```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
```

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

Next steps: Generate code with df View recommended plots New interactive sheet

Problem Statement

Develop a predictive model that estimates the "Chance of Admit" (a continuous value between 0 and 1) based on various student profile attributes such as GRE score, TOEFL score, GPA, university rating, SOP/LOR strength, and research experience.

```
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
```

In the data sets there are no null values present. Total 500 row and 9 columns. Also, all columns are numerical either int or float data type. In total 5 columns are integer data type and 4 are float data type

```
df.describe()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.00000
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.72174
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.14114
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.34000
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.63000
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.72000
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.82000
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.97000

This is statical summary of data base

```
df.shape
```

```
(500, 9)
```

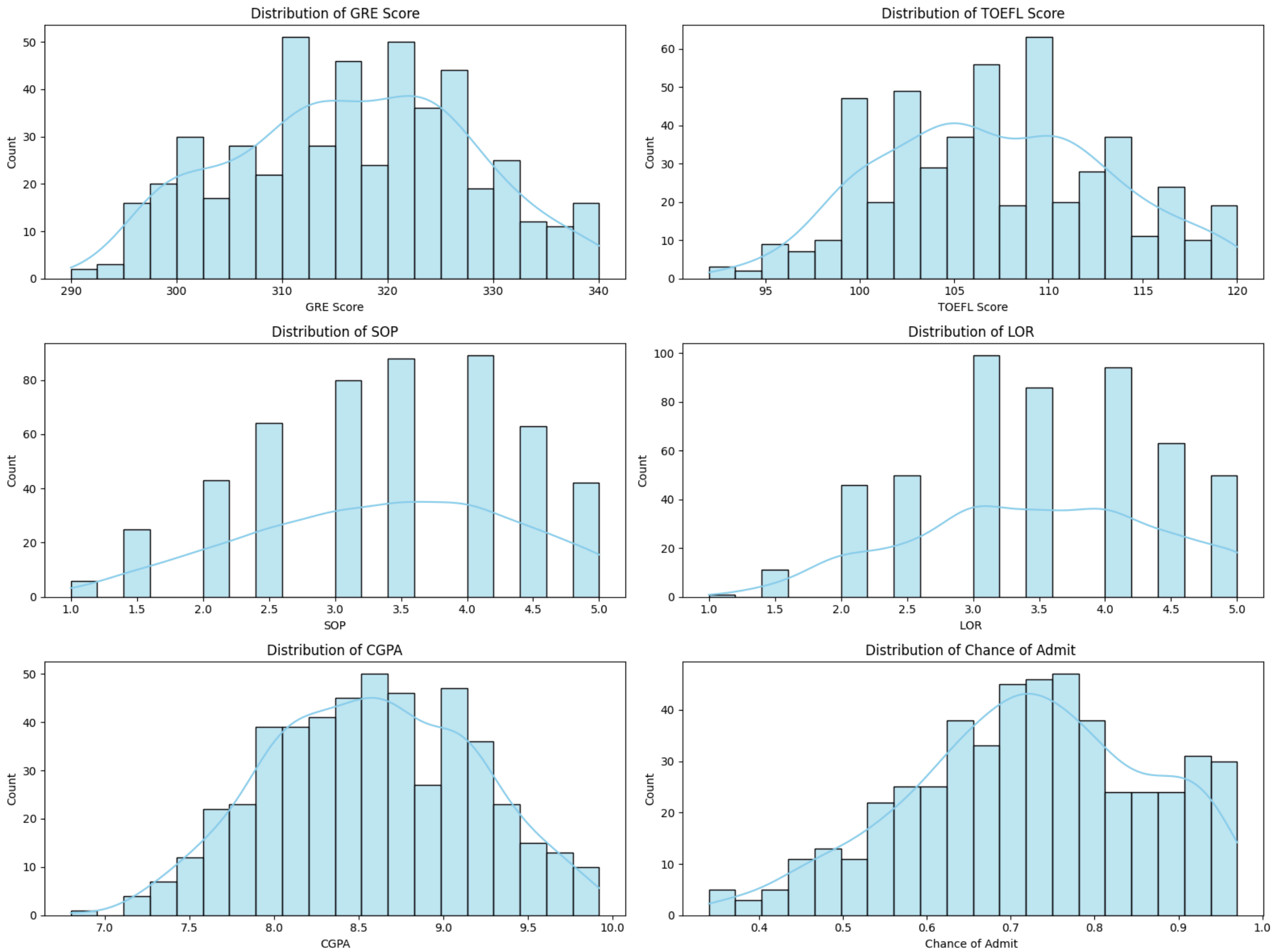
Data set having 9 columns and 500 rows

```
# checking null values
for i in range(len(df.columns)):
    print(df.columns[i],df.iloc[i].isnull().sum())
```

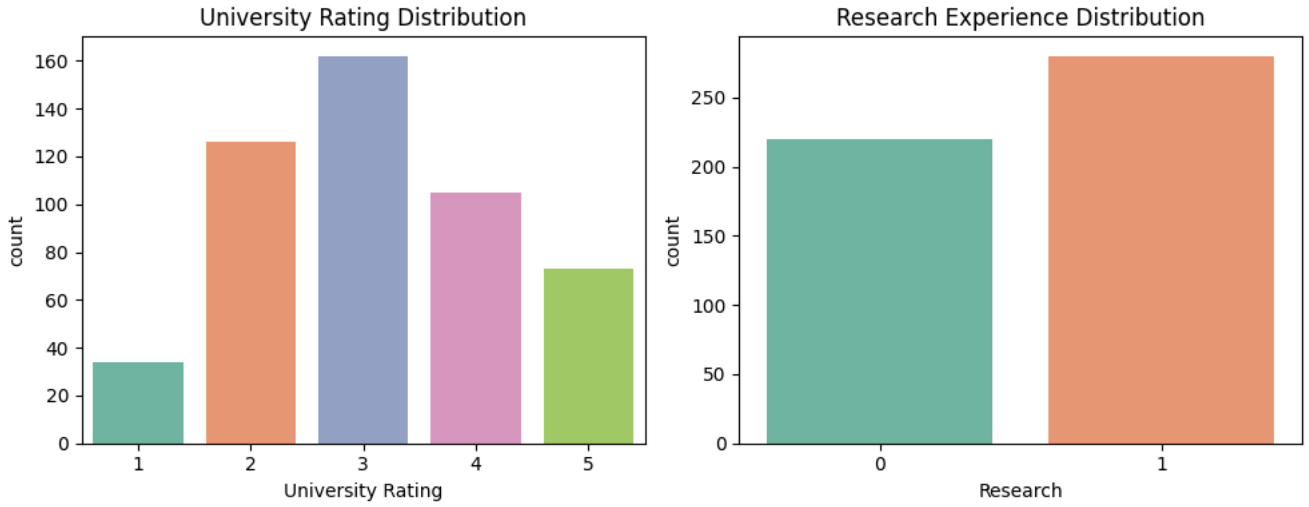
```
Serial No. 0
GRE Score 0
TOEFL Score 0
University Rating 0
SOP 0
LOR  0
CGPA 0
Research 0
Chance of Admit  0
```

As per above code no null or missing values found in data set

```
# Updated column names based on the Jamboree dataset structure
continuous_vars = ['GRE Score', 'TOEFL Score', 'SOP', 'LOR ', 'CGPA', 'Chance of Admit ']

