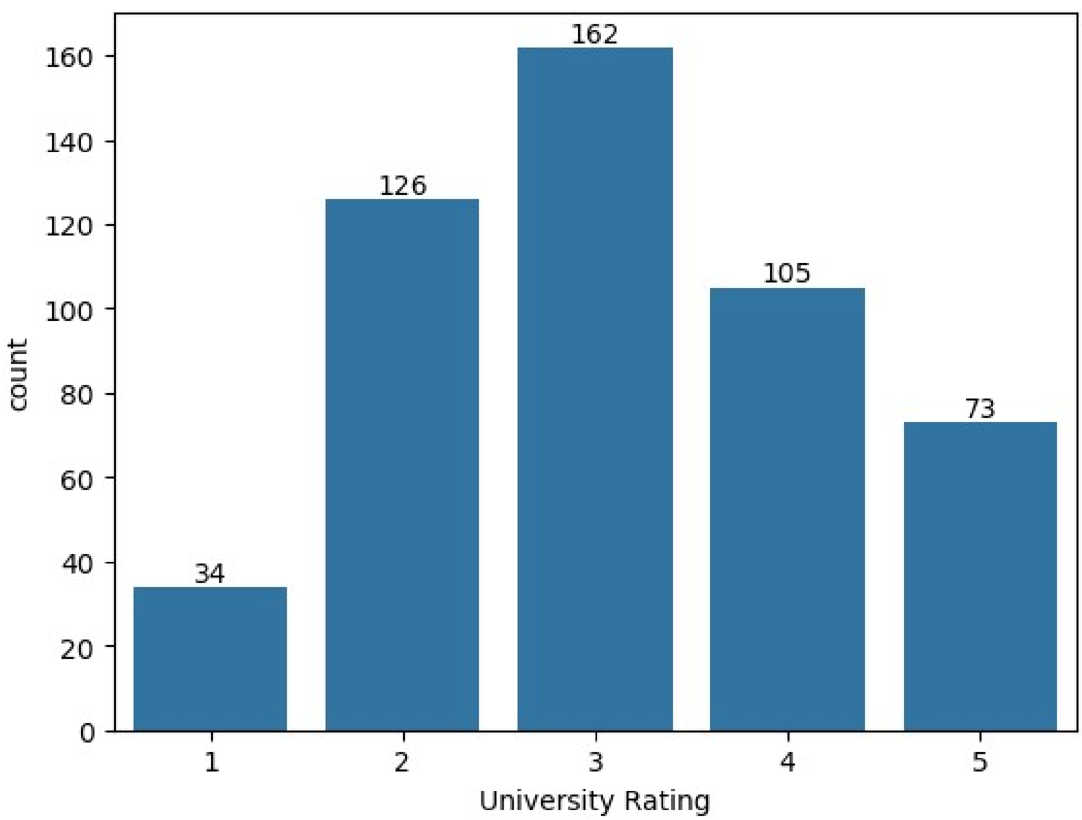
```
plt.figure(figsize=(20,8))
plt.subplot(1,2,1)
#jb['SOP'].plot(kind='hist', bins=9, alpha=0.7, color='green',
figsize=(10, 6),width=0.4)
counts, bins, patches = plt.hist(jb['SOP'], bins=6, alpha=0.7,
color='green', rwidth=0.8)
# Calculate and annotate percentages
total = len(jb)
for count, patch in zip(counts, patches):
  height = patch.get_height()
  plt.text(patch.get_x() + patch.get_width() / 2, height + 1, '{:.1f}
%'.format(100 * count / total), ha='center')
plt.subplot(1,2,2)
sns.boxplot(data=jb,x="SOP",color="green")
plt.show()
```



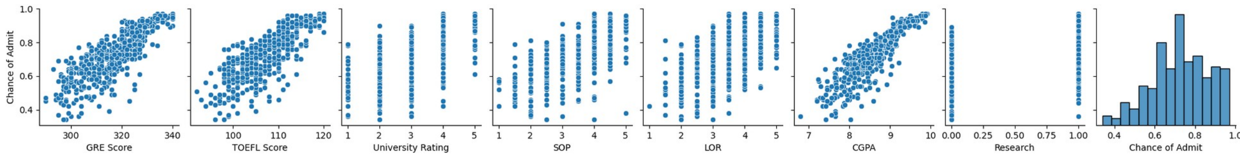
Insight

1. About 50% of students have SOP score around 3.5 to 4.0.
2. There are no outliers in GRE Score.

```
researchcount=jb["University Rating"].value_counts()
ax1=sns.countplot(data=jb,x="University Rating")
for bars in ax1.containers:
  ax1.bar_label(bars)
```



```
jb.columns = jb.columns.str.strip()
sns.pairplot(data=jb, y_vars=["Chance of Admit"])
plt.show()
```



