

Jamboree - Business Case Study

✓ ABOUT:

- Jamboree is a renowned educational institution that has successfully assisted numerous students in gaining admission to top colleges abroad. With their proven problem-solving methods, they have helped students achieve exceptional scores on exams like GMAT, GRE, and SAT with minimal effort.
- To further support students, Jamboree has recently introduced a new feature on their website. This feature enables students to assess their probability of admission to Ivy League colleges, considering the unique perspective of Indian applicants.
- By conducting a thorough analysis, we can assist Jamboree in understanding the crucial factors impacting graduate admissions and their interrelationships. Additionally, we can provide predictive insights to determine an individual's admission chances based on various variables.

Why this Case study?

- Solving this business case holds immense importance for aspiring data scientists and ML engineers.
- Building predictive models using machine learning is widely popular among the data scientists/ML engineers. By working through this case study, individuals gain hands-on experience and practical skills in the field.
- Additionally, it will enhance one's ability to communicate with the stakeholders involved in data-related projects and help the organization take better, data-driven decisions.

Work that has to be done:

As a data scientist/ML engineer hired by Jamboree, your primary objective is to analyze the given dataset and derive valuable insights from it. Additionally, utilize the dataset to construct a predictive model capable of estimating an applicant's likelihood of admission based on the available features.

✓ Features of the dataset:

Column Profiling:

Feature	Description
Serial No.	This column represents the unique row identifier for each applicant in the dataset.
GRE Scores	This column contains the GRE (Graduate Record Examination) scores of the applicants, which are measured on a scale of 0 to 340.
TOEFL Scores	This column includes the TOEFL (Test of English as a Foreign Language) scores of the applicants, which are measured on a scale of 0 to 120.
University Rating	This column indicates the rating or reputation of the university that the applicants are associated with, & The rating is based on a scale of 0 to 5, with 5 representing the highest rating.
SOP	This column represents the strength of the applicant's statement of purpose, rated on a scale of 0 to 5, with 5 indicating a strong and compelling SOP.
LOR	This column represents the strength of the applicant's letter of recommendation, rated on a scale of 0 to 5, with 5 indicating a strong and compelling LOR.
CGPA	This column contains the undergraduate Grade Point Average (GPA) of the applicants, which is measured on a scale of 0 to 10.
Research	This column indicates whether the applicant has research experience (1) or not (0).
Chance of Admit	This column represents the estimated probability or chance of admission for each applicant, ranging from 0 to 1.

These columns provide relevant information about the applicants' academic qualifications, test scores, university ratings, and other factors that may influence their chances of admission.

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


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
```



```
import warnings
warnings.filterwarnings('ignore')

# importing dataset and doing further operations

jm_data = pd.read_csv('/content/sample_data/Jamboree_Admission.csv')
df = jm_data.copy()
df.tail()
```




	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
495	496	332	108	5	4.5	4.0	9.02	1	0.87
496	497	337	117	5	5.0	5.0	9.87	1	0.96
497	498	330	120	5	4.5	5.0	9.56	1	0.93
498	499	312	103	4	4.0	5.0	8.43	0	0.73
499	500	327	113	4	4.5	4.5	9.04	0	0.84



✖ Exploration of data :


```
df = df.rename(columns={'Chance of Admit ': 'Chance_of_Admit'})
```

```
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
```

✖ Statistical Summary

```
df.cov()
```





	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance_of_Admit
Serial No.	20875.000000	-169.458918	-124.511022	-11.175351	-19.666333	-0.493988	-6.491703	-0.382766	0.173437
GRE Score	-169.458918	127.580377	56.825026	8.206605	6.867206	5.484521	5.641944	3.162004	1.291862
TOEFL Score	-124.511022	56.825026	36.989114	4.519150	3.883960	3.048168	2.981607	1.411303	0.680046
University Rating	-11.175351	8.206605	4.519150	1.307619	0.825014	0.644112	0.487761	0.242645	0.111384
SOP	-19.666333	6.867206	3.883960	0.825014	0.982088	0.608701	0.426845	0.200962	0.095691
LOR	-0.493988	5.484521	3.048168	0.644112	0.608701	0.856457	0.356807	0.171303	0.084296
CGPA	-6.491703	5.641944	2.981607	0.487761	0.426845	0.356807	0.365799	0.150655	0.075326
Research	-0.382766	3.162004	1.411303	0.242645	0.200962	0.171303	0.150655	0.246894	0.038282
Chance_of_Admit	0.173437	1.291862	0.680046	0.111384	0.095691	0.084296	0.075326	0.038282	0.019921

```
df.describe().T
```



	count	mean	std	min	25%	50%	75%	max
Serial No.	500.0	250.50000	144.481833	1.00	125.7500	250.50	375.25	500.00
GRE Score	500.0	316.47200	11.295148	290.00	308.0000	317.00	325.00	340.00
TOEFL Score	500.0	107.19200	6.081868	92.00	103.0000	107.00	112.00	120.00
University Rating	500.0	3.11400	1.143512	1.00	2.0000	3.00	4.00	5.00
SOP	500.0	3.37400	0.991004	1.00	2.5000	3.50	4.00	5.00
LOR	500.0	3.48400	0.925450	1.00	3.0000	3.50	4.00	5.00
CGPA	500.0	8.57644	0.604813	6.80	8.1275	8.56	9.04	9.92
Research	500.0	0.56000	0.496884	0.00	0.0000	1.00	1.00	1.00
Chance_of_Admit	500.0	0.72174	0.141140	0.34	0.6300	0.72	0.82	0.97



```
df = df.drop(columns='Serial No.')
```


```
df.sample()
```



GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance_of_Admit	
344	295	96	2	1.5	2.0	7.34	0	0.47




```
df.columns
```




```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',  
      'Research', 'Chance_of_Admit'],  
      dtype='object')
```

Duplicate Detection

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



GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance_of_Admit
-----------	-------------	-------------------	-----	-----	------	----------	-----------------




Insights

- The dataset does not contain any duplicates.

Null Detection

```
df.isna().any()
```



	0
GRE Score	False
TOEFL Score	False
University Rating	False
SOP	False
LOR	False
CGPA	False
Research	False
Chance_of_Admit	False

dtype: bool

```
df.isna().sum()
```

```

0
GRE Score    0
TOEFL Score  0
University Rating
SOP          0
LOR          0
CGPA         0
Research     0
Chance_of_Admit  0

dtype: int64

```

```

plt.figure(figsize=(25,8))

# Using a custom style with a dark background
plt.style.use('dark_background')

# Customize the colormap, add a linewidth for grid effect, and adjust the cbar aesthetics
sns.heatmap(df.isnull(), cmap='YlGn', cbar=True, cbar_kws={'shrink': 0.5, 'label': 'Missing Data'},
            linewidths=0.5, linecolor='black')

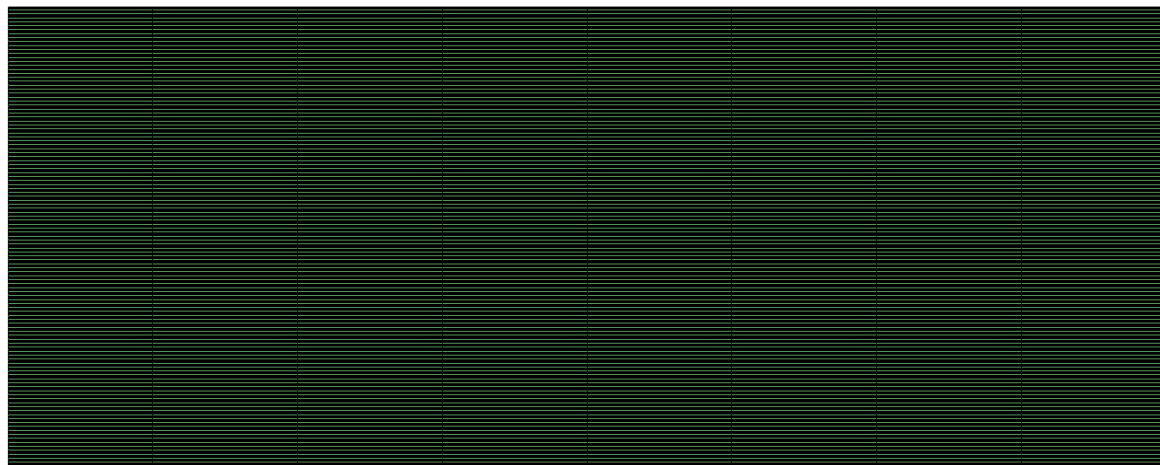
# Adding title with custom font, color, and positioning
plt.title('Visual Check of Nulls', fontsize=25, color='lime', pad=20, weight='bold', loc='center')

# Enhancing the x and y axis label fonts
plt.xticks(fontsize=12, color='white', rotation=45)
plt.yticks(fontsize=12, color='white')

plt.show()

```

Visual Check of Nulls



Insights:

- The dataset does not have missing values, as evidenced by the uniform color across the entire heatmap.