# Plotting distributions for continuous variables
plt.figure(figsize=(16, 12))
for i, col in enumerate(continuous_vars):
    plt.subplot(3, 2, i + 1)
    sns.histplot(df[col], kde=True, bins=20, color='skyblue')
    plt.title(f'Distribution of {col}')
plt.tight_layout()
plt.show()

# Univariate plots for categorical variables
plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1)
sns.countplot(x='University Rating', data=df, palette='Set2')
plt.title('University Rating Distribution')

plt.subplot(1, 2, 2)
sns.countplot(x='Research', data=df, palette='Set2')
plt.title('Research Experience Distribution')
plt.tight_layout()
plt.show()
```



```
<ipython-input-7-1142741701>:16: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(x='University Rating', data=df, palette='Set2')
<ipython-input-7-1142741701>:20: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(x='Research', data=df, palette='Set2')
```



1. GRE Score

Range: Most scores cluster between 300–330, with fewer applicants scoring below 290 or above 340.

Implication: Competitive scores are likely in the 310–330 range. Extremely high (>330) or low (<290) scores are rare.

screwness - Slightly Left

2. TOEFL Score

Peak: Scores are concentrated around 105–115, indicating strong English proficiency among applicants.

Tail: Few applicants score below 95 or above 120, suggesting these are outliers.

screwness - Near Symmetric

**3. CGPA (Undergraduate GPA) **

Distribution: Most applicants have a CGPA between 8.0–9.0, with very few below 7.5 or above 9.5.

Insight: A CGPA of 8.5+ appears to be a competitive benchmark for admissions.

screwness - Left-Skewed

4. SOP (Statement of Purpose Strength) Trend: Ratings are skewed toward 3.0–4.5, with fewer at the extremes (1.0–2.0 or 5.0).

Observation: Most applicants submit moderately strong SOPs, but very few achieve a perfect 5.0.

screwness - Right-Skewed

5. LOR (Letter of Recommendation Strength)

Pattern: Similar to SOP, most ratings are in the 3.0–4.5 range, with very few at 1.0–2.0 or 5.0.

Takeaway: Strong LORs (≥4.0) may be a differentiating factor.

screwness - Right-Skewed

6. Chance of Admit

Bimodal?: Peaks around 0.6–0.7 and 0.8–0.9, suggesting two distinct applicant pools (moderate vs. high chance)

screwness - left-Skewed

7. University Rating

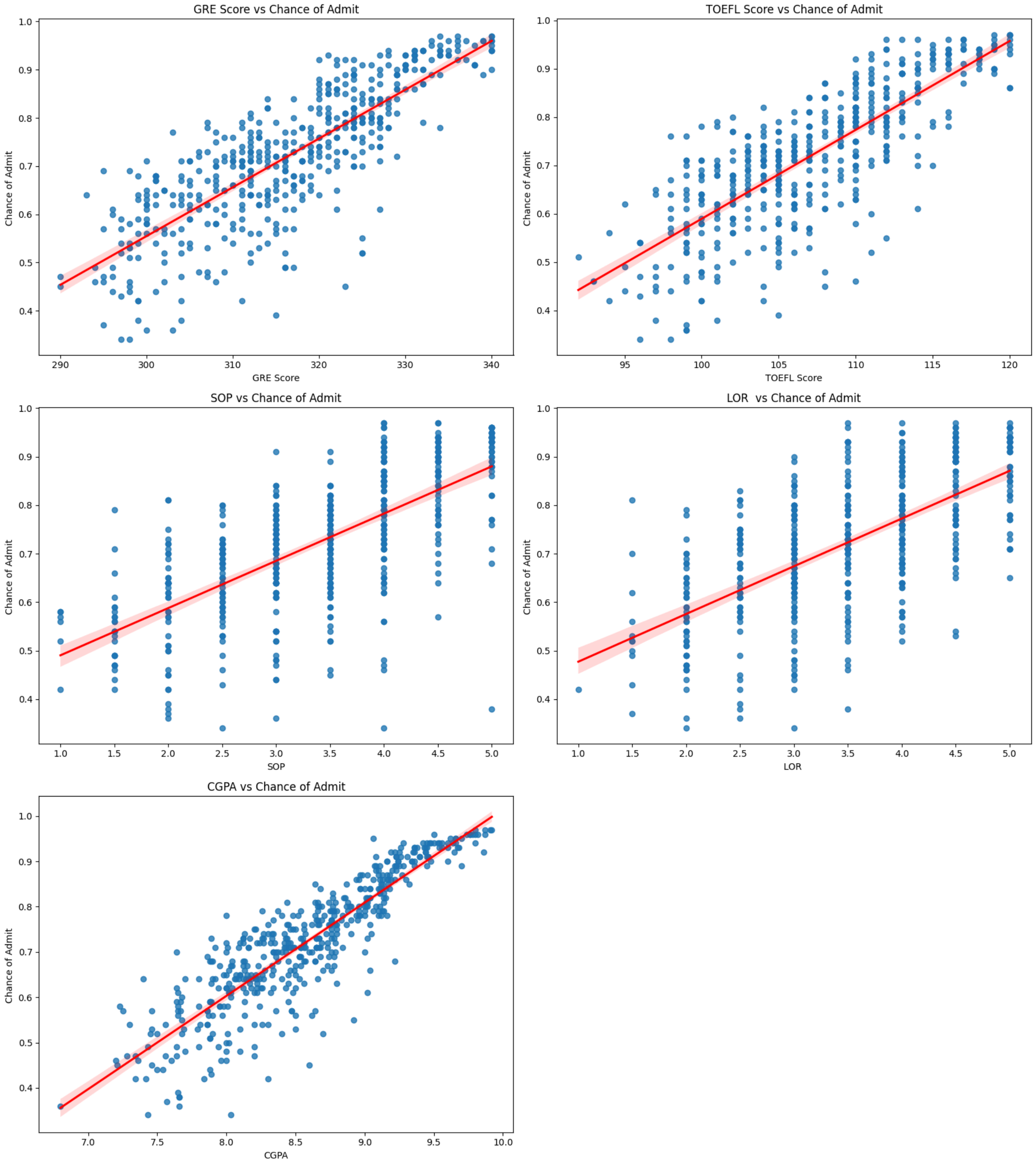
Most universities are rated mid-tier (3 or 4), with fewer at extremes (1 or 5).

8. Research Experience

0: No research experience (~50–100 applicants).

1: Has research experience (~200–250 applicants).