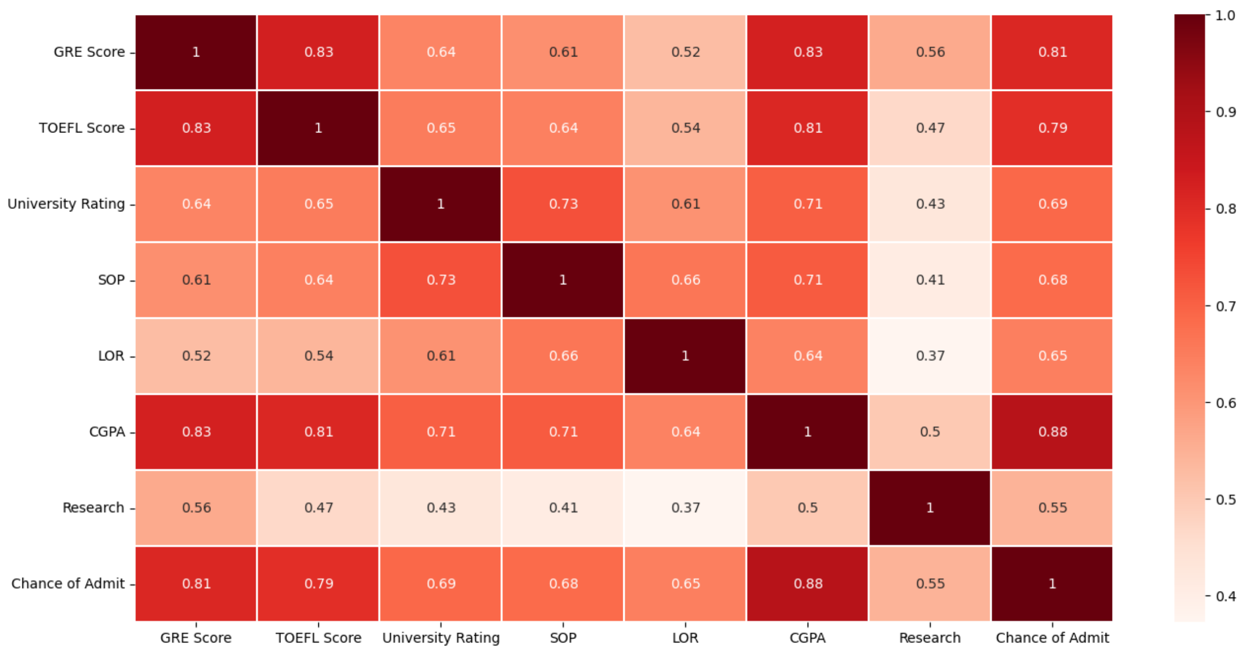
Insight

1. By the above series of graph, chance of admit increases by increament in GRE Score. Same trend can be seen in TOEFL Score and CGPA.
2. On the other hand, University Rating, SOP, LOR, Research doesn't show any trend in chance of admit.

Bivariate Analysis

Correlation Analysis

```
plt.figure(figsize=(16,8))
sns.heatmap(jb.corr(), annot=True, cmap='Reds',linewidths=0.1)
plt.show()
```



Insight

1. CGPA have the highest correlation with the chance of admission.
2. Research have the lowest correlation with the chance of admission.

```
scaler = StandardScaler()
scaled_jb = pd.DataFrame(scaler.fit_transform(jb), columns =
jb.columns)
scaled_jb
```

```
     GRE Score  TOEFL Score  University Rating       SOP       LOR
CGPA  \
0     1.819238     1.778865           0.775582  1.137360  1.098944
1.776806
1     0.667148    -0.031601           0.775582  0.632315  1.098944
0.485859
2    -0.041830    -0.525364          -0.099793 -0.377773  0.017306 -
0.954043
3     0.489904     0.462163          -0.099793  0.127271 -1.064332
0.154847
4    -0.219074    -0.689952          -0.975168 -1.387862 -0.523513 -
0.606480
..         ...          ...                ...       ...       ...
...
495   1.376126     0.132987           1.650957  1.137360  0.558125
0.734118
496   1.819238     1.614278           1.650957  1.642404  1.639763
2.140919
497   1.198882     2.108041           1.650957  1.137360  1.639763
1.627851
498  -0.396319    -0.689952           0.775582  0.632315  1.639763 -
0.242367
499   0.933015     0.955926           0.775582  1.137360  1.098944
0.767220

     Research  Chance of Admit
0    0.886405         1.406107
1    0.886405         0.271349
2    0.886405        -0.012340
3    0.886405         0.555039
4   -1.128152        -0.508797
..        ...              ...
495  0.886405         1.051495
496  0.886405         1.689797
497  0.886405         1.477030
498 -1.128152         0.058582
499 -1.128152         0.838728

[500 rows x 8 columns]
```

Splitting data for training and testing

```
x=scaled_jb.iloc[:,:-1]
y=scaled_jb.iloc[:,-1]
print(x.shape,y.shape)
```

```
(500, 7) (500,)
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, 
test_size=0.2,random_state=42)
print(f'Shape of x_train: {x_train.shape}')
print(f'Shape of x_test: {x_test.shape}')
print(f'Shape of y_train: {y_train.shape}')
print(f'Shape of y_test: {y_test.shape}')
```

```
Shape of x_train: (400, 7)
Shape of x_test: (100, 7)
Shape of y_train: (400,)
Shape of y_test: (100,)
```

Linear Regression

```
lr_model = LinearRegression()
lr_model.fit(x_train,y_train)
```

```
LinearRegression()
```

```
y_pred_train = lr_model.predict(x_train)
y_pred_test = lr_model.predict(x_test)
```

R2 score on train data

```
r2=r2_score(y_train,y_pred_train)
print("r2 score-> ",r2)
lr=lr_model.score(x_train,y_train)
print("lr score-> ",lr)
```

```
r2 score->  0.8210671369321554
lr score->  0.8210671369321554
```

R2 score on test data

```
r2_score(y_test,y_pred_test)
print("r2 score-> ",r2)
lr=lr_model.score(x_test,y_test)
print("lr score-> ",lr)
```

```
r2 score->  0.8210671369321554
lr score->  0.8188432567829628
```

All features coefficients and features

```
lr_model_weights = pd.DataFrame(lr_model.coef_.reshape(1,-
1),columns=jb.columns[:-1])
lr_model_weights["Intercept"] = lr_model.intercept_
lr_model_weights
```

```
   GRE Score  TOEFL Score  University Rating       SOP       LOR
CGPA  \
0   0.194823     0.129095           0.020812  0.012735  0.113028
0.482199

   Research  Intercept
0  0.084586   0.007736
```

Insight

1. CGPA,GRE Score,TOEFL Score have highest weights.
2. University Rating, SOP, Research have lowest weights.
3. Intercept (w0) is very low

```
import numpy as np
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score

def model_evaluation(y_actual, y_forecast, model):
    n = len(y_actual)

    # Determine the number of coefficients
    if len(model.coef_.shape) == 1:
        p = len(model.coef_)
    else:
        p = len(model.coef_[0])

    # Calculate error metrics
    MSE = np.round(mean_squared_error(y_true=y_actual,
y_pred=y_forecast, squared=True), 2)
    MAE = np.round(mean_absolute_error(y_true=y_actual,
y_pred=y_forecast), 2)
    RMSE = np.round(mean_squared_error(y_true=y_actual,
y_pred=y_forecast, squared=False), 2)

    # Calculate R2 and Adjusted R2
    r2 = np.round(r2_score(y_true=y_actual, y_pred=y_forecast), 2)
    adj_r2 = np.round(1 - ((1 - r2) * (n - 1) / (n - p - 1)), 2)

    # Print the evaluation metrics
    print(f"MSE: {MSE}\nMAE: {MAE}\nRMSE: {RMSE}\nR2 Score: {r2}\
nAdjusted R2: {adj_r2}")

# Example usage (you would replace y_actual, y_forecast, and model
```

```
with your actual data)
# model_evaluation(y_actual, y_forecast, model)
```

```
model_evaluation(y_train.values, y_pred_train, lr_model)
```

```
MSE: 0.18
MAE: 0.3
RMSE: 0.42
R2 Score: 0.82
Adjusted R2: 0.82
```

```
model_evaluation(y_test.values, y_pred_test, lr_model)
```

```
MSE: 0.19
MAE: 0.3
RMSE: 0.43
R2 Score: 0.82
Adjusted R2: 0.81
```