```

for _ in df.columns:
    print()
    print(f'Total Unique Values in {_} column are :- {df[_].nunique()}')
    print(f'Value counts in {_} column are :-\n {df[_].value_counts(normalize=True)}')
    print()
    print('-'*120)

```

```
LOR
3.0    0.198
4.0    0.188
3.5    0.172
4.5    0.126
2.5    0.100
5.0    0.100
2.0    0.092
1.5    0.022
1.0    0.002
Name: proportion, dtype: float64
```

```
Total Unique Values in CGPA column are :- 184
Value counts in CGPA column are :-
CGPA
8.76    0.018
8.00    0.018
8.12    0.014
8.45    0.014
8.54    0.014
...
9.92    0.002
9.35    0.002
8.71    0.002
9.32    0.002
7.69    0.002
Name: proportion, Length: 184, dtype: float64
```

```
Total Unique Values in Research column are :- 2
Value counts in Research column are :-
Research
1    0.56
0    0.44
Name: proportion, dtype: float64
```

```
Total Unique Values in Chance_of_Admit column are :- 61
Value counts in Chance_of_Admit column are :-
Chance_of_Admit
0.71    0.046
0.64    0.038
0.73    0.036
0.72    0.032
0.79    0.032
...
0.38    0.004
0.36    0.004
0.43    0.002
0.39    0.002
0.37    0.002
Name: proportion, Length: 61, dtype: float64
```

```
for _ in df.columns:
    print()
    print(f'Range of {_} column is from {df[_].min()} to {df[_].max()}')
    print()
    print('-'*120)
```

```
Range of GRE Score column is from 290 to 340
```

```
Range of TOEFL Score column is from 92 to 120
```

```
Range of University Rating column is from 1 to 5
```

```
Range of SOP column is from 1.0 to 5.0
```

```
Range of LOR column is from 1.0 to 5.0
```


```
Range of CGPA column is from 6.8 to 9.92
```

Range of Research column is from 0 to 1

Range of Chance_of_Admit column is from 0.34 to 0.97

```
df['Research'] = df.Research.astype('category')
```

```
df.dtypes
```



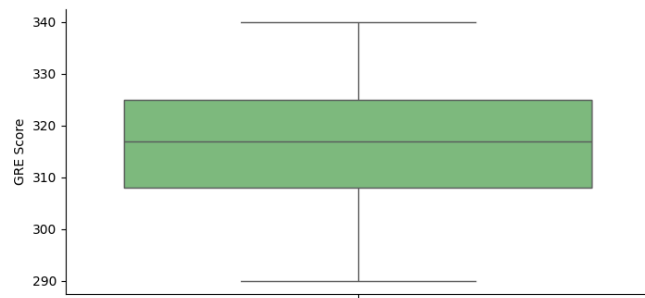
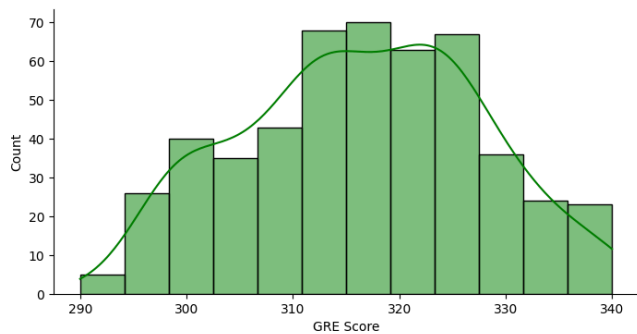
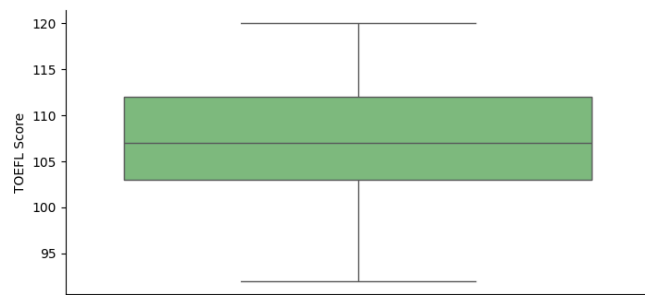
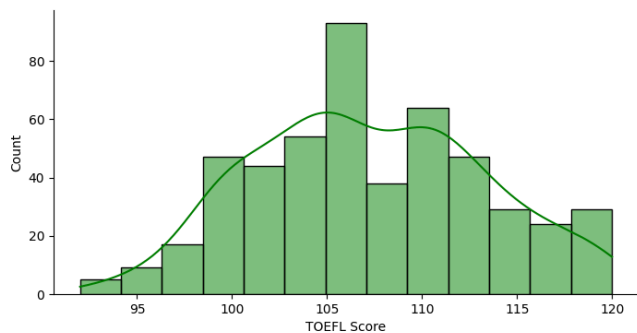
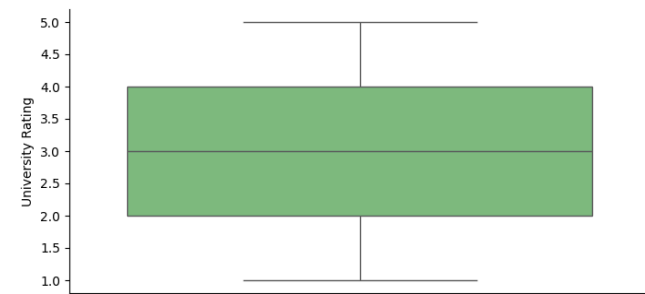
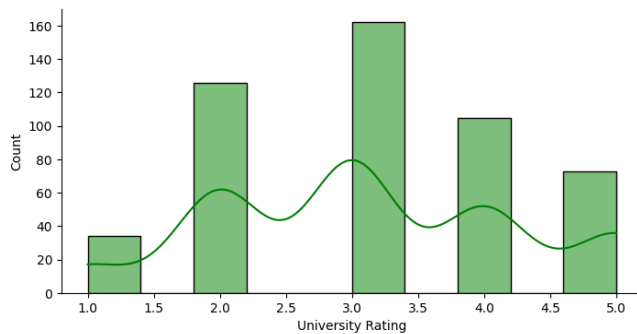
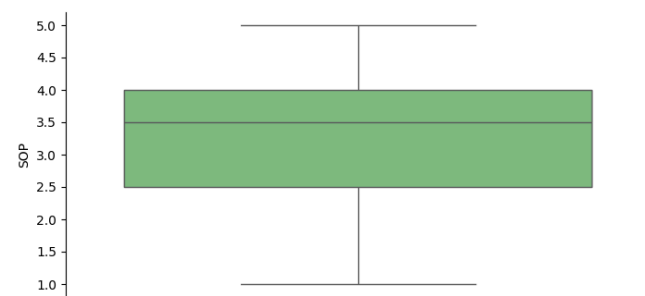
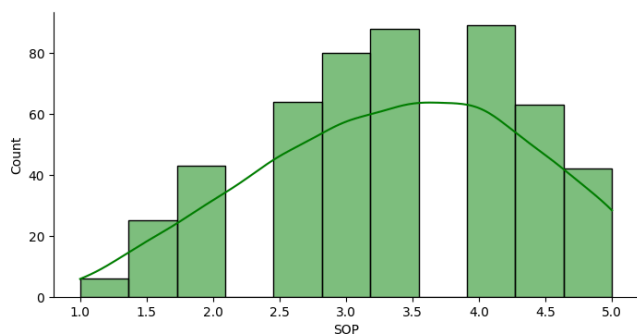
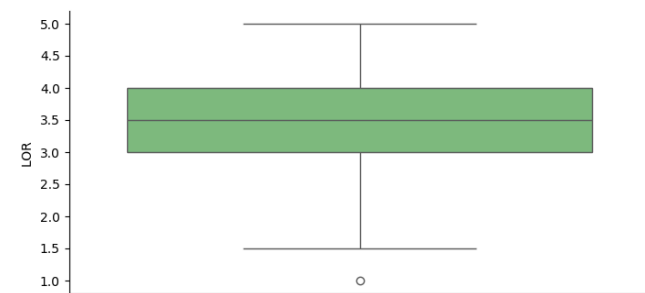
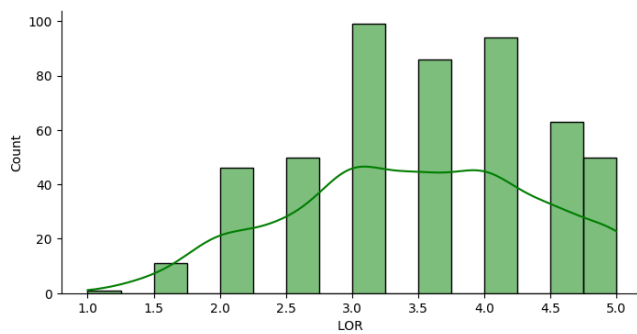
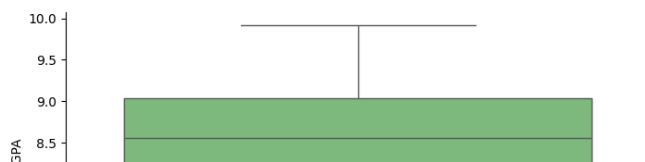
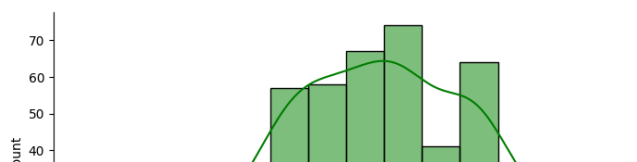
	0
GRE Score	int64
TOEFL Score	int64
University Rating	int64
SOP	float64
LOR	float64
CGPA	float64
Research	category
Chance_of_Admit	float64

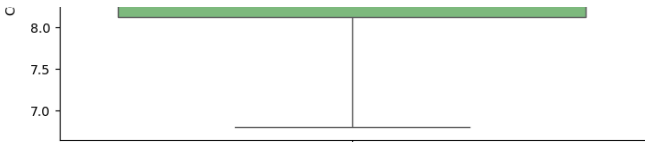
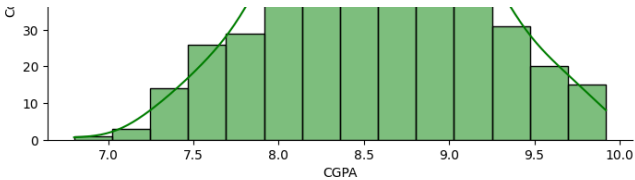
dtype: object

Graphical Analysis:

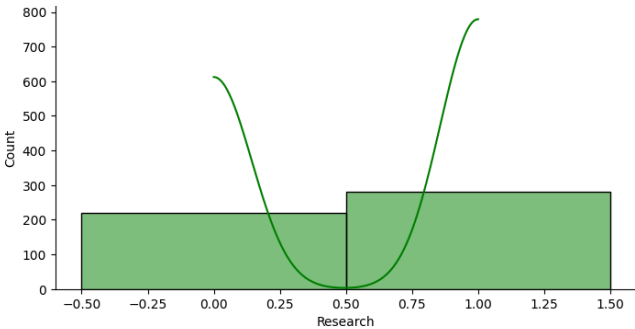
```
cp = 'Greens'