```
# Scatter/regression plots: Relation of each independent variable with Chance of Admit
plt.figure(figsize=(16, 18))
for i, col in enumerate(continuous_vars[:-1]):  # exclude target variable itself
    plt.subplot(3, 2, i + 1)
    sns.regplot(x=df[col], y=df['Chance of Admit '], line_kws={"color": "red"})
    plt.title(f'{col} vs Chance of Admit')
plt.tight_layout()
plt.show()
```



1. GRE Score vs Chance of Admit

Trend: Strong positive linear relationship.

Insight: Higher GRE scores are significantly associated with a higher probability of admission.

Implication: Students should aim for a high GRE score to improve their chances.

2. TOEFL Score vs Chance of Admit

Trend: Positive linear correlation, though slightly weaker than GRE.

Insight: TOEFL is important, but has a more moderate influence compared to GRE.

Implication: A good TOEFL score helps, especially for non-native English speakers, but may not be the most critical factor.

3. SOP (Statement of Purpose) vs Chance of Admit

Trend: Mild to moderate positive correlation.

Insight: SOP strength does influence admission chances, but not as sharply as academic scores.

Implication: A well-crafted SOP can enhance chances, especially when scores are borderline.

4. LOR (Letter of Recommendation) vs Chance of Admit

Trend: Similar to SOP, showing a modest positive trend.

Insight: Strong LORs support applications, but on their own may not drastically shift outcomes.

Implication: Good LORs are necessary, especially when paired with solid academics.

5. CGPA vs Chance of Admit Trend: Very strong positive linear relationship.

Insight: CGPA is a major determinant in predicting admission chances—possibly the most important among all.

Implication: Students with higher undergrad GPAs are far more likely to get into top programs.

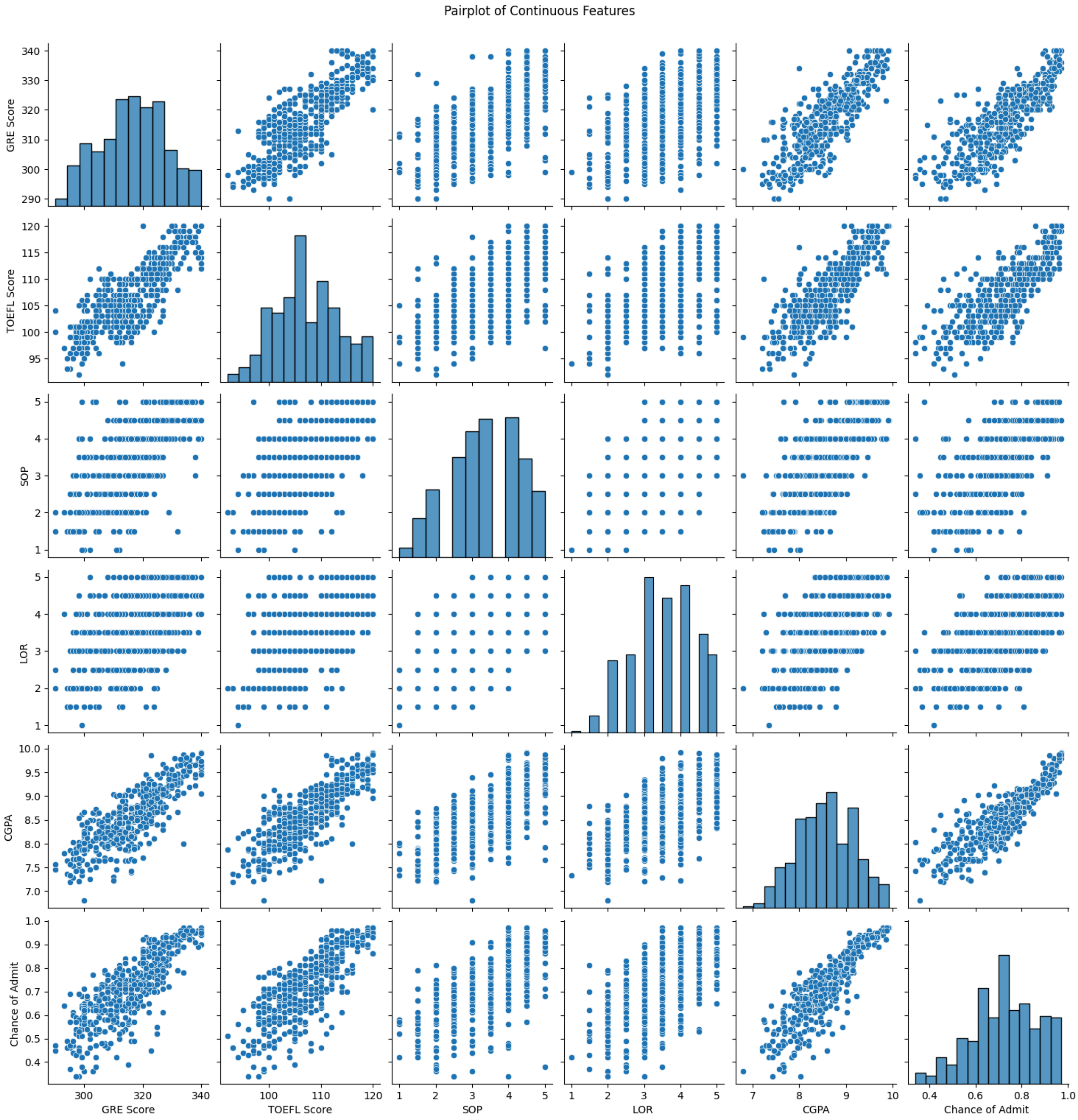
General Observation:

CGPA and GRE Score are the most influential predictors of admission chances.

TOEFL, SOP, and LOR have secondary but still positive impact.

All variables show a positive relationship with admission chances—higher values → higher probability.

```
# Pairplot for selected continuous features
sns.pairplot(df[continuous_vars])
plt.suptitle("Pairplot of Continuous Features", y=1.02)
plt.show()
```



```
# Check for missing values in each column
missing_values = df.isnull().sum()
print("Missing values in each column:")
print(missing_values)
```

```
Missing values in each column:
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
```

```
Research             0
Chance of Admit      0
dtype: int64
```

NO missing values found in data set

```
# checking outliers
# Dictionary to store outlier counts
outlier_counts = {}
for i in continuous_vars:
    Q1 = df[i].quantile(0.25)
    Q3 = df[i].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[i] < lower_bound) | (df[i] > upper_bound)]
    print (outliers)
    outlier_counts[i] = outliers.shape[0]  # rows are caculated as number of outliers
    print(f"Number of outliers in {i}: {outlier_counts[i]}")
```

```
Empty DataFrame
Columns: [Serial No., GRE Score, TOEFL Score, University Rating, SOP, LOR , CGPA, Research, Chance of Admit ]
Index: []
Number of outliers in GRE Score: 0
Empty DataFrame
Columns: [Serial No., GRE Score, TOEFL Score, University Rating, SOP, LOR , CGPA, Research, Chance of Admit ]
Index: []
Number of outliers in TOEFL Score: 0
Empty DataFrame
Columns: [Serial No., GRE Score, TOEFL Score, University Rating, SOP, LOR , CGPA, Research, Chance of Admit ]
Index: []
Number of outliers in SOP: 0
     Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  \
347         348        299           94                  1  1.0   1.0  7.34

     Research  Chance of Admit 
347         0              0.42
Number of outliers in LOR : 1
Empty DataFrame
Columns: [Serial No., GRE Score, TOEFL Score, University Rating, SOP, LOR , CGPA, Research, Chance of Admit ]
Index: []
Number of outliers in CGPA: 0
     Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  \
92           93        298           98                  2  4.0   3.0  8.03
376         377        297           96                  2  2.5   2.0  7.43