Linear Regression using OLS

```
new_x_train = sm.add_constant(x_train)
model = sm.OLS(y_train, new_x_train)
results = model.fit()
# statstical summary of the model
print(results.summary())
```

```
                            OLS Regression Results

======================================================================
========
Dep. Variable:        Chance of Admit   R-squared:
0.821
Model:                            OLS   Adj. R-squared:
0.818
Method:                 Least Squares   F-statistic:
257.0
Date:                Mon, 28 Oct 2024   Prob (F-statistic):
3.41e-142
Time:                        10:55:50   Log-Likelihood:
-221.69
No. Observations:                 400   AIC:
459.4
Df Residuals:                     392   BIC:
491.3
Df Model:                           7

Covariance Type:            nonrobust

======================================================================
```

```
===============
                        coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
---------------
const                 0.0077      0.021      0.363      0.717      -
0.034      0.050
GRE Score             0.1948      0.046      4.196      0.000
0.104      0.286
TOEFL Score           0.1291      0.041      3.174      0.002
0.049      0.209
University Rating     0.0208      0.034      0.611      0.541      -
0.046      0.088
SOP                   0.0127      0.036      0.357      0.721      -
0.057      0.083
LOR                   0.1130      0.030      3.761      0.000
0.054      0.172
CGPA                  0.4822      0.046     10.444      0.000
0.391      0.573
Research              0.0846      0.026      3.231      0.001
0.033      0.136
==============================================================================
========
Omnibus:                       86.232   Durbin-Watson:
2.050
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
190.099
Skew:                          -1.107   Prob(JB):
5.25e-42
Kurtosis:                       5.551   Cond. No.
5.72
==============================================================================
========

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

Testing Assumptions of Linear Regression Model

1. No multicolinearity: Multicollinearity check by VIF(Variance Inflation Factor) score. Variables are dropped one-by-one till none has a VIF>5.
2. Mean of Residuals should be close to zero.
3. Linear relationship between independent & dependent variables. This can be checked using the following methods: Scatter plots Regression plots Pearson Correlation
4. Test for Homoscedasticity Create a scatterplot of residuals against predicted values. r2 = np.round(r2_score(y_true=y_actual, y_pred=y_forecast),2) adj_r2 = np.round(1 - ((1-r2)*(n-1)/(n-p-1)),2) return print(f"MSE: {MSE}\nMAE: {MAE}\nRMSE: {RMSE}\nR2 Score: {r2}\nAdjusted R2: {adj_r2}") model_evaluation(y_train.values, y_pred_train, lr_model)

model_evaluation(y_test.values, y_pred_test, lr_model)

new_x_train = sm.add_constant(x_train) model = sm.OLS(y_train, new_x_train) results = model.fit()

statstical summary of the model

print(results.summary())

Perform a Goldfeld-Quandt test to check the presence of Heteroscedasticity in the data.

- If the obtained `p-value > 0.05`, there is no strong evidence of heteroscedasticity.

1. Normality of Residuals Almost bell-shaped curve in residuals distribution.

2. Impact of Outliers

Multicolinearity check:

VIF (Variance Inflation Factor) is a measure that quantifies the severity of multicollinearity in a regression analysis. It assesses how much the variance of the estimated regression coefficient is inflated due to collinearity. The formula for VIF is as follows: VIF(j) = 1 / (1 - R(j)^2)

Where:

j represents the jth predictor variable.

R(j)^2 is the coefficient of determination (R-squared) obtained from regressing the jth predictor variable on all the other predictor variables. "

Calculate the VIF for each variable. Identify variables with VIF greater than 5. Drop the variable with the highest VIF. Repeat steps 1-3 until no variable has a VIF greater than 5.

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

	Variable	VIF
5	CGPA	4.653698
0	GRE Score	4.489201
1	TOEFL Score	3.665067
3	SOP	2.785753
2	University Rating	2.571847
4	LOR	1.977668
6	Research	1.517206

Insight

As the Variance Inflation Factor(VIF) score is less than 5 for all the features we can say that there is no much multicolinearity between the features.

Mean Of Residuals

1. If mean of residuals is significantly non-zero, then the model is overestimating or underestimating the observed values.
2. If the mean of residuals is close to zero then on average predections made by linear regression model are accurate, within the equal balance of overestimating and underestimating. This is the desired charecteristics for well-fitted regression model.

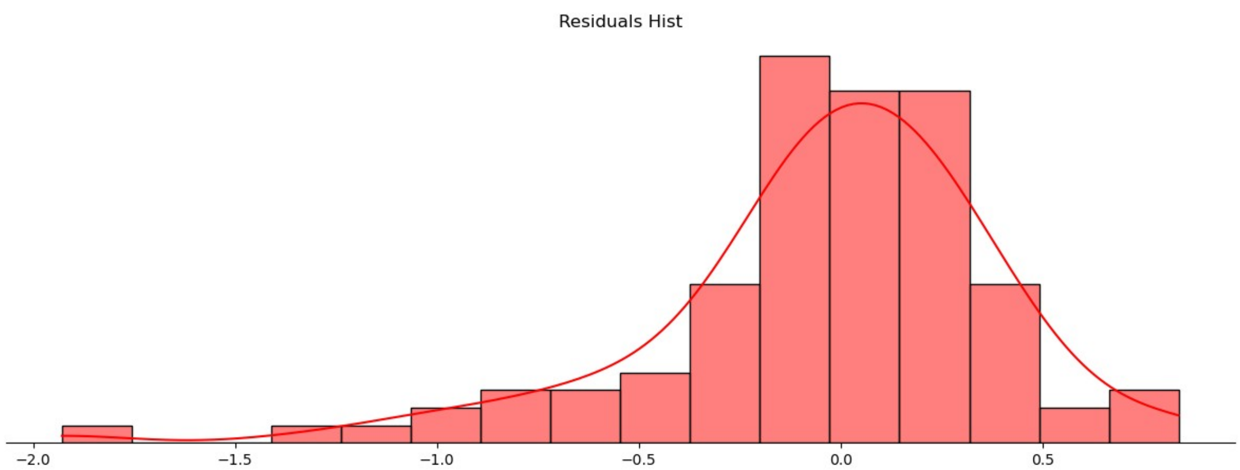
```
residual = y_test.values - y_pred_test