for _ in df.columns:
    plt.style.use('default')
    plt.style.use('seaborn-bright')
    plt.figure(figsize = (18,4))
    plt.subplot(122)
    sns.boxplot(df[_],palette=cp)
    plt.subplot(121)
    sns.histplot(df[_],kde=True,color='g')
    plt.suptitle(f'Plot of {_}', fontsize=14,fontfamily='serif',fontweight='bold',backgroundcolor='g',color='w')
    sns.despine()
    plt.show()
```

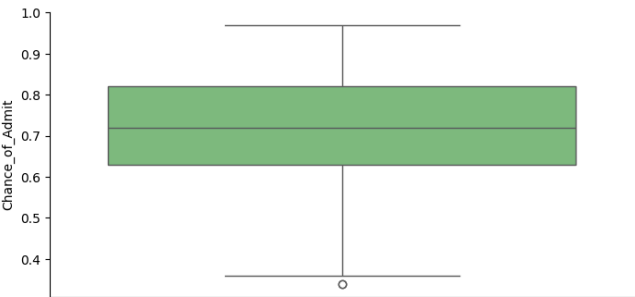
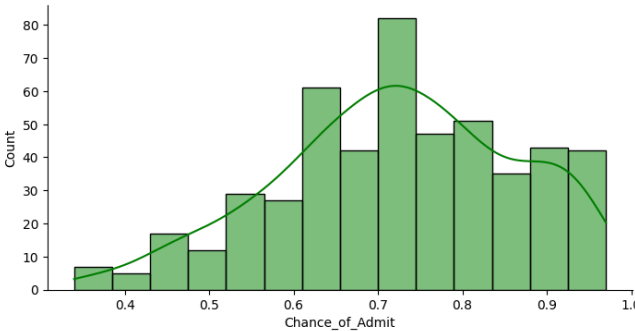
**Plot of GRE Score****Plot of TOEFL Score****Plot of University Rating****Plot of SOP****Plot of LOR****Plot of CGPA**



Plot of Research



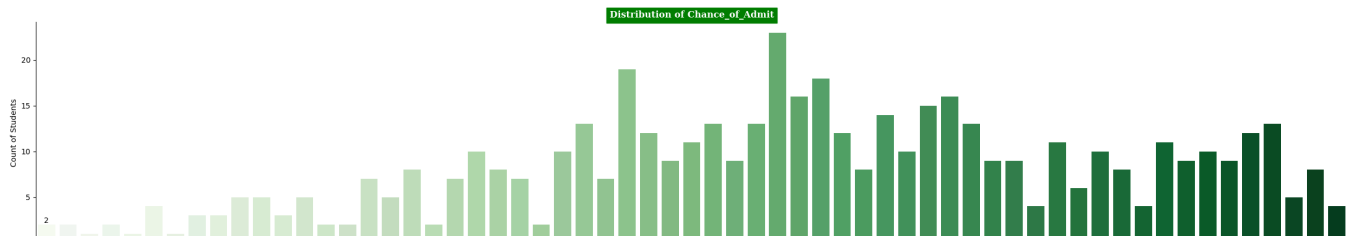
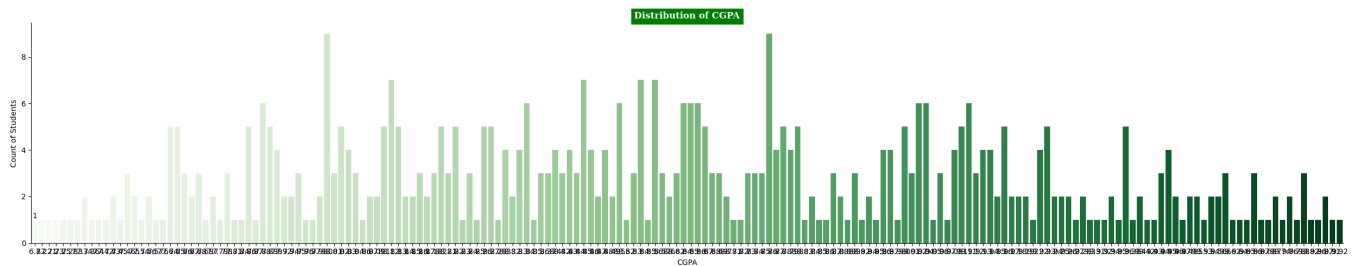
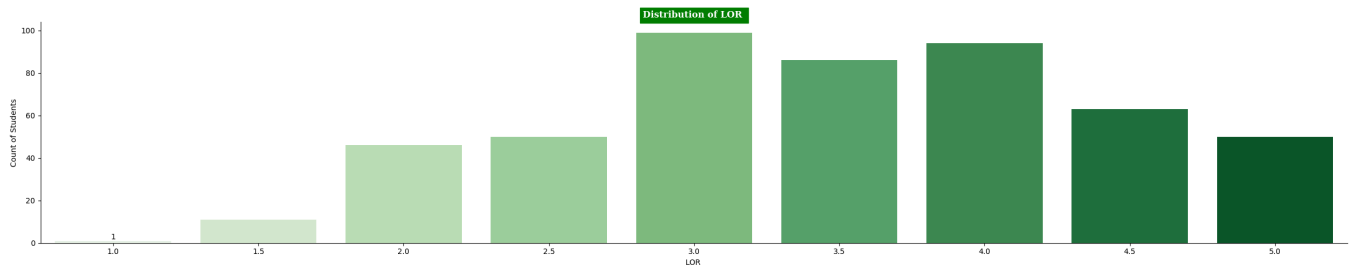
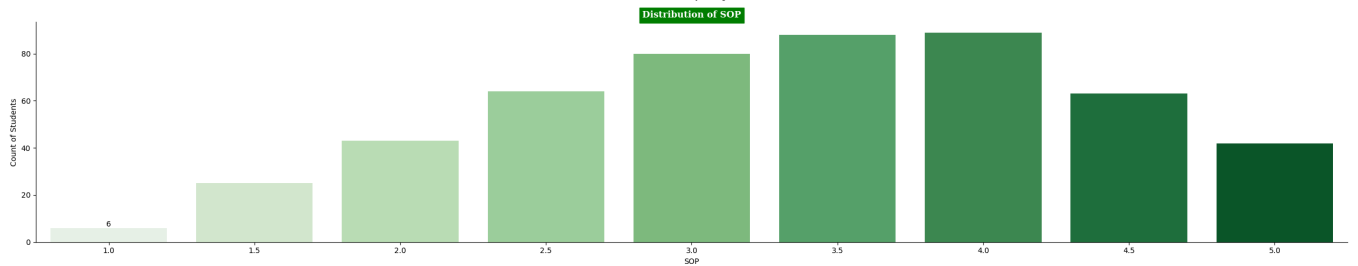
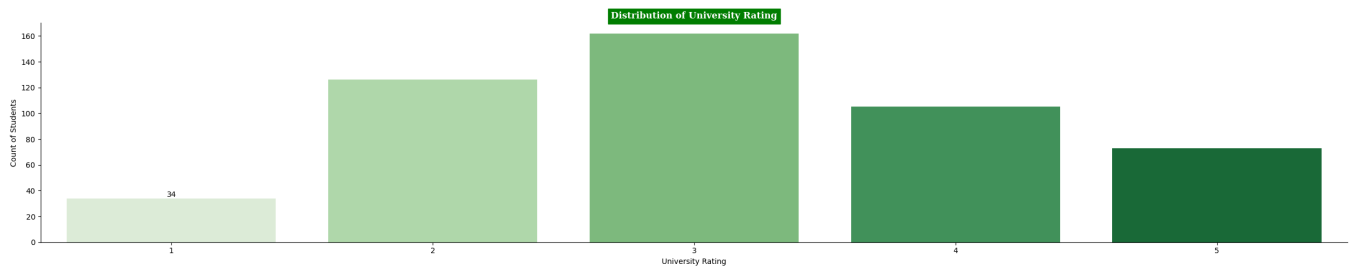
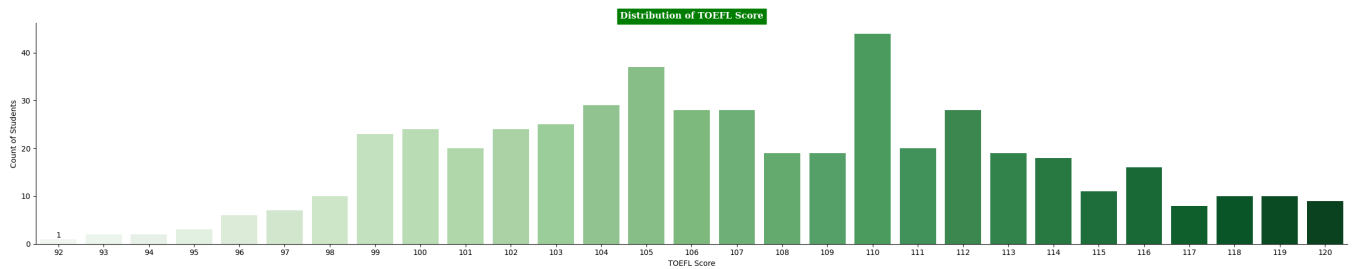
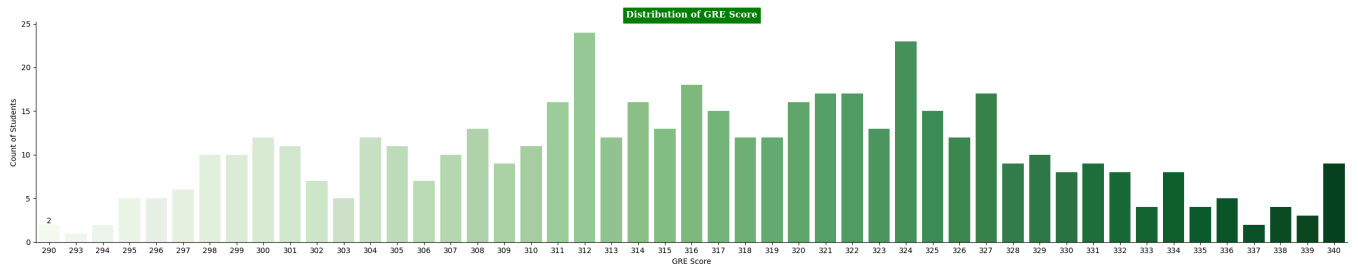
Plot of Chance_of_Admit



✓ Insights:

- Other than LOR there no outliers found in other features. And there is no need for treating LOR as it is one of the ratings given on scale 0-5.