     Research  Chance of Admit 
92          0              0.34
376         0              0.34
Number of outliers in Chance of Admit : 2
```

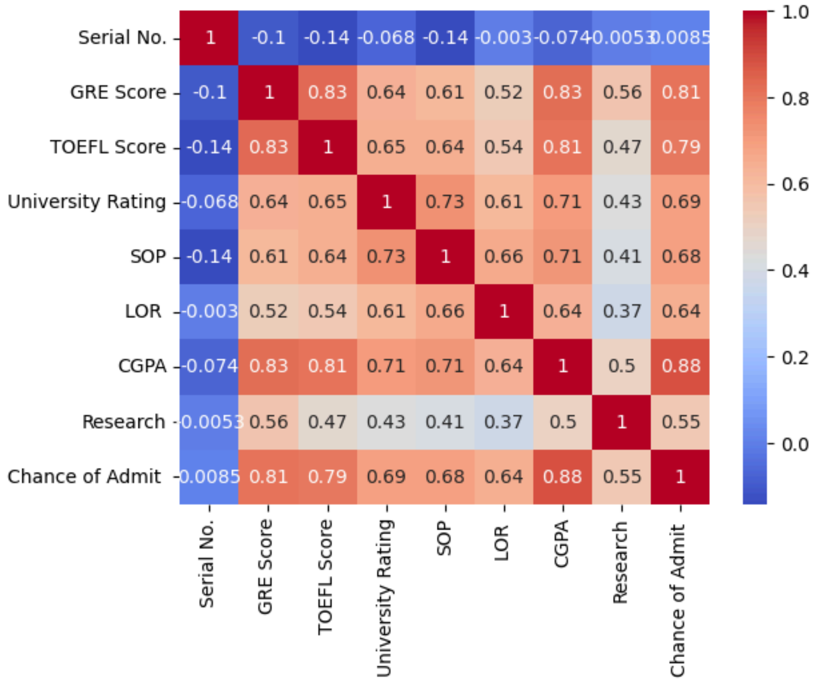
```
# treating outliers
for col in continuous_vars:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    df[col] = df[col].clip(lower, upper)
df.shape
```

```
(500, 9)
```

Outliers has been clipped

```
# heat map checking for correlation checking
corr = df.corr()
sns.heatmap(corr, annot=True, cmap="coolwarm")
```

```
<Axes: >
```



No feature pairs have a correlation higher than 0.90 — this means:

There is no multicollinearity strong enough to warrant dropping a variable.

All features are independently contributing useful information.

```
df
```

	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
...	...	...	...	...	...	...	...	...	...
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

500 rows × 9 columns

Next steps: Generate code with df View recommended plots New interactive sheet

```
X=df.drop(columns=['Chance of Admit ','Serial No.'])
y = df['Chance of Admit ']                # Target variable
# Step 2: Perform train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 3: Standardize the features (IMPORTANT: Do it after split)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Print shapes to verify
print("X_train_scaled shape:", X_train_scaled.shape)
print("X_test_scaled shape:", X_test_scaled.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train_scaled shape: (400, 7)
X_test_scaled shape: (100, 7)
y_train shape: (400,)
y_test shape: (100,)
```

```
import pandas as pd
import statsmodels.api as sm

# Step 1: Convert scaled features back to DataFrame and align index
X_train_scaled_df = pd.DataFrame(X_train_scaled, columns=X_train.columns, index=X_train.index)

# Step 2: Add constant term
X_train_sm = sm.add_constant(X_train_scaled_df)

# Step 3: Build and fit the model
model = sm.OLS(y_train, X_train_sm).fit()

# Step 4: View summary
print(model.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:                     257.4
Date:               Thu, 12 Jun 2025    Prob (F-statistic):          2.54e-142
Time:                       14:03:23    Log-Likelihood:                 562.41
No. Observations:                400    AIC:                            -1109.
Df Residuals:                    392    BIC:                            -1077.
Df Model:                          7
Covariance Type:           nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                 0.7242      0.003    241.749      0.000       0.718       0.730
GRE Score             0.0266      0.006      4.197      0.000       0.014       0.039
TOEFL Score           0.0182      0.006      3.172      0.002       0.007       0.029
University Rating     0.0029      0.005      0.610      0.542      -0.007       0.012
SOP                   0.0019      0.005      0.371      0.711      -0.008       0.012
LOR                   0.0158      0.004      3.759      0.000       0.008       0.024
CGPA                  0.0676      0.006     10.456      0.000       0.055       0.080
Research              0.0119      0.004      3.233      0.001       0.005       0.019
==============================================================================
Omnibus:                      85.580    Durbin-Watson:                   2.049
Prob(Omnibus):                 0.000    Jarque-Bera (JB):              187.432
Skew:                         -1.102    Prob(JB):                     1.99e-41
Kurtosis:                      5.528    Cond. No.                         5.65
==============================================================================

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

Comments on Initial Model Statistics: R-squared (0.821) and Adjusted R-squared (0.818): These values indicate that approximately 82.5% of the variance in 'Chance of Admit' can be explained by the independent variables in the model. The adjusted R-squared is very close to the R-squared, suggesting that the features included are generally contributing to the model's explanatory power.

F-statistic (257.4) and Prob (F-statistic) (2.54e-142): The extremely low p-value (close to 0) for the F-statistic indicates that the overall model is statistically significant. This means that at least one of the independent variables is useful in predicting the 'Chance of Admit'.

Individual Feature p-values:

'GRE Score', 'TOEFL Score', 'LOR', 'CGPA', and 'Research' all have p-values very close to 0, indicating they are highly statistically significant in predicting the 'Chance of Admit'.

'University Rating' has a p-value of 0.524, which is above the common significance level of 0.05. This suggests it's not significant.

'SOP' (Statement of Purpose) has a very high p-value of 0.711, meaning it is not statistically significant and likely does not contribute meaningfully to predicting the 'Chance of Admit' in this model.

```
# Step 5: Drop features with p-value > 0.05 (excluding constant)
insignificant_cols = model.pvalues[model.pvalues > 0.05].index.tolist()
insignificant_cols = [col for col in insignificant_cols if col != 'const']

# Step 6: Drop these from the DataFrame
X_reduced = X_train_scaled_df.drop(columns=insignificant_cols)
X_reduced_const = sm.add_constant(X_reduced)

# Step 7: Refit the model
model_refined = sm.OLS(y_train, X_reduced_const).fit()

# Step 8: View summary
print(model_refined.summary())
print(model_refined.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        Chance of Admit   R-squared:                       0.821
Model:                            OLS   Adj. R-squared:                  0.819
Method:                 Least Squares   F-statistic:                     361.4
Date:                Thu, 12 Jun 2025   Prob (F-statistic):         1.02e-144
Time:                        14:03:30   Log-Likelihood:                 562.03
No. Observations:                 400   AIC:                            -1112.
Df Residuals:                     394   BIC:                            -1088.
Df Model:                           5
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
const           0.7242      0.003    242.134      0.000       0.718       0.730
GRE Score       0.0268      0.006      4.245      0.000       0.014       0.039
TOEFL Score     0.0191      0.006      3.391      0.001       0.008       0.030
LOR             0.0172      0.004      4.469      0.000       0.010       0.025
CGPA            0.0691      0.006     11.163      0.000       0.057       0.081
Research        0.0122      0.004      3.331      0.001       0.005       0.019
==============================================================================
Omnibus:                       84.185   Durbin-Watson:                   2.053
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              182.540
Skew:                          -1.089   Prob(JB):                     2.30e-40
Kurtosis:                       5.492   Cond. No.                         4.76
==============================================================================

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

Comments on Retrained Model Statistics:

R-squared (0.821) and Adjusted R-squared (0.818): The R-squared value remained virtually the same (0.821) while the Adjusted R-squared slightly increased (0.819) compared to the initial model. This is a positive outcome, as it indicates that by removing the insignificant 'SOP' and 'University Rating' features, the model's explanatory power for new data (adjusted R-squared) has marginally improved, or at least not significantly decreased, while becoming simpler.

F-statistic (361.4) and Prob (F-statistic) (1.02e-144): The F-statistic increased, and its p-value remains extremely low. This confirms that the retrained model, even with fewer features, is still highly statistically significant, and the remaining features are collectively useful in predicting the 'Chance of Admit'.

Individual Feature p-values: All remaining features ('GRE Score', 'TOEFL Score', 'LOR', 'CGPA', and 'Research') now have p-values of 0.000, confirming their high statistical significance in the refined model.

Coefficients: The coefficients for the remaining features are very similar to those in the initial model, indicating that dropping 'University Rating' and 'SOP' did not drastically change the impact of the other predictors.

'CGPA' has the largest positive coefficient (0.0691), indicating it has the strongest positive impact on 'Chance of Admit'.

'LOR' (Letters of Recommendation), 'GRE Score', 'TOEFL Score', and 'Research' also show positive and significant effects.