residual_train = y_train.values - y_pred_train
residual_train.mean()
```

```
3.7192471324942746e-17
```

```
residual.mean()
```

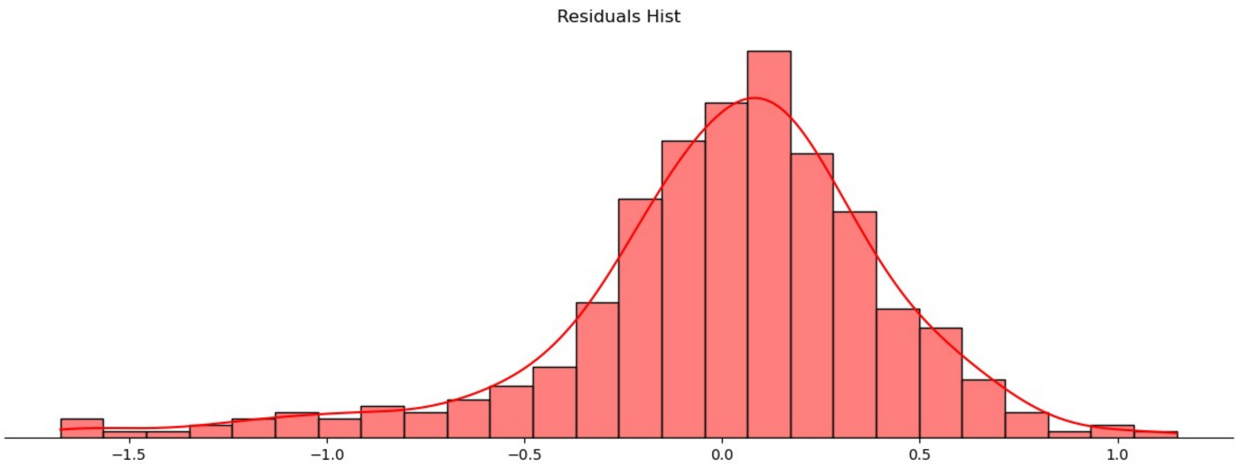
```
-0.03867840379282768
```

```
plt.figure(figsize=(15,5))
sns.histplot(residual, kde= True,color='r')
plt.title('Residuals Hist')
sns.despine(left=True)
plt.ylabel("")
plt.yticks([])
plt.show()
```




```
plt.figure(figsize=(15,5))
sns.histplot(residual_train, kde= True,color='r')
plt.title('Residuals Hist')
sns.despine(left=True)
```

```
plt.ylabel("")
plt.yticks([])
plt.show()
```

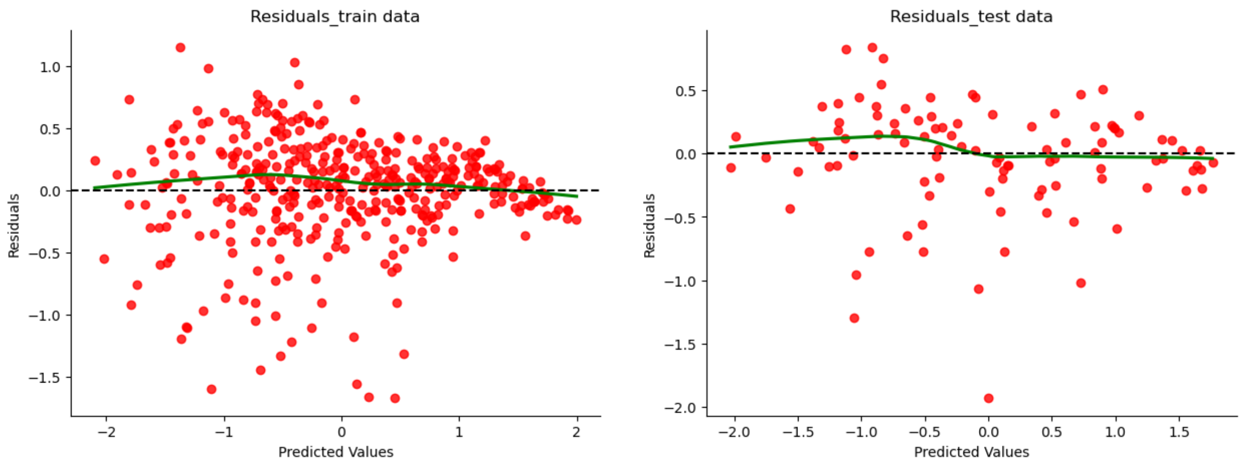


Insight

The mean of residual is close to zero, therefore our model is unbiased

Linear Relationships:

```
plt.figure(figsize=(15,5))
plt.subplot(121)
plt.title('Residuals_train data')
sns.regplot(x=y_pred_train, y=residual_train, lowess=True,
color='r',line_kws={'color': 'green'})
plt.axhline(y=0, color='k', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.subplot(122)
plt.title('Residuals_test data')
sns.regplot(x=y_pred_test, y=residual,
lowess=True,color='r' ,line_kws={'color': 'green'})
plt.axhline(y=0, color='k', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
sns.despine()
plt.show()
```

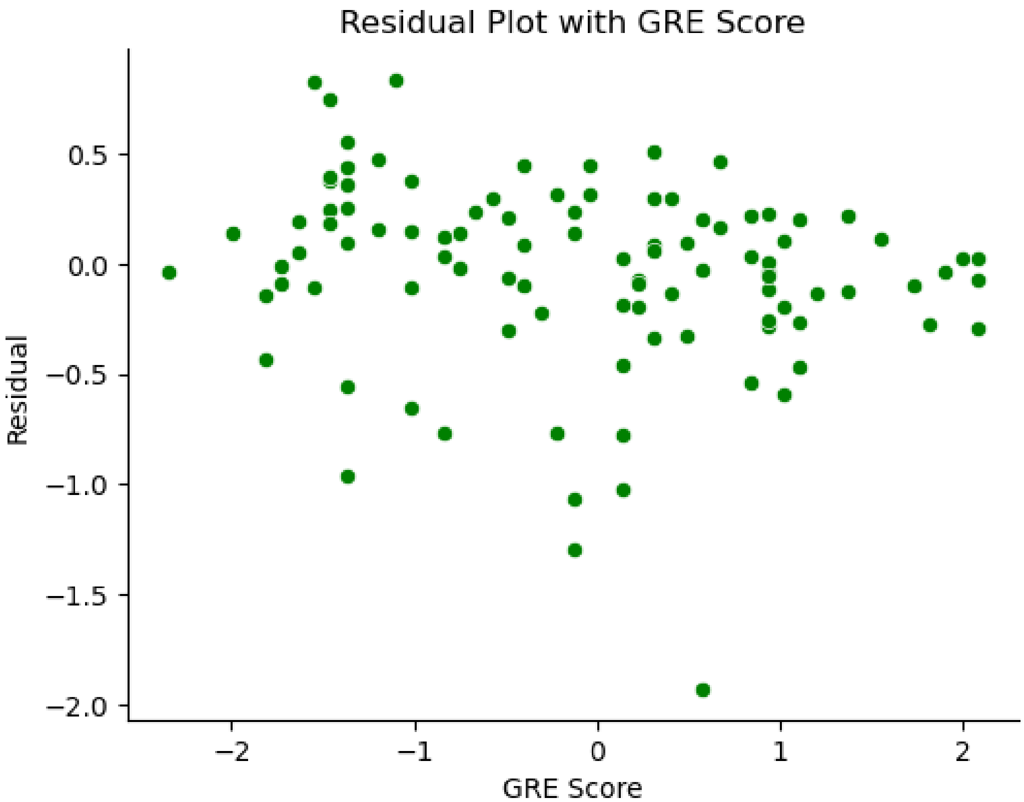


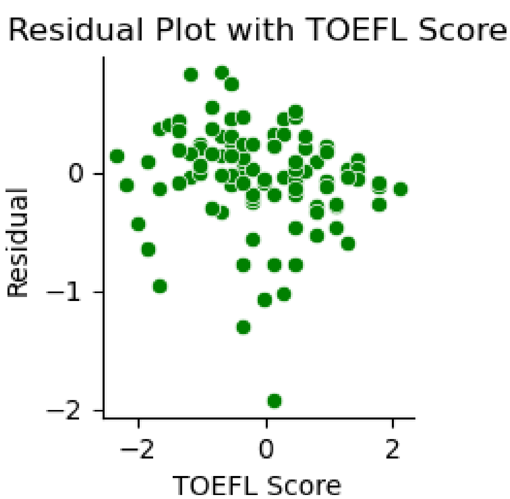
Insights:

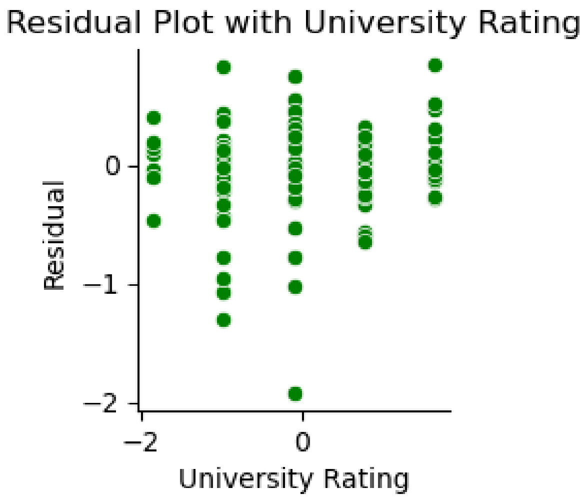
1. From the Joint plot & pairplot in the graphical analysis, we can say that there is linear relationship between dependent variable and independent variables.

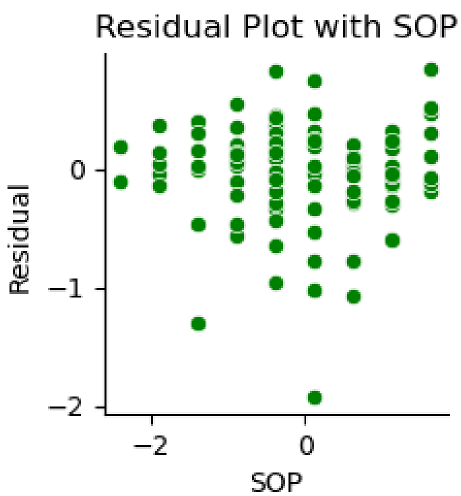
Homoscedacity

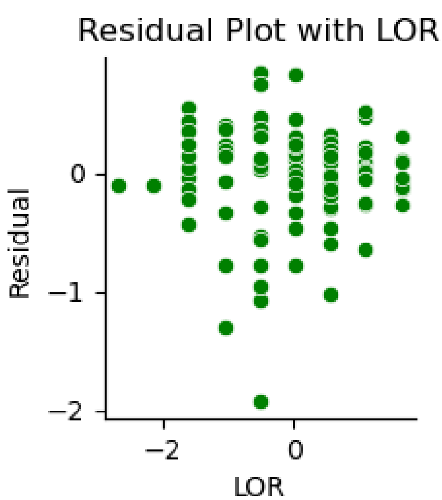
```
# Scatterplot of residuals with each independent variable to check for Homoscedasticity
plt.figure(figsize=(15,8))
i=1
for col in x_test.columns[:-1]:
    plt.subplot(2,3,i)
    sns.scatterplot(x=x_test[col].values.reshape((-1,)),
y=residual.reshape((-1,)),color='g')
    plt.title(f'Residual Plot with {col}')
    plt.xlabel(col)
    plt.ylabel('Residual')
    i+=1
    plt.tight_layout()
    sns.despine()
    plt.show();
```

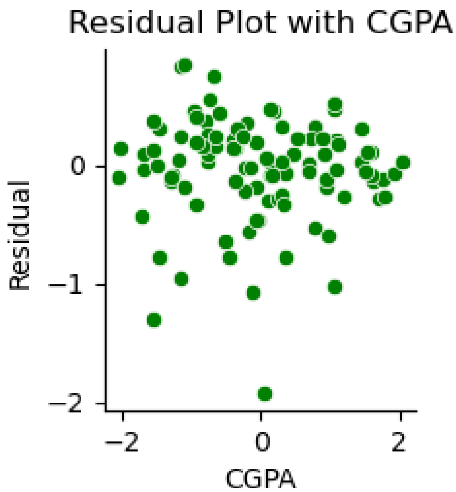

Residual Plot with GRE Score
Residual
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2
−1
0
1
2
GRE Score


Residual Plot with TOEFL Score
Residual
0
−1
−2
−2
0
2
TOEFL Score


Residual Plot with University Rating
Residual
0
−1
−2
−2
0
University Rating


Residual Plot with SOP
Residual
0
−1
−2
−2
0
SOP


Residual Plot with LOR
Residual
0
−1
−2
−2
0
LOR