```
for col in df.columns:
    plt.figure(figsize=(25,5))
    plt.style.use('default')
    plt.style.use('seaborn-bright')
    b = sns.countplot(x=df[col],palette=cp)
    plt.title(f'Distribution of {col}',fontsize=12,fontfamily='serif',fontweight='bold',backgroundcolor='g',color='w')
    b.bar_label(b.containers[0], label_type='edge',fmt='%d')
    plt.xlabel(col)
    plt.ylabel('Count of Students')
    plt.tight_layout()
    sns.despine()
    plt.show();
```

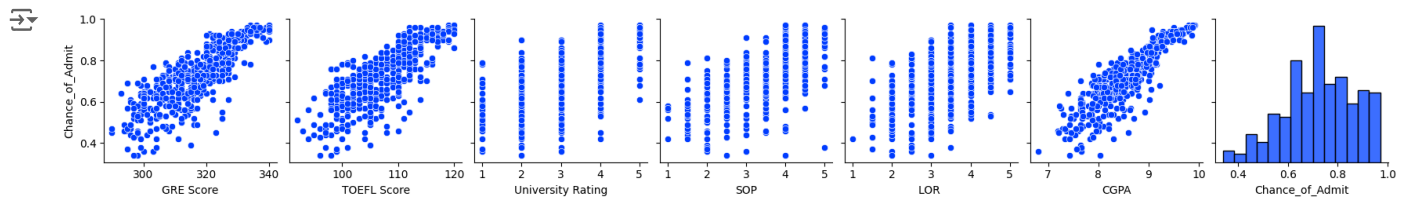




```
df.columns
```

```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',  
      'Research', 'Chance_of_Admit'],  
      dtype='object')
```

```
sns.pairplot(data=df, y_vars='Chance_of_Admit')  
plt.show()
```

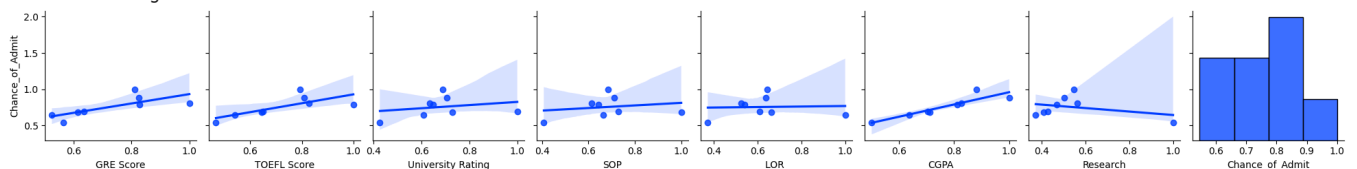


Insights:

- Exam scores (GRE, TOEFL and CGPA) have a high positive correlation with chance of admit
- While university ranking, rating of SOP and LOR also have an impact on chances of admit, research is the only variable which doesn't have much of an impact
- We can see from the scatterplot that the values of university ranking, SOP, LOR and research are not continuous. We can convert these columns to categorical variables

```
sns.pairplot(df.corr(), y_vars='Chance_of_Admit', kind='reg')
```

```
<seaborn.axisgrid.PairGrid at 0x7e118845d360>
```



```
df.columns
```

```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',  
      'Research', 'Chance_of_Admit'],  
      dtype='object')
```

```
# Check for categorical columns  
categorical_columns = df.select_dtypes(include=['category']).columns.tolist()  
print("Categorical columns in the DataFrame:", categorical_columns)
```

```
# Loop through all columns except the last one ("Chance_of_Admit")
for col in df.columns[:-1]:
    # Skip categorical columns that cannot be used for regression
    if col in categorical_columns:
        print(f"Skipping categorical column: {col}")
        continue

    print(f"Generating plot for: {col}")

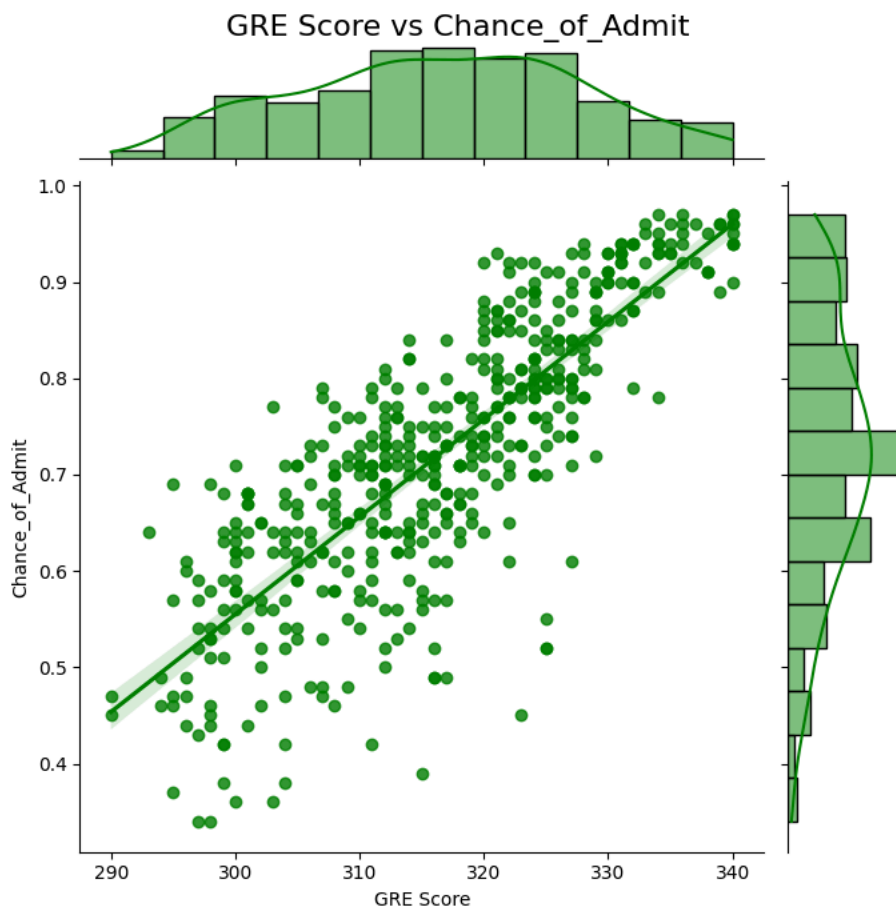
    # Create jointplot with regression line
    g = sns.jointplot(data=df, x=col, y="Chance_of_Admit", kind="reg", color='g', height=7)

    # Add a title to each plot
    g.fig.suptitle(f'{col} vs Chance_of_Admit', fontsize=16, color='black')

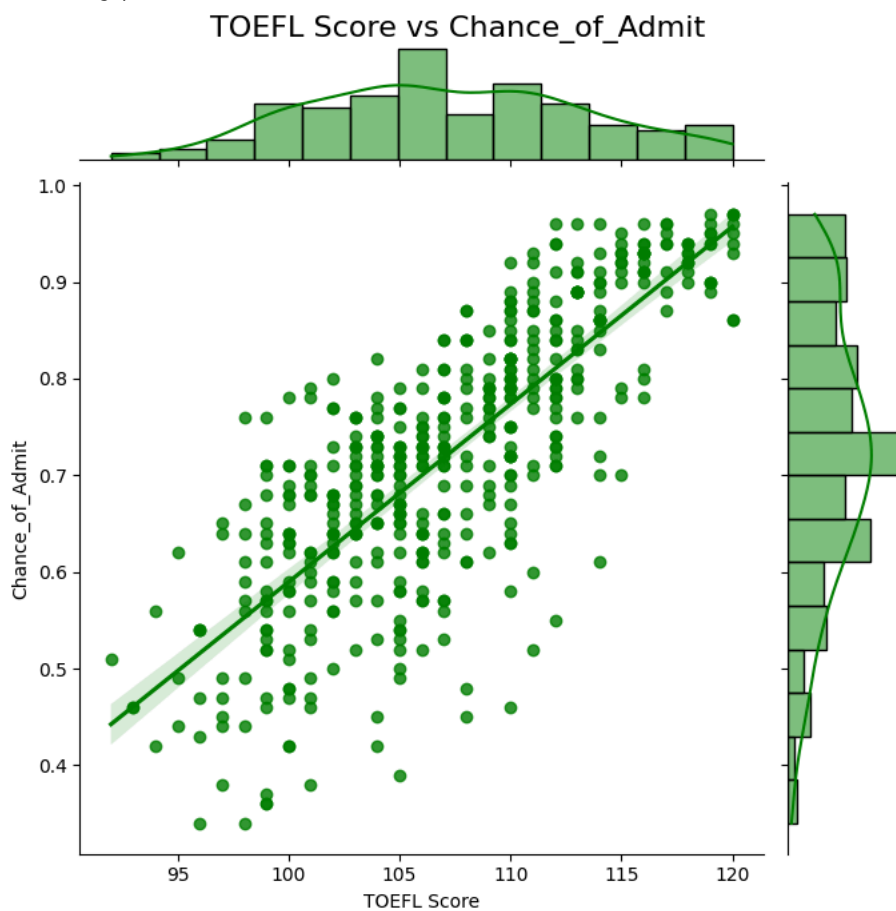
    # Adjust the title position to not overlap with the plot
    g.fig.subplots_adjust(top=0.95)