```
# checking  Variance inflection factor VIF greater than 5

from statsmodels.stats.outliers_influence import variance_inflation_factor
vif = pd.DataFrame() # blank data frame
X_t = pd.DataFrame(X_reduced, columns=X_reduced.columns) # columns gives columns names
vif['Features'] = X_t.columns
vif['values'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.shape[1])]
# round off to 2 decima
vif['values'] = vif['values'].round(2)
# sort in decreasing order of vif
vif = vif.sort_values(by='values', ascending=False)
vif
```

	Features	values
0	GRE Score	4.47
3	CGPA	4.28
1	TOEFL Score	3.54
2	LOR	1.66
4	Research	1.50

Next steps: Generate code with vif View recommended plots New interactive sheet

Did not found any feature greater than VIF 5, hence did not removed any features

No multicollinearity problem among your current feature

```
# Get residuals from the model
residuals = model_refined.resid

# Print mean of residuals
print(f"Mean of residuals: {residuals.mean():.6f}")
```

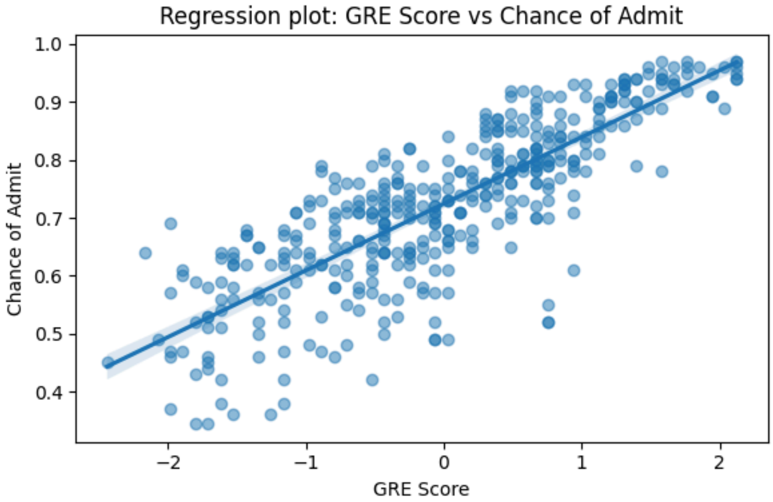
```
Mean of residuals: -0.000000
```

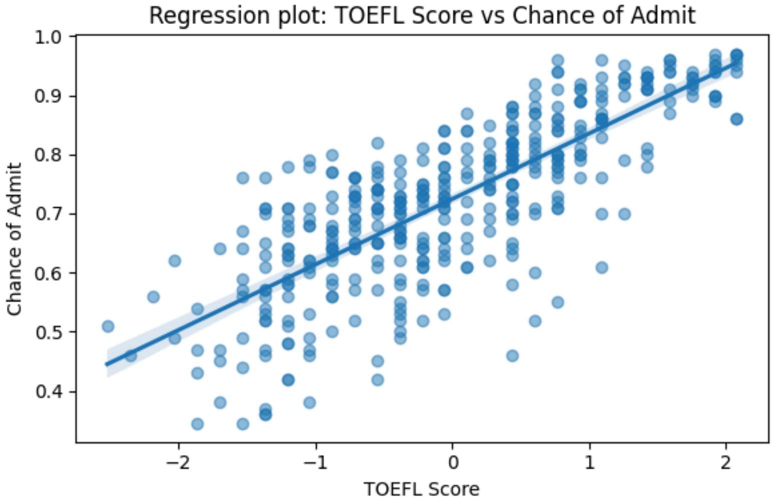
Interpretation:

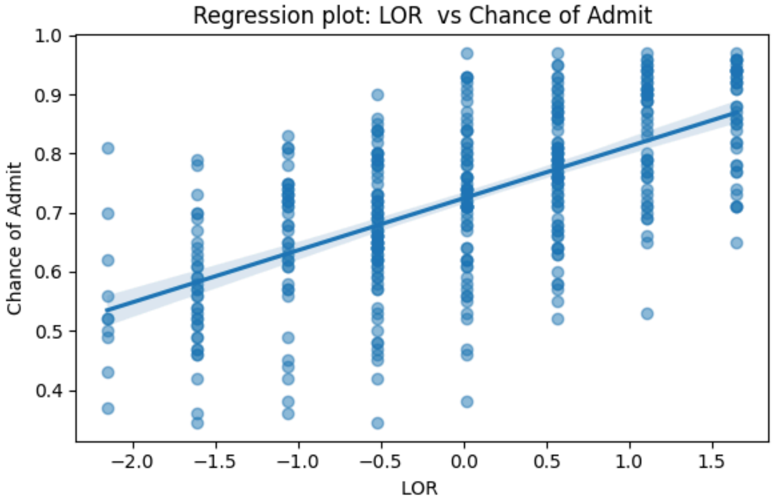
The output is something like 0.0000, it means the model does not systematically under- or over-predict.

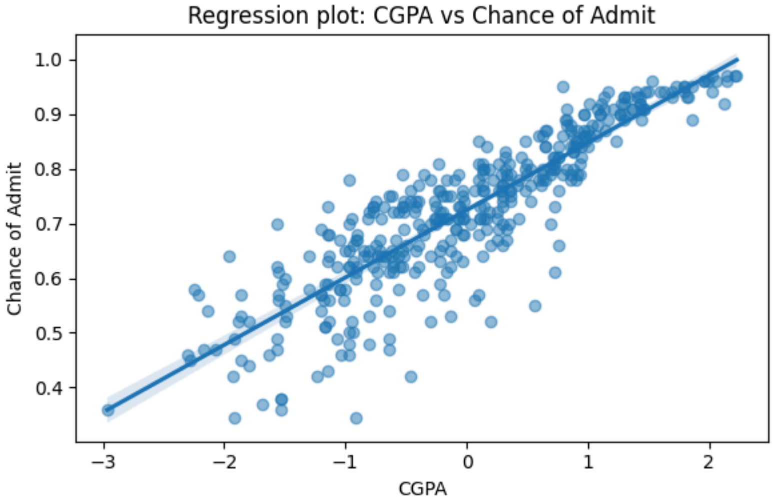
This is a key assumption of linear regression: Residuals should be normally distributed and centered around zero

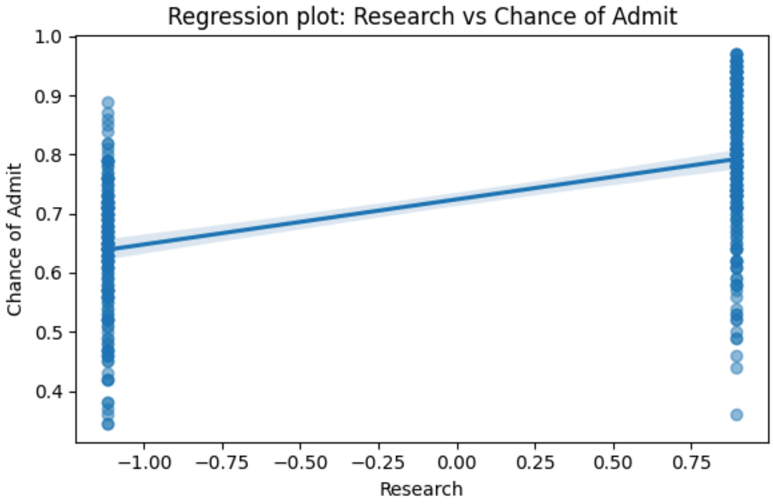
```
# List of final features used in the model
features = X_reduced.columns.tolist()