```
ols_model = results
predicted = ols_model.predict()
residuals = ols_model.resid
```

Breusch-Pagan test for Homoscedasticity

Null Hypothesis -- H0 : Homoscedasticity is present in residuals. Alternate Hypothesis -- Ha : Heteroscedasticity is present in residuals. alpha : 0.05

Insights

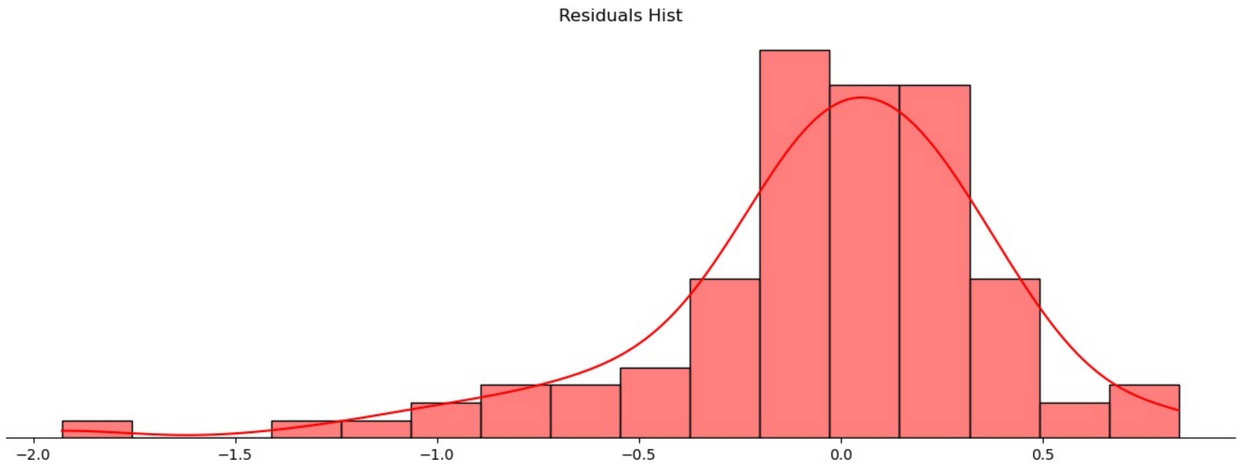
Since we do not see any significant change in the spread of residuals with respect to change in independent variables, we can conclude that Homoscedasticity is met. Since the p-value is much lower than the alpha value, we can Reject the null hypothesis and conclude that *Heteroscedasticity is present* Since the p-value is significantly less than the conventional significance level (e.g., 0.05), we reject the null hypothesis of homoscedasticity. This suggests that there is evidence of heteroscedasticity in the residuals, indicating that the variance of the residuals is not constant across all levels of the independent variables. This violation of the homoscedasticity assumption may affect the validity of the linear regression model's results.

Normality of Residuals:

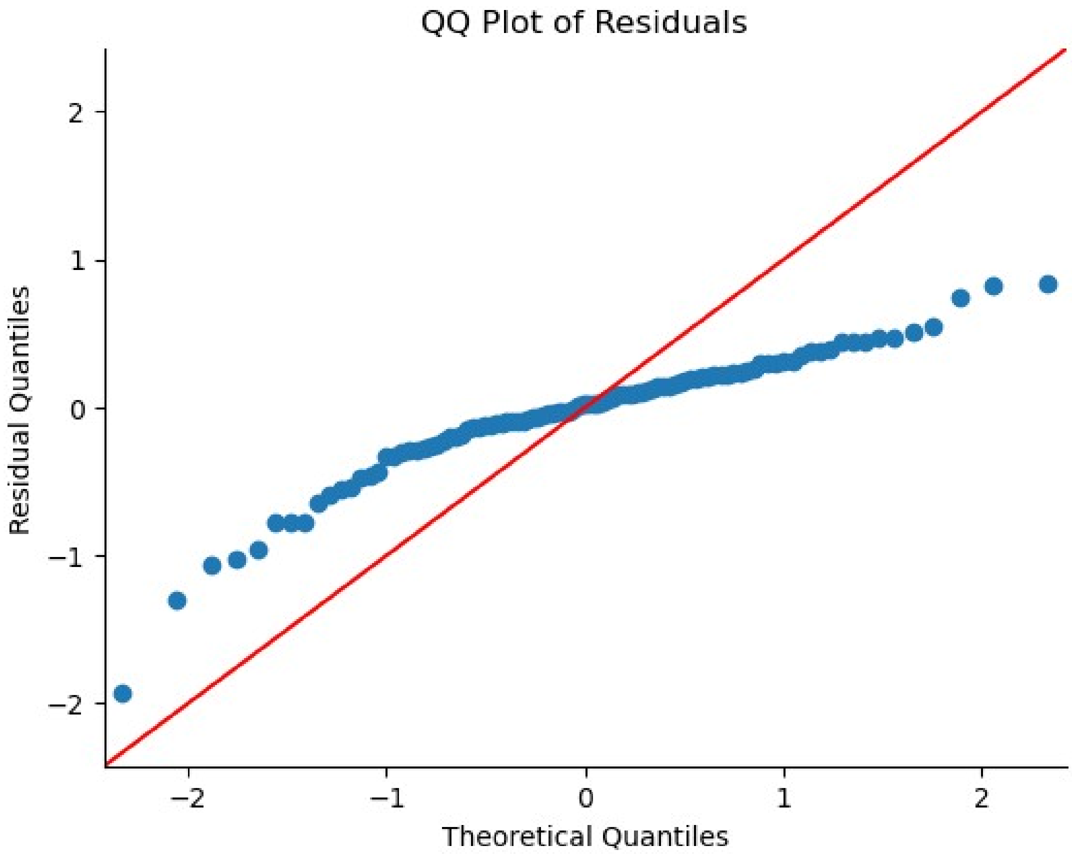
To check normality, we will follow below methods:-

1. Residual Histogram
2. Q-Q Plot
3. Anderson-Darling or Jarque_Bera Test

```
plt.figure(figsize=(15,5))
sns.histplot(residual, kde= True,color='r')
plt.title('Residuals Hist')
sns.despine(left=True)
plt.ylabel("")
plt.yticks([])
plt.show()
```




```
plt.figure(figsize=(15,5))
sm.qqplot(residual,line='45')
plt.title('QQ Plot of Residuals')
plt.ylabel('Residual Quantiles')
sns.despine()
plt.show()

<Figure size 1500x500 with 0 Axes>
```



JARQUE BERA test:

```
jb_stat, jb_p_value = stats.jarque_bera(residual)
print("Jarque-Bera Test Statistic:", jb_stat)
print("p-value:", jb_p_value)
if jb_p_value < 0.05:
    print("Reject the null hypothesis: Residuals are not normally
distributed.")
else:
    print("Fail to reject the null hypothesis: Residuals are normally
distributed.")