plt.show()
```

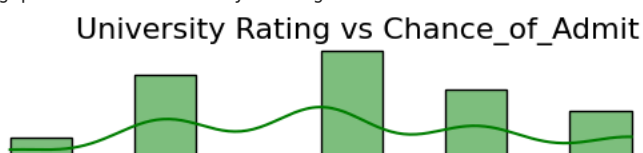
Generating plot for: GRE Score

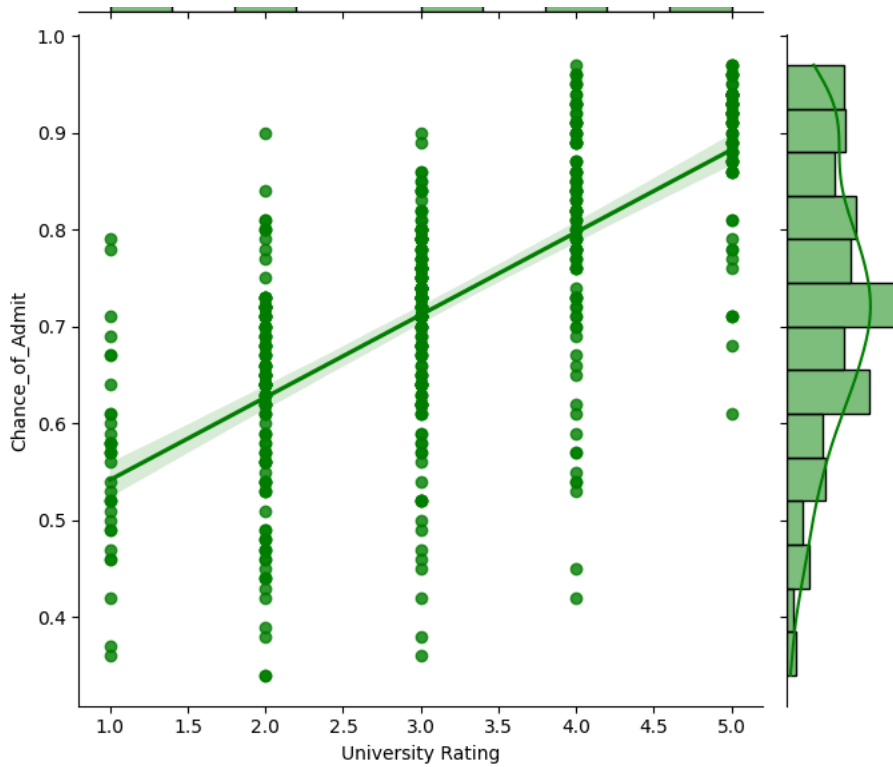


Generating plot for: TOEFL Score

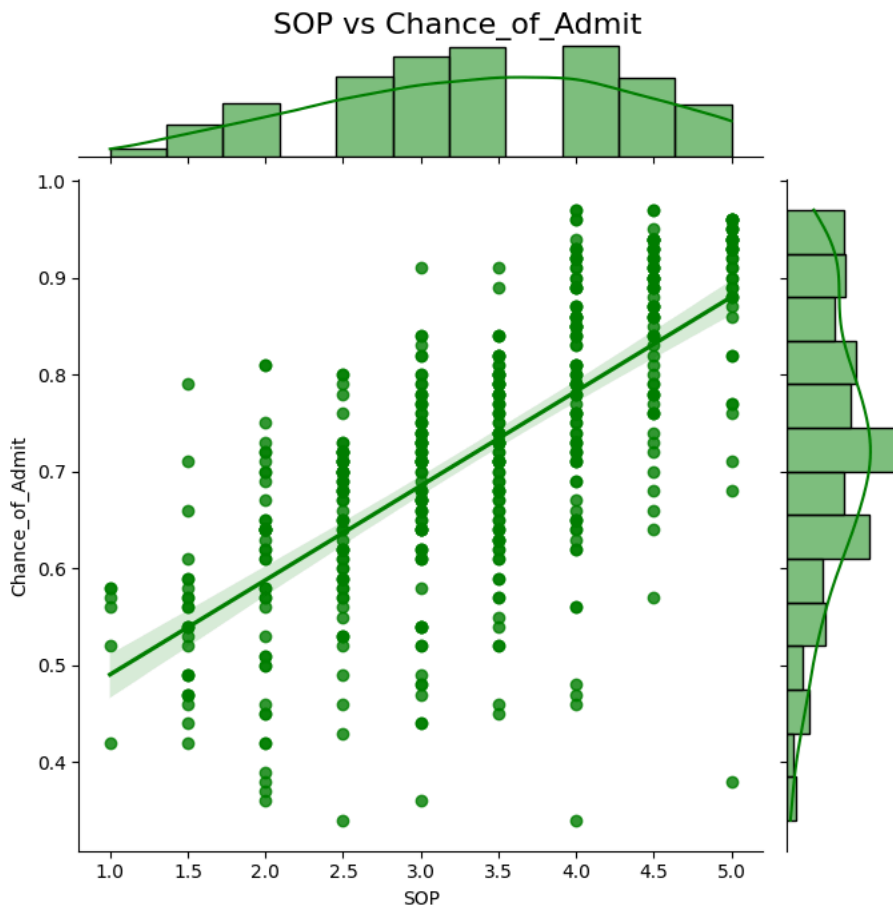


Generating plot for: University Rating

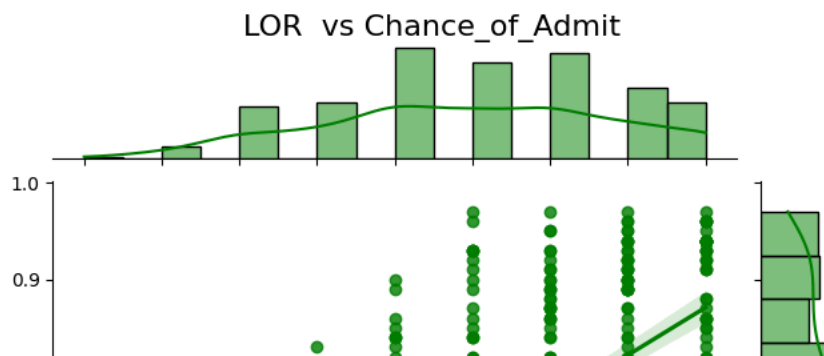


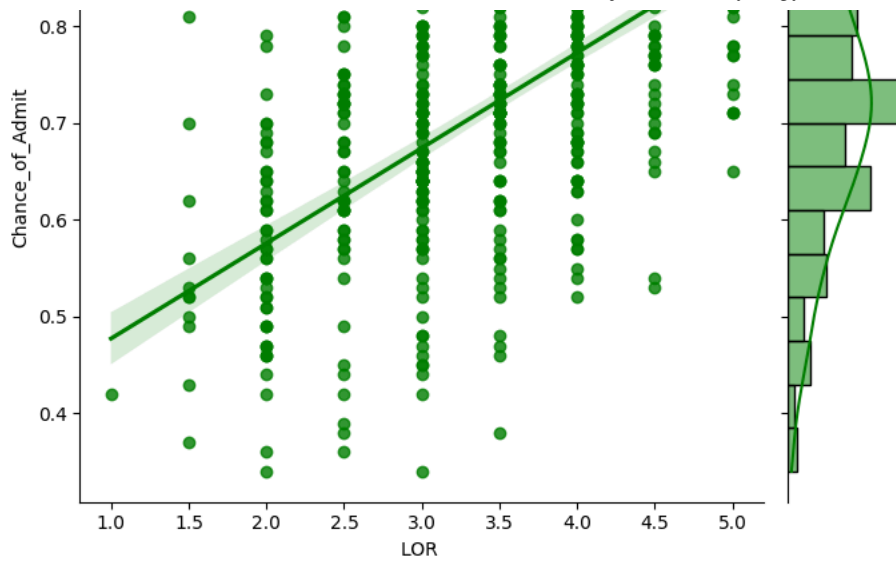


Generating plot for: SOP

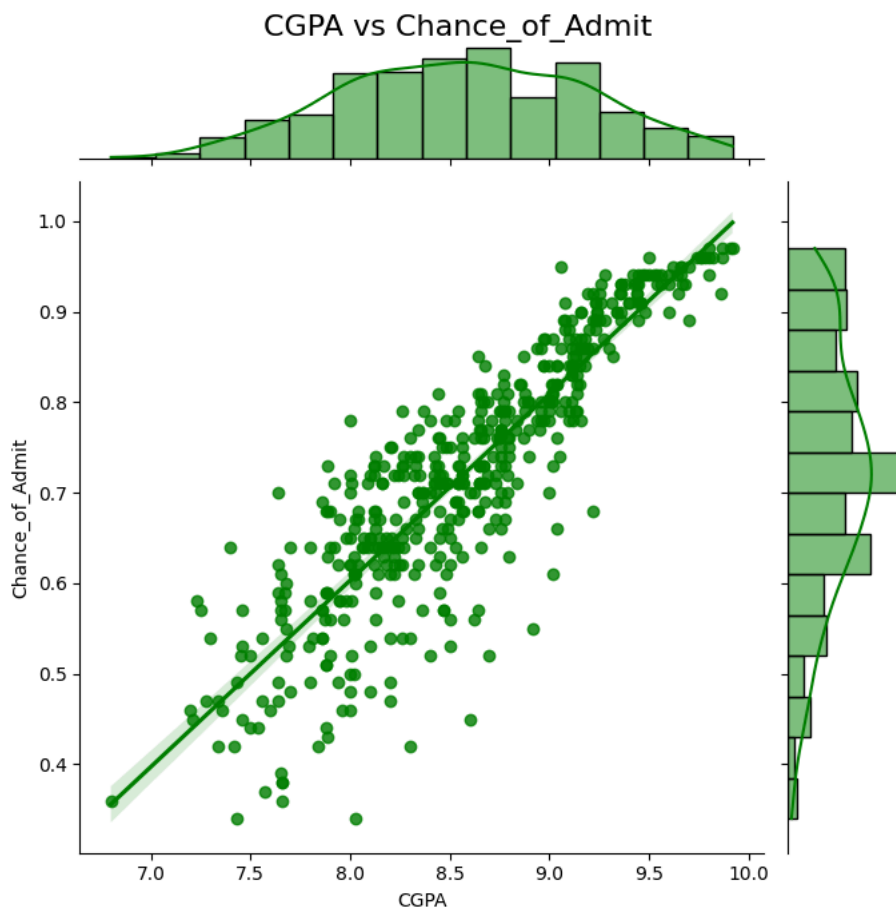


Generating plot for: LOR





Generating plot for: CGPA



Skipping categorical column: Research

Insights:

- with higher GRE score , there is high probability of getting an admission.
- Students having high toefl score , has higher probability of getting admission .

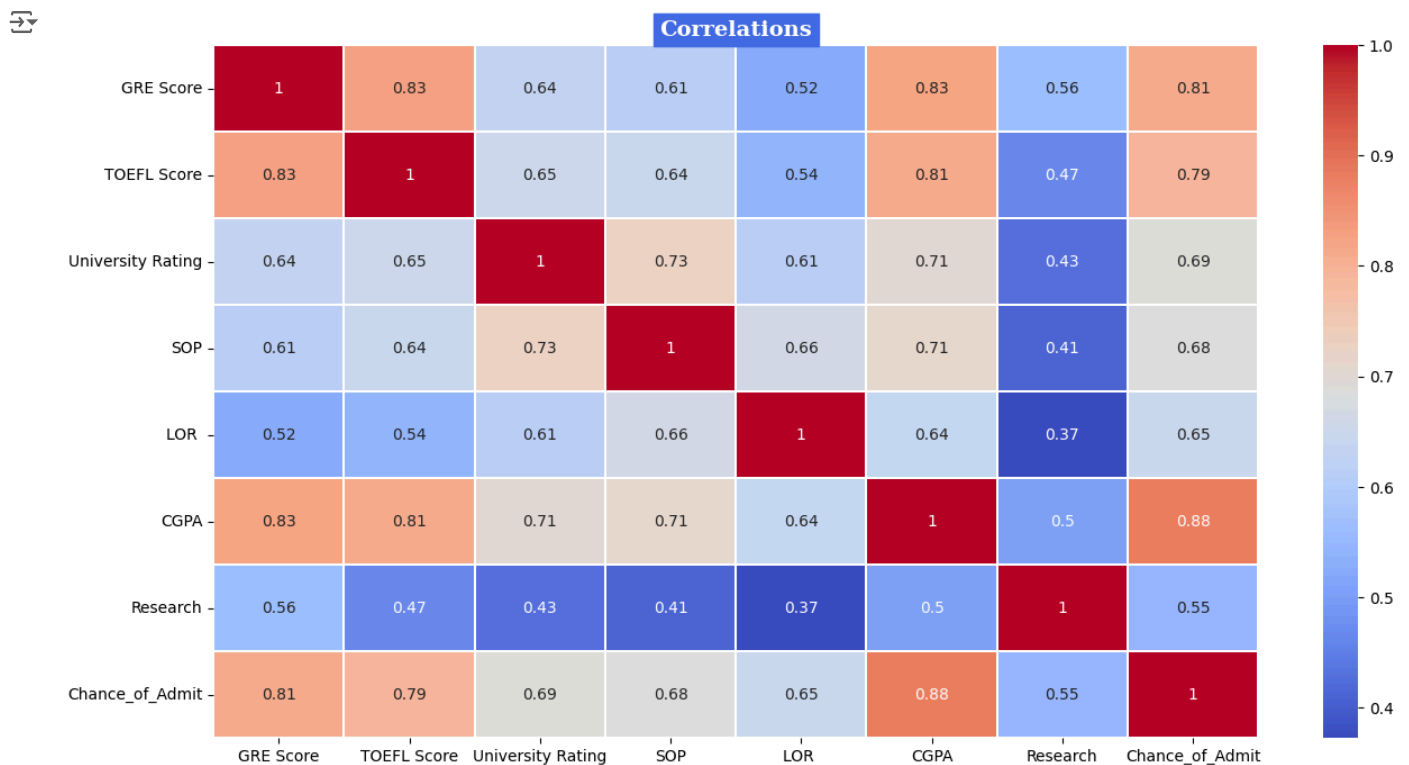
```
plt.figure(figsize=(15, 8))
```