# Scatter and regression plots
for col in features:
    plt.figure(figsize=(6, 4))
    sns.regplot(x=X_reduced[col], y=y_train, scatter_kws={"alpha": 0.5})
    plt.title(f'Regression plot: {col} vs Chance of Admit')
    plt.xlabel(col)
    plt.ylabel('Chance of Admit')
    plt.tight_layout()
    plt.show()
```


Regression plot: GRE Score vs Chance of Admit
Chance of Admit
1.0
0.9
0.8
0.7
0.6
0.5
0.4
−2
−1
0
1
2
GRE Score


Regression plot: TOEFL Score vs Chance of Admit
Chance of Admit
1.0
0.9
0.8
0.7
0.6
0.5
0.4
−2
−1
0
1
2
TOEFL Score

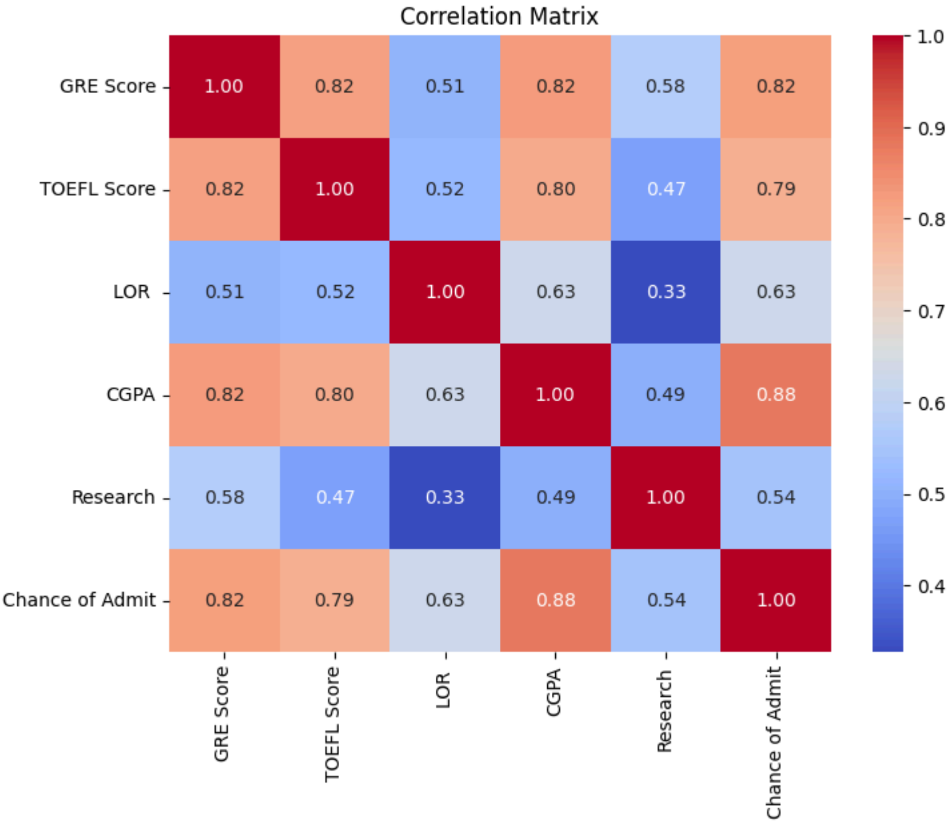

Regression plot: LOR vs Chance of Admit
Chance of Admit
1.0
0.9
0.8
0.7
0.6
0.5
0.4
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
LOR


Regression plot: CGPA vs Chance of Admit
Chance of Admit
1.0
0.9
0.8
0.7
0.6
0.5
0.4
−3
−2
−1
0
1
2
CGPA



```
# Combine features and target into one DataFrame
df_corr = X_reduced.copy()
df_corr['Chance of Admit'] = y_train

# Compute Pearson correlation matrix
corr_matrix = df_corr.corr()

# Show correlations with target variable
print("Pearson Correlation with Chance of Admit:")
print(corr_matrix['Chance of Admit'].sort_values(ascending=False))

# Heatmap for visualization
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```

```
Pearson Correlation with Chance of Admit:
Chance of Admit    1.000000
CGPA               0.878279
GRE Score          0.820594
TOEFL Score        0.789338
LOR                0.627064
Research           0.544122
Name: Chance of Admit, dtype: float64
```



CGPA and Chance of Admit: 0.88 → Strongest correlation → Most important predictor.

GRE Score and Chance of Admit: 0.82 → Strong influence.

TOEFL Score and Chance of Admit: 0.79 → Also an important factor.

LOR and Chance of Admit: 0.63 → Moderate influence.

Research and Chance of Admit: 0.54 → Has a positive but weaker correlation.

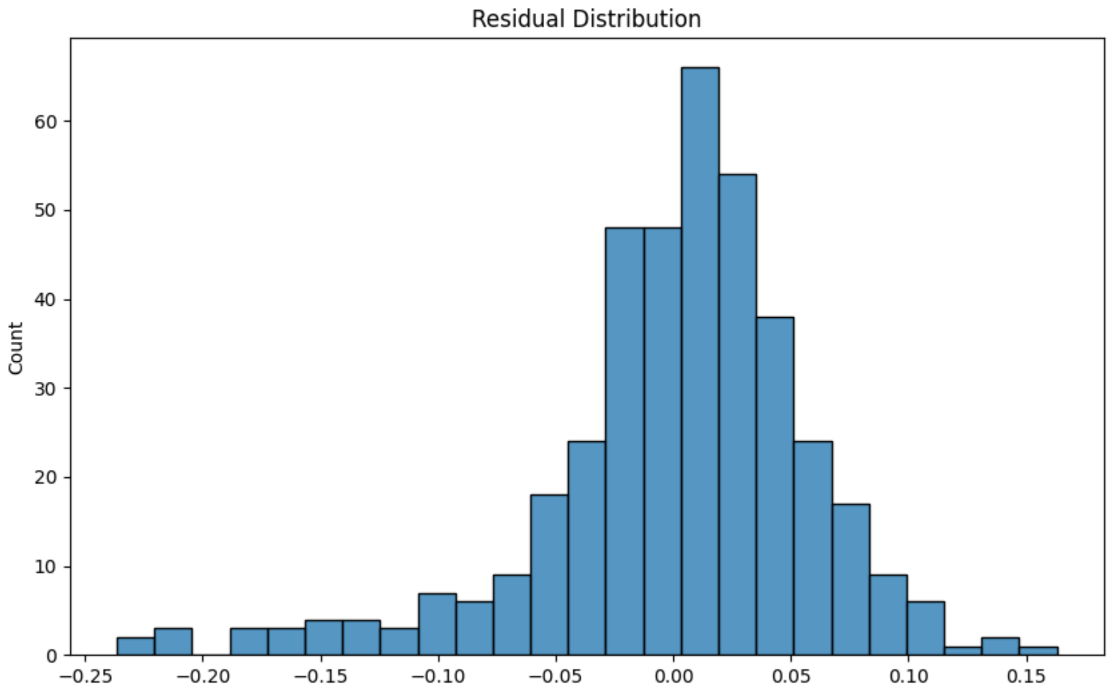
Conclusion

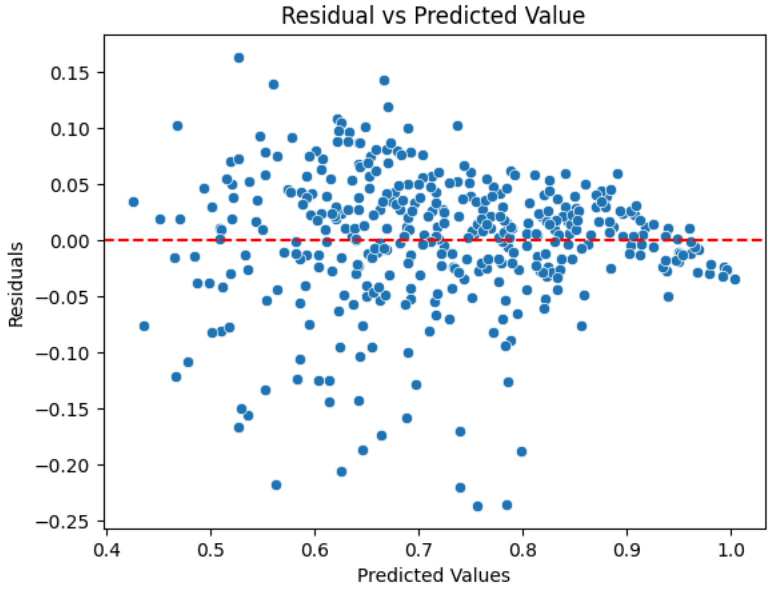
CGPA, GRE, and TOEFL are the most influential features for predicting admissions.

No need to drop any variable for high correlation (>0.90).

Research and LOR add useful information but are less predictive than academic scores.

```
y_hat = model_refined.predict(X_reduced_const)
errors  = y_train - y_hat


# checking by histrogram the normal disctrubution of errors
plt.figure(figsize=(10, 6))
sns.histplot(errors)
plt.title("Residual Distribution")
plt.show()

# Check for Homoscedestacity(No Heteroscedeasticity)

# scatter plot of y_hat vs error
sns.scatterplot(x=y_hat, y=errors)
plt.axhline(0, color='red', linestyle='--')
plt.xlabel("Predicted Values")
plt.ylabel("Residuals")
plt.title("Residual vs Predicted Value")
plt.show()
```





Plot 1: Residual Distribution Histogram.

Observations:

The residuals are roughly bell-shaped and centered around 0. It shows slight right skew, but the distribution is mostly symmetric.

Conclusion:

Residuals are approximately normally distributed, which is a key assumption in linear regression.

You can confirm this using Shapiro-Wilk or Q-Q plots if needed.

Plot 2: Residuals vs Predicted Values

Observations:

The residuals are scattered around the horizontal axis (near 0), which is a good sign.

However, the spread seems to reduce slightly as predicted values increase. This may indicate mild heteroscedasticity (funnel shape).

Still, there are no strong patterns, which is acceptable for linear regression.

Further to confirm it statistically with the Goldfeld-Quandt test (statsmodels.stats.diagnostic.het_goldfeldquandt).

```
# perform the goldfeld-quandt test to check for homoscedasticity
from statsmodels.stats.diagnostic import het_goldfeldquandt
from statsmodels.compat import lzip
import statsmodels.api as sm
import statsmodels.stats.api as sms
# Run Goldfeld-Quandt Test
```

```
name = ['F statistic', 'p-value']
test = sms.het_goldfeldquandt(errors, X_reduced_const)
results = lzip(name, test)
print(results)
# Extract p-value from test
p_value = test[1]
alpha = 0.05
# Correct hypothesis logic
if p_value <= alpha:
    print("Reject the null hypothesis → Heteroscedasticity is present.")
else:
    print("Fail to reject the null hypothesis → Data is likely homoscedastic.")
```

```
[('F statistic', np.float64(0.9637608457816248)), ('p-value', np.float64(0.6013043502651503))]
Fail to reject the null hypothesis → Data is likely homoscedastic.
```

p-value = 0.6013

α (significance level) = 0.05

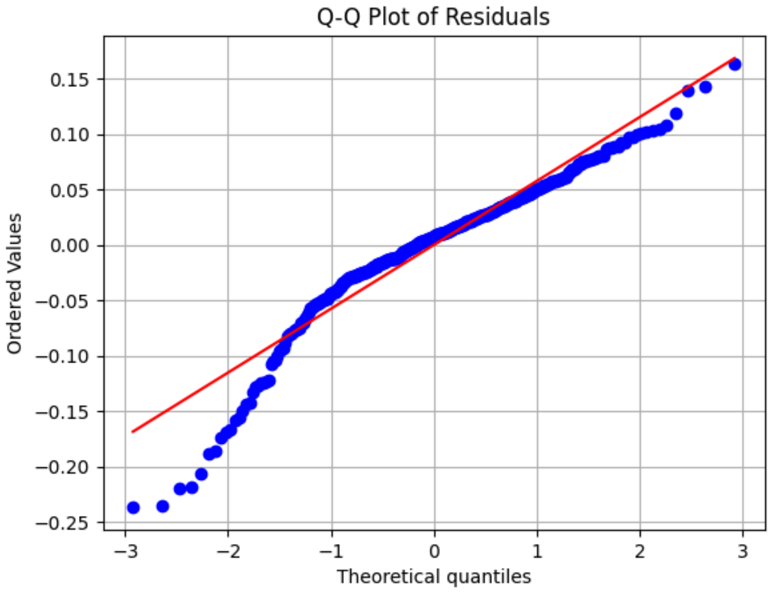
Since p-value > 0.05, fail to reject the null hypothesis.

The null hypothesis of the Goldfeld-Quandt test is that the data is homoscedastic (i.e., residuals have constant variance).

Result suggests that the residuals have constant variance, meaning homoscedasticity is present, which is a good thing for linear regression assumptions.

Final Takeaway: p = 0.6013 is good. It means there is no strong evidence of heteroscedasticity. Our model satisfies the homoscedasticity assumption.

```
import scipy.stats as stats
import matplotlib.pyplot as plt

# Q-Q plot of residuals
stats.probplot(errors, dist="norm", plot=plt)
plt.title("Q-Q Plot of Residuals")
plt.grid(True)
plt.show()
```



The majority of points lie close to the red diagonal line, especially in the middle quantiles.

This suggests that residuals are approximately normally distributed.

Some points deviate from the line at the extreme ends (both left and right tails).

This indicates mild skewness or heavier tails, but this level of deviation is common.

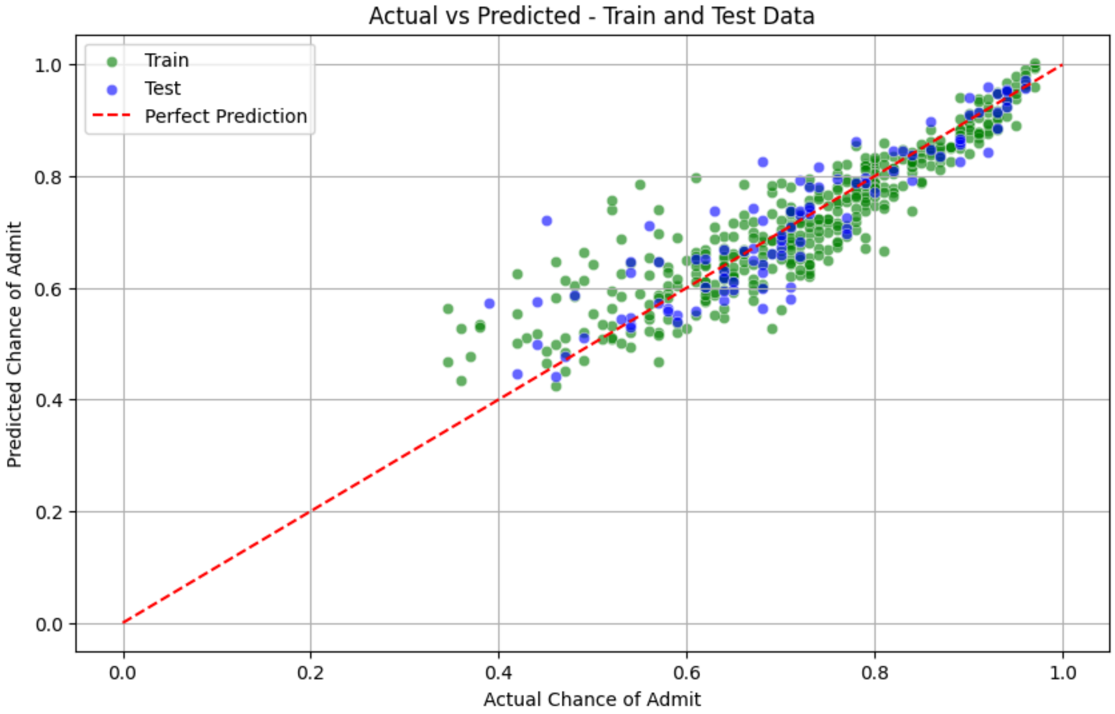
The plot supports the normality assumption of residuals, which is important for reliable inference in linear regression (e.g., p-values, confidence intervals).

Normality of residuals is reasonably satisfied. No strong violation is observed

```
# evaluating model perfomance
# Convert the scaled test data back into a DataFrame with correct column names and index
X_test_scaled_df = pd.DataFrame(X_test_scaled, columns=X_test.columns, index=X_test.index)


# Predict on test set
y_pred = model_refined.predict(sm.add_constant(X_test_scaled_df[X_reduced.columns]))

# Evaluation Metrics
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

# Adjusted R²
n = X_test_scaled_df.shape[0]  # Number of samples
p = X_reduced.shape[1]         # Number of predictors
adj_r2 = 1 - (1 - r2) * (n - 1) / (n - p - 1)

# Display
print(f"MAE: {mae:.4f}")
print(f"RMSE: {rmse:.4f}")
print(f"R² Score: {r2:.4f}")
print(f"Adjusted R²: {adj_r2:.4f}")
```

```
MAE: 0.0430
RMSE: 0.0614
R² Score: 0.8154
Adjusted R²: 0.8055
```

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

# Assuming these variables are defined:
# y_train, y_train_pred, y_test, y_test_pred

plt.figure(figsize=(10, 6))

# Plot for training data
sns.scatterplot(x=y_train, y=y_train_pred, label='Train', color='green', alpha=0.6)

# Plot for test data
sns.scatterplot(x=y_test, y=y_pred, label='Test', color='blue', alpha=0.6)

# Reference line for perfect prediction
plt.plot([0, 1], [0, 1], color='red', linestyle='--', label='Perfect Prediction')

# Labels and title
plt.xlabel('Actual Chance of Admit')
plt.ylabel('Predicted Chance of Admit')
plt.title('Actual vs Predicted - Train and Test Data')
plt.legend()
plt.grid(True)
plt.show()
```



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

# Predict on training data
y_train_pred = model_refined.predict(X_reduced_const)

# Calculate metrics
mae_train = mean_absolute_error(y_train, y_train_pred)
rmse_train = np.sqrt(mean_squared_error(y_train, y_train_pred))
r2_train = r2_score(y_train, y_train_pred)
n = X_reduced_const.shape[0]
k = X_reduced_const.shape[1] - 1  # subtracting constant
adj_r2_train = 1 - ((1 - r2_train) * (n - 1)) / (n - k - 1)

# Print results
print(f"Train MAE: {mae_train:.4f}")
print(f"Train RMSE: {rmse_train:.4f}")
print(f"Train R² Score: {r2_train:.4f}")
print(f"Train Adjusted R²: {adj_r2_train:.4f}")
```

```
Train MAE: 0.0427
Train RMSE: 0.0594
Train R² Score: 0.8210
Train Adjusted R²: 0.8187
```

Start coding or generate with AI.

training performance metrics

Train MAE: 0.0427 Train RMSE: 0.0594 Train R² Score: 0.8210 Train Adjusted R²: 0.8187

test set metrics performance

Test MAE: 0.0430 Test RMSE: 0.0614 Test R² Score: 0.8154 Test Adjusted R²: 0.8055

Insights on Model Performance:

Train and Test R² scores are close (0.8210 vs 0.8154) → The model generalizes well to unseen data.

Very low MAE and RMSE on both sets → Prediction errors are minimal.

There is no big gap between training and test performance.

The model is neither too complex (overfitting) nor too simple (underfitting).

Adjusted R² values (0.8187 for train and 0.8055 for test) are strong, meaning the model explains a large portion of the variance in the target.

Final Comment: This linear regression model is performing very well and is suitable for predicting graduate admission chances.

Actionable insights

1. GRE, TOEFL, and CGPA are Strong Predictors These three features show a strong positive correlation with Chance of Admit.

Students with high scores in GRE (out of 340) and TOEFL (out of 120) have significantly better chances.

Undergraduate GPA (out of 10) also plays a crucial role.

Recommendation: Help students target strong test preparation for GRE & TOEFL and provide academic support to improve GPA.

2. Research Experience Matters Candidates with Research Experience (