Jarque-Bera Test Statistic: 74.10190609972128
p-value: 8.109153870348849e-17
Reject the null hypothesis: Residuals are not normally distributed.
```

Jarque-Bera Test Statistic: 74.10190609972094 p-value: 8.109153870350212e-17 Reject the null hypothesis: Residuals are not normally distributed.

Insight

1. From Hisplot and Kdeplot we can say that Residuals are left skewed.
2. The QQ plot shows that residuals are slightly deviating from the straight diagonal , thus not Gaussian.
3. From Jarque Bera test , we conclude that the Residuals are Not Normally distributed. Hence this assumption is not met.

Lasso and Ridge Regression - L1 & L2 Regularization

Lasso Regression:

```
model_lasso = Lasso(alpha=0.45)


model_lasso = Lasso(alpha=0.45)
model_lasso.fit(x_train, y_train)

Lasso(alpha=0.45)

model_ridge = Ridge()
model_ridge.fit(x_train, y_train)

Ridge()

y_pred_train_ridge = model_ridge.predict(x_train)
y_pred_test_ridge = model_ridge.predict(x_test)
y_pred_train_lasso = model_lasso.predict(x_train)
y_pred_test_lasso = model_lasso.predict(x_test)

lasso_model_weights = pd.DataFrame(model_lasso.coef_.reshape(1,-
1),columns=jb.columns[:-1])
lasso_model_weights["Intercept"] = model_lasso.intercept_
lasso_model_weights

   GRE Score  TOEFL Score  University Rating  SOP  LOR      CGPA
Research  \
0   0.019231          0.0                0.0  0.0  0.0  0.408647
0.0

   Intercept
0   0.013919

ridge_model_weights = pd.DataFrame(model_ridge.coef_.reshape(1,-
1),columns=jb.columns[:-1])
ridge_model_weights["Intercept"] = model_ridge.intercept_
ridge_model_weights
```

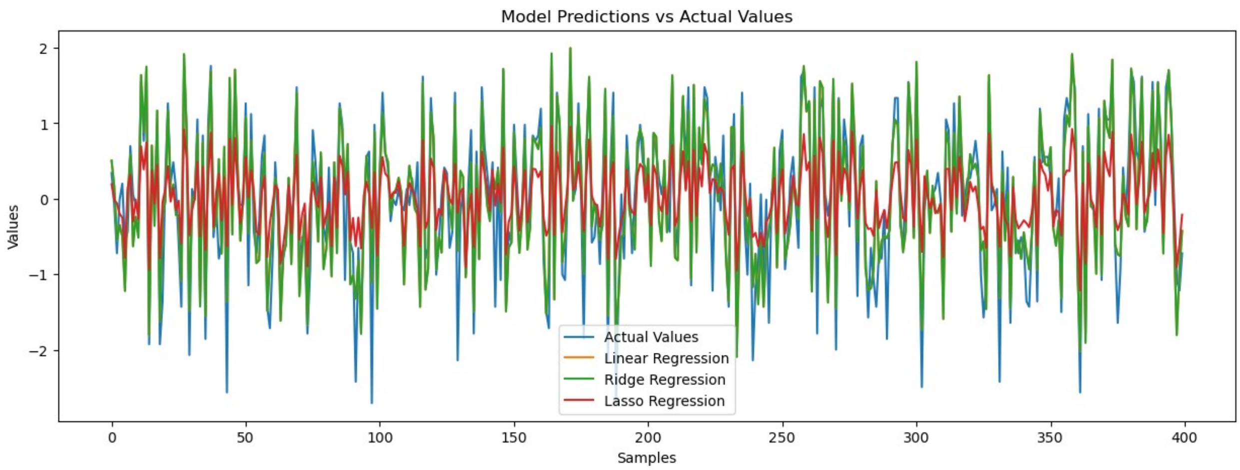
```
   GRE Score  TOEFL Score  University Rating       SOP       LOR
CGPA  \
0   0.195584     0.130073           0.021575  0.013802  0.113221
0.478123

   Research  Intercept
0  0.084673   0.007726
```

```
print('Linear Regression Training Accuracy\n')
model_evaluation(y_train.values, y_pred_train, lr_model)
print('-'*25)
print('\nLinear Regression Test Accuracy\n')
model_evaluation(y_test.values, y_pred_test, lr_model)
print('---'*25)
print('\nRidge Regression Training Accuracy\n')
model_evaluation(y_train.values, y_pred_train_ridge, model_ridge)
print('-'*25)
print('\n\nRidge Regression Test Accuracy\n')
model_evaluation(y_test.values, y_pred_test_ridge, model_ridge)
print('---'*25)
print('\n\nLasso Regression Training Accuracy\n')
model_evaluation(y_train.values, y_pred_train_lasso, model_lasso)
print('-'*25)
print('\n\nLasso Regression Test Accuracy\n')
model_evaluation(y_test.values, y_pred_test_lasso, model_lasso)
print('---'*25)
```

```
Linear Regression Training Accuracy

MSE: 0.18
MAE: 0.3
RMSE: 0.42
R2 Score: 0.82
Adjusted R2: 0.82
-------------------------

Linear Regression Test Accuracy

MSE: 0.19
MAE: 0.3
RMSE: 0.43
R2 Score: 0.82
Adjusted R2: 0.81
----------------------------------------------------------------------
-----

Ridge Regression Training Accuracy

MSE: 0.18
MAE: 0.3
```

```
RMSE: 0.42
R2 Score: 0.82
Adjusted R2: 0.82
-------------------------


Ridge Regression Test Accuracy

MSE: 0.19
MAE: 0.3
RMSE: 0.43
R2 Score: 0.82
Adjusted R2: 0.81
----------------------------------------------------------------------
-----


Lasso Regression Training Accuracy

MSE: 0.43
MAE: 0.52
RMSE: 0.65
R2 Score: 0.57
Adjusted R2: 0.56
-------------------------