```
# Change cmap to a more appealing option (e.g., 'coolwarm')
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', linewidths=0.1)
```

```
# Set title with improved aesthetics
plt.title('Correlations', fontsize=14, fontfamily='serif', fontweight='bold', backgroundcolor='royalblue', color='white')
```

```
# Adjust y-ticks
plt.yticks(rotation=0)
```

```
# Show the plot
plt.show()
```



Data Preprocessing - Standardization!

```
scaler = StandardScaler()
scaled_df = pd.DataFrame(scaler.fit_transform(df), columns = df.columns)
```

```
scaled_df
```


	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance_of_Admit	
0	1.819238	1.778865	0.775582	1.137360	1.098944	1.776806	0.886405	1.406107	
1	0.667148	-0.031601	0.775582	0.632315	1.098944	0.485859	0.886405	0.271349	
2	-0.041830	-0.525364	-0.099793	-0.377773	0.017306	-0.954043	0.886405	-0.012340	
3	0.489904	0.462163	-0.099793	0.127271	-1.064332	0.154847	0.886405	0.555039	
4	-0.219074	-0.689952	-0.975168	-1.387862	-0.523513	-0.606480	-1.128152	-0.508797	
...	
495	1.376126	0.132987	1.650957	1.137360	0.558125	0.734118	0.886405	1.051495	
496	1.819238	1.614278	1.650957	1.642404	1.639763	2.140919	0.886405	1.689797	
497	1.198882	2.108041	1.650957	1.137360	1.639763	1.627851	0.886405	1.477030	
498	-0.396319	-0.689952	0.775582	0.632315	1.639763	-0.242367	-1.128152	0.058582	
499	0.933015	0.955926	0.775582	1.137360	1.098944	0.767220	-1.128152	0.838728	

500 rows × 8 columns

Next steps:

[Generate code with scaled_df](#)[View recommended plots](#)[New interactive sheet](#)

✓ Train-Test data split

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

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

```
# Split the data into training and test data
```

```
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()
```

```
# Predicting values for the training and test data
y_pred_train = lr_model.predict(x_train)
y_pred_test = lr_model.predict(x_test)
```

✓ r2 score on train data :

```
r2_score(y_train, y_pred_train)
```

```
0.8210671369321554
```

```
lr_model.score(x_train, y_train)
```

```
0.8210671369321554
```

✓ r2 score on test data :

```
r2_score(y_test,y_pred_test)
```

```
0.8188432567829628
```

```
lr_model.score(x_test,y_test)
```

```
0.8188432567829628
```

✓ All the feature's coefficients and Intercept :

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

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Intercept
0	0.194823	0.129095	0.020812	0.012735	0.113028	0.482199	0.084586	0.007736

✓ Insights:

- CGPA,GRE,TOEFL scores have the highest weight
- SOP, University rating, and research have the lowest weights
- W0 - intercept is low

```
def model_evaluation(y_actual, y_forecast, model):
    n = len(y_actual)
    if len(model.coef_.shape)==1:
        p = len(model.coef_)
    else:
        p = len(model.coef_[0])
    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)
    #rsme = np.sqrt(mean_squared_error(y_test,y_pred))
    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)
```

```
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
```

Insights:

- Since there is No difference in the loss scores of training and test data, we can conclude that there is NO overfitting of the model.

✓ Linear Regression using OLS :

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

```
# statistical 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:	Sat, 12 Oct 2024	Prob (F-statistic):	3.41e-142
Time:	04:27:39	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.

💡 Considering the very low p_valued Features and highly weighted coef features as the major contributors of Model Prediction, CGPA, GRE, TOEFL, LOR are the features contributing to model building...

✓ 📁 Testing Assumptions of Linear Regression Model :

1. No multicollinearity:

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.
- 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.

5. Normality of Residuals

- Almost bell-shaped curve in residuals distribution.

6. Impact of Outliers

✓ 💎 Multicollinearity 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 j th predictor variable.
- $R(j)^2$ is the coefficient of determination (R-squared) obtained from regressing the j th 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	

Next steps:

[Generate code with vif](#)
[View recommended plots](#)
[New interactive sheet](#)

Insights:

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

◆ Mean of Residuals:

- The mean of residuals represents the average of residual values in a regression model.
- Residuals are the discrepancies or errors between the observed values and the values predicted by the regression model.
- The mean of residuals is useful to assess the overall bias in the regression model. If the mean of residuals is close to zero, it indicates that the model is unbiased on average.
- However, if the mean of residuals is significantly different from zero, it suggests that the model is systematically overestimating or underestimating the observed values.
- **The mean of residuals being close to zero indicates that, on average, the predictions made by the linear regression model are accurate, with an equal balance of overestimations and underestimations. This is a desirable characteristic of a well-fitted regression model.**

```
residuals = y_test.values - y_pred_test
```

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

```
9.436895709313831e-18
```

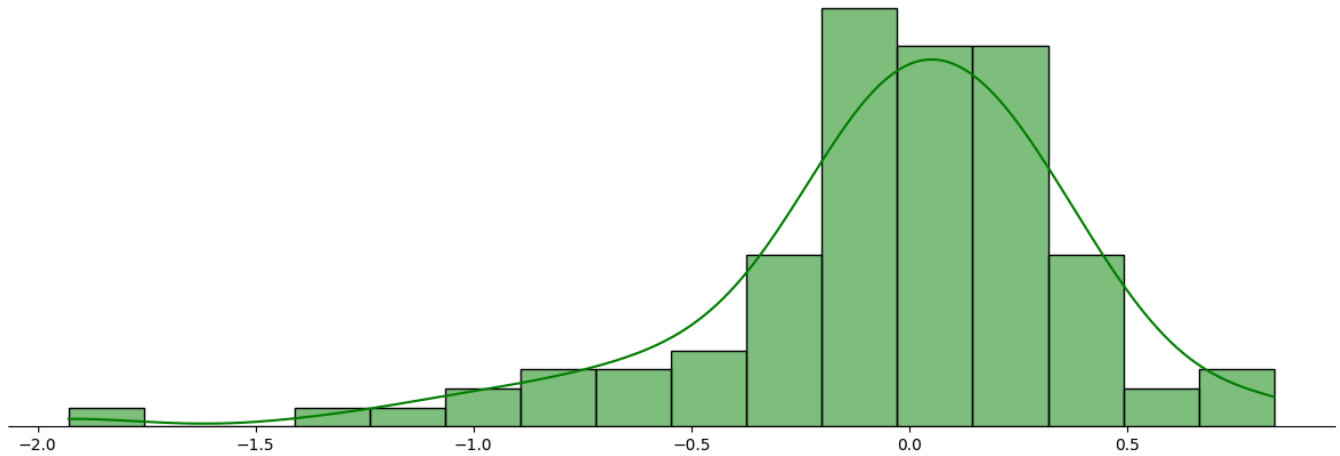
```
residuals.mean()
```

```
-0.03867840379282768
```

```
plt.figure(figsize=(15,5))
sns.histplot(residuals, kde= True,color='g')
plt.title('Residuals Hist',fontsize=12,fontfamily='serif',fontweight='bold',backgroundcolor='mediumseagreen',color='w')
sns.despine(left=True)
plt.ylabel('')
plt.yticks([])
plt.show()
```