Lasso Regression Test Accuracy

MSE: 0.43
MAE: 0.51
RMSE: 0.65
R2 Score: 0.58
Adjusted R2: 0.55
----------------------------------------------------------------------
-----

# Assuming y_pred_train, y_pred_train_ridge, and y_pred_train_lasso
are your prediction arrays
actual_values = y_train.values.reshape((-1,))

predicted_values = [
    y_pred_train.reshape((-1,)),
    y_pred_train_ridge.reshape((-1,)),
    y_pred_train_lasso.reshape((-1,))  # Change from reshape(()) to
reshape((-1,))
]

# Continue with your plotting or analysis
model_names = ['Linear Regression', 'Ridge Regression', 'Lasso
Regression']
```

```
plt.figure(figsize=(15, 5))

# Example plotting (assuming you want to plot these values)
plt.plot(actual_values, label='Actual Values')
for i, preds in enumerate(predicted_values):
    plt.plot(preds, label=model_names[i])

plt.legend()
plt.title('Model Predictions vs Actual Values')
plt.xlabel('Samples')
plt.ylabel('Values')
plt.show()
```



Elastic-Net Regression

```
ElasticNet_model = ElasticNet(alpha=0.108)
ElasticNet_model.fit(x_train , y_train)
```

```
ElasticNet(alpha=0.108)
```

```
y_pred_train_el = ElasticNet_model.predict(x_train)
y_pred_test_el = ElasticNet_model.predict(x_test)
```

```
train_R2 = ElasticNet_model.score(x_train,y_train)
test_R2 = ElasticNet_model.score(x_test,y_test)
train_R2 , test_R2
```

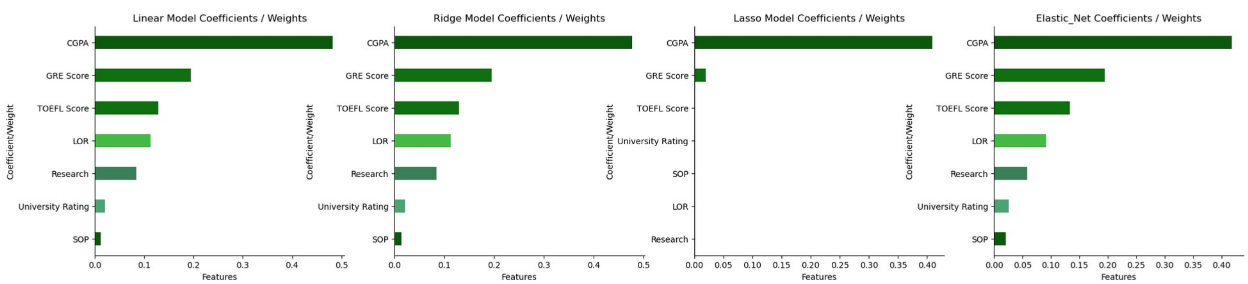
```
(0.814348667393518, 0.8153699952125263)
```

```
train_R2 = ElasticNet_model.score(x_train,y_train)
test_R2 = ElasticNet_model.score(x_test,y_test)
train_R2 , test_R2
```

```
(0.814348667393518, 0.8153699952125263)
```

```
en_model_weights = pd.DataFrame(ElasticNet_model.coef_.reshape(1,-
1),columns=jb.columns[:-1])
en_model_weights["Intercept"] = ElasticNet_model.intercept_
en_model_weights

   GRE Score  TOEFL Score  University Rating      SOP       LOR
CGPA  \
0   0.194912      0.13349           0.025508  0.02075  0.091694
0.418398

   Research  Intercept
0  0.058465   0.007728

print('ElasticNet Regression Training Accuracy\n')
model_evaluation(y_train.values, y_pred_train_el, ElasticNet_model)
print('*'*25)
print('\nElasticNet Regression Test Accuracy\n')
model_evaluation(y_test.values, y_pred_test_el, ElasticNet_model)
print('---'*25)

ElasticNet Regression Training Accuracy

MSE: 0.18
MAE: 0.31
RMSE: 0.43
R2 Score: 0.81
Adjusted R2: 0.81
*************************

ElasticNet Regression Test Accuracy

MSE: 0.19
MAE: 0.3
RMSE: 0.44
R2 Score: 0.82
Adjusted R2: 0.81
----------------------------------------------------------------------
-----

model_major_weights = {"Linear Model":lr_model_weights,
"Ridge Model":ridge_model_weights,
"Lasso Model":lasso_model_weights,
"Elastic_Net":en_model_weights}

# excluding w0-intercept

plt.figure(figsize=(25, 5))
i = 1

for model, data in model_major_weights.items():
    model_weights_data = data.melt()  # Melt the DataFrame for
```

```
plotting
    plt.subplot(1, 4, i)  # Create subplots
    sns.barplot(
        data=model_weights_data[:-1].sort_values(by='value',
ascending=False),  # Sort values
        y='variable',
        x='value',
        width=0.4,
        palette=['darkgreen', 'g', 'green', 'limegreen', 'seagreen',
'mediumseagreen']
    )

    plt.xlabel('Features')
    plt.ylabel('Coefficient/Weight')
    plt.title(f'{model} Coefficients / Weights')
    i += 1

sns.despine()  # Remove top and right spines from plots
plt.show()  # Display the plots
```



Regression Analysis Summary:

1. By conducting regression analysis, it's evident that CGPA emerges as the most influential feature in predicting admission chances.

2. Additionally, GRE and TOEFL scores also holds significant importance.

3. Following the initial regression model, a thorough check for multicollinearity was performed, revealing VIF scores consistently below 5, indicative of low multicollinearity among predictors.

4. Despite the absence of high multicollinearity, it's noteworthy that the residuals do not conform perfectly to a normal distribution. Furthermore, the residual plots indicate some level of heteroscedasticity.

5. After exploring involving regularized models such as Ridge and Lasso regression showed comparable results to the Linear Regression Model.

6. Moreover, employing ElasticNet (L1+L2) regression yielded results consistent with the other regression models.

Recommendation

7. Encourage students to focus on improving GRE scores, CGPA, and Letters of Recommendation (LOR), as these factors influence a lot your chances of admission.

8. Beyond academic metrics applicants can also add like extracurricular achievements, personal statements, and diversity factors.

9. We can enhance our predictive model by adding other important and diverse features like Work-experiece, internships or extra- curriculum activites.