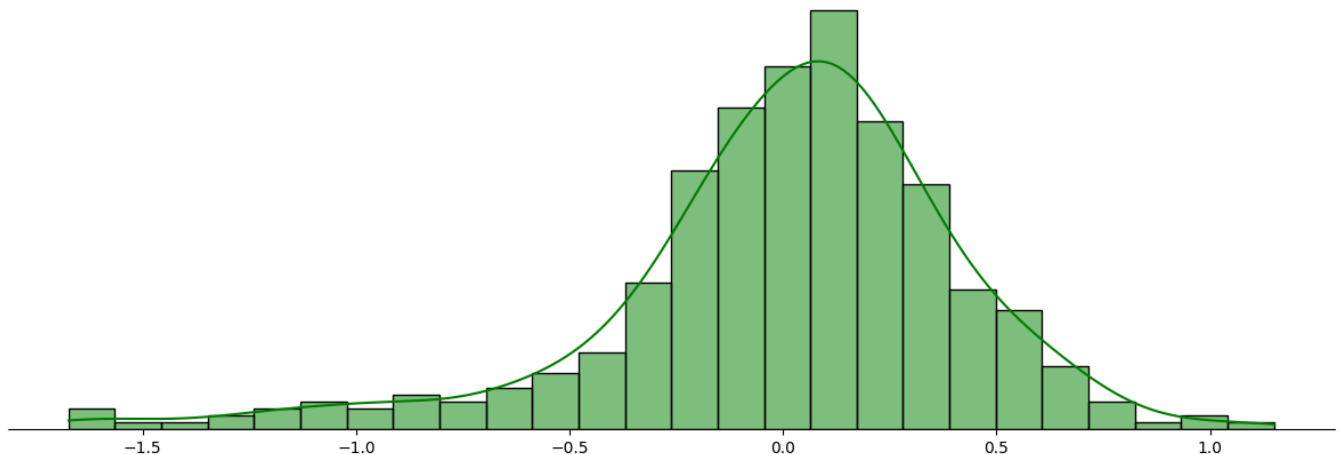
Residuals Hist



```
plt.figure(figsize=(15,5))
sns.histplot(residuals_train, kde= True,color='g')
plt.title('Residuals Hist',fontsize=12,fontfamily='serif',fontweight='bold',backgroundcolor='mediumseagreen',color='w')
sns.despine(left=True)
plt.ylabel('')
plt.yticks([])
plt.show()
```



Residuals Hist



Insights:

- Since the mean of residuals is very close to 0, we can say that the model is UnBiased.

Linear Relationships:

Linearity of variables refers to the assumption that there is a linear relationship between the independent variables and the dependent variable in a regression model. It means that the effect of the independent variables on the dependent variable is constant across different levels of the independent variables.

When we talk about "no pattern in the residual plot" in the context of linearity, we are referring to the plot of the residuals (the differences between the observed and predicted values of the dependent variable) against the predicted values or the independent variables.

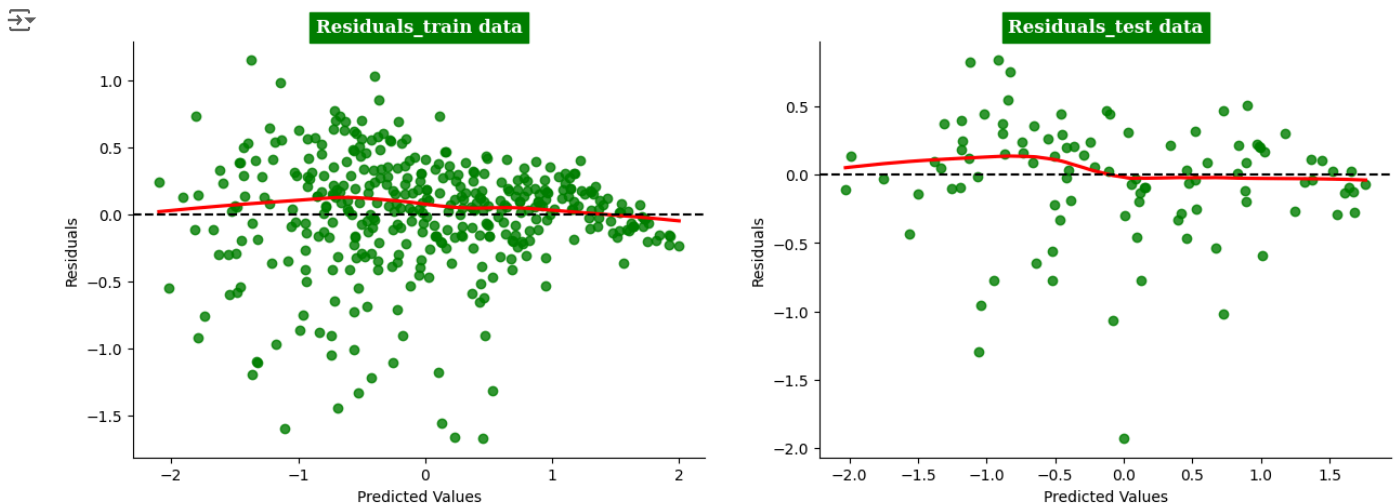
Ideally, in a linear regression model, the residuals should be randomly scattered around zero, without any clear patterns or trends. This indicates that the model captures the linear relationships well and the assumption of linearity is met.

If there is a visible pattern in the residual plot, it suggests a violation of the linearity assumption. Common patterns that indicate non-linearity include:

1. Curved or nonlinear shape: The residuals form a curved or nonlinear pattern instead of a straight line.
2. U-shaped or inverted U-shaped pattern: The residuals show a U-shape or inverted U-shape, indicating a nonlinear relationship.
3. Funnel-shaped pattern: The spread of residuals widens or narrows as the predicted values or independent variables change, suggesting heteroscedasticity.
4. Clustering or uneven spread: The residuals show clustering or uneven spread across different levels of the predicted values or independent variables.

If a pattern is observed in the residual plot, it may indicate that the linear regression model is not appropriate, and nonlinear regression or other modeling techniques should be considered. Additionally, transformations of variables, adding interaction terms, or using polynomial terms can sometimes help capture nonlinear relationships and improve linearity in the residual plot.

```
plt.figure(figsize=(15,5))
plt.subplot(121)
plt.title('Residuals_train data',fontsize=12,fontfamily='serif',fontweight='bold',backgroundcolor='g',color='w')
sns.regplot(x=y_pred_train, y=residuals_train, lowess=True, color='g',line_kws={'color': 'red'})
plt.axhline(y=0, color='k', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.subplot(122)
plt.title('Residuals_test data',fontsize=12,fontfamily='serif',fontweight='bold',backgroundcolor='g',color='w')
sns.regplot(x=y_pred_test, y=residuals, lowess=True,color='g' ,line_kws={'color': 'red'})
plt.axhline(y=0, color='k', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
sns.despine()
plt.show()
```



Insights:

- From the Joint plot & pairplot in the graphical analysis, we can say that there is linear relationship between dependent variable and independent variables.
- As we can observe, GRE Score, TOEFL Score and CGPA have a linear relationship with the Chance of Admit. Although GRE score and TOEFL score are more scattered, CGPA has a much more more linear relationship with the Chance of Admit.
- In a linear regression model, the residuals are randomly scattered around zero, without any clear patterns or trends. This indicates that the model captures the linear relationships well and the assumption of linearity is met.

◆ Homoscedasticity :

Homoscedasticity refers to the assumption in regression analysis that the variance of the residuals (or errors) should be constant across all levels of the independent variables. In simpler terms, it means that the spread of the residuals should be similar across different values of the predictors.

When homoscedasticity is violated, it indicates that the variability of the errors is not consistent across the range of the predictors, which can lead to unreliable and biased regression estimates.

To test for homoscedasticity, there are several graphical and statistical methods that you can use:

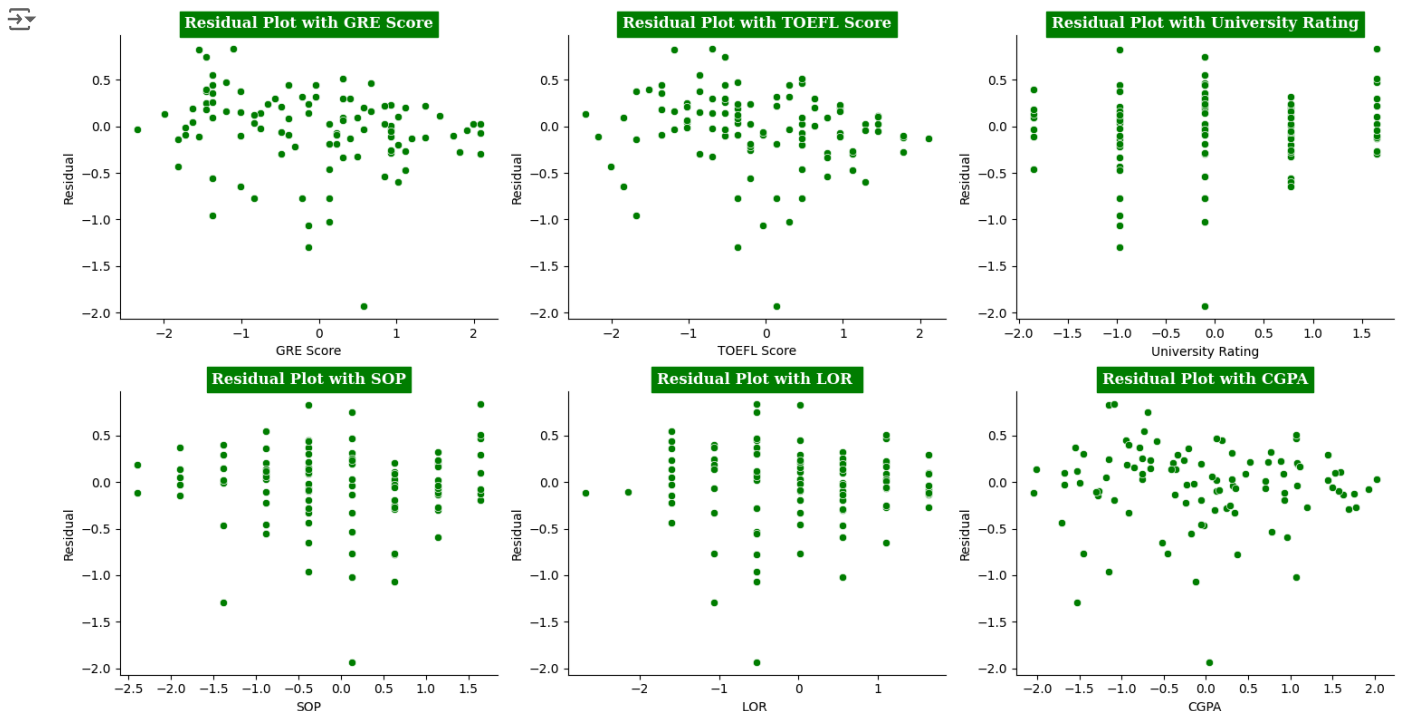
1. Residual plot: Plot the residuals against the predicted values or the independent variables. Look for any systematic patterns or trends in the spread of the residuals. If the spread appears to be consistent across all levels of the predictors, then homoscedasticity is likely met.
2. Scatterplot: If you have multiple independent variables, you can create scatter plots of the residuals against each independent variable separately. Again, look for any patterns or trends in the spread of the residuals.
3. Breusch-Pagan Test: This is a statistical test for homoscedasticity. It involves regressing the squared residuals on the independent variables and checking the significance of the resulting model. If the p-value is greater than a chosen significance level (e.g., 0.05), it suggests homoscedasticity. However, this test assumes that the errors follow a normal distribution.
4. Goldfeld-Quandt Test: This test is used when you suspect heteroscedasticity due to different variances in different parts of the data. It involves splitting the data into two subsets based on a specific criterion and then comparing the variances of the residuals in each subset. If the difference in variances is not significant, it suggests homoscedasticity.

It's important to note that the visual inspection of plots is often the first step to identify potential violations of homoscedasticity. Statistical tests can provide additional evidence, but they may have assumptions or limitations that need to be considered.

✓ Scatterplot of residuals with each independent variable to check for Homoscedasticity

```
# 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=residuals.reshape((-1,)),color='g')
    plt.title(f'Residual Plot with {col}',fontsize=12,fontfamily='serif',fontweight='bold',backgroundcolor='g',color='w')
    plt.xlabel(col)
    plt.ylabel('Residual')
    i+=1

plt.tight_layout()
sns.despine()
plt.show();
```



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

✓ Breusch-Pagan test for Homoscedasticity

Null Hypothesis – H_0 : Homoscedasticity is present in residuals.

Alternate Hypothesis – H_a : Heteroscedasticity is present in residuals.

alpha : 0.05

```
bp_test = pd.DataFrame(sms.het_breuschpagan(residuals, ols_model.model.exog),
                      columns=['value'],
                      index=['Lagrange multiplier statistic', 'p-value', 'f-value', 'f p-value'])
```

bp_test

	value
Lagrange multiplier statistic	25.155866
p-value	0.000712
f-value	3.758171
f p-value	0.000588

Next steps:

[Generate code with bp_test](#)[View recommended plots](#)[New interactive sheet](#)

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.

It's important to consider alternative modeling approaches or corrective measures to address this issue.

Normality of Residuals:

Normality of residuals refers to the assumption that the residuals (or errors) in a statistical model are normally distributed. Residuals are the differences between the observed values and the predicted values from the model.

The assumption of normality is important in many statistical analyses because it allows for the application of certain statistical tests and the validity of confidence intervals and hypothesis tests. When residuals are normally distributed, it implies that the errors are random, unbiased, and have consistent variability.

To check for the normality of residuals, you can follow these steps:

Residual Histogram: Create a histogram of the residuals and visually inspect whether the shape of the histogram resembles a bell-shaped curve. If the majority of the residuals are clustered around the mean with a symmetric distribution, it suggests normality.

Q-Q Plot (Quantile-Quantile Plot): This plot compares the quantiles of the residuals against the quantiles of a theoretical normal distribution. If the points in the Q-Q plot are reasonably close to the diagonal line, it indicates that the residuals are normally distributed. Deviations from the line may suggest departures from normality.

Shapiro-Wilk Test: This is a statistical test that checks the null hypothesis that the residuals are normally distributed. The Shapiro-Wilk test calculates a test statistic and provides a p-value. If the p-value is greater than the chosen significance level (e.g., 0.05), it suggests that the residuals follow a normal distribution. However, this test may not be reliable for large sample sizes.

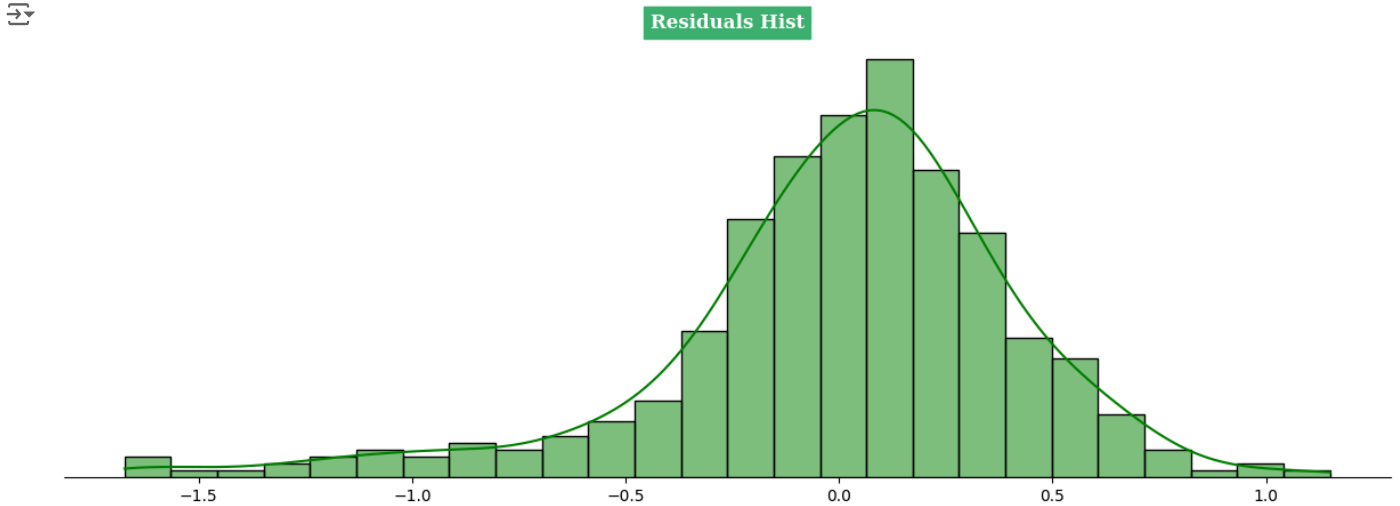
Anderson-Darling or Jarque-Bera can also be done as data size increases.

Skewness and Kurtosis: Calculate the skewness and kurtosis of the residuals. Skewness measures the asymmetry of the distribution, and a value close to zero suggests normality. Kurtosis measures the heaviness of the tails of the distribution compared to a normal distribution, and a value close to zero suggests similar tail behavior.

```
plt.figure(figsize=(15,5))
sns.histplot(residuals, kde= True,color='g')
plt.title('Residuals Hist',fontsize=12,fontfamily='serif',fontweight='bold',backgroundcolor='mediumseagreen',color='w')
sns.despine(left=True)
```

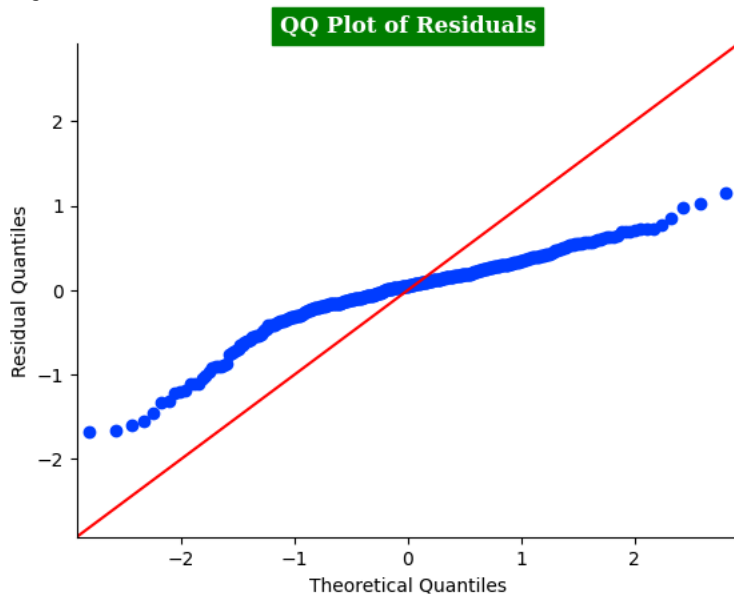


```
plt.ylabel("")
plt.yticks([])
plt.show()
```



```
# QQ-Plot of residuals
plt.figure(figsize=(15,5))
sm.qqplot(residuals,line='45')
plt.title('QQ Plot of Residuals',fontsize=12,fontfamily='serif',fontweight='bold',backgroundcolor='g',color='w')
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(residuals)
```

```
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: 190.09887364276915
p-value: 5.25477446043155e-42
Reject the null hypothesis: Residuals are not normally distributed.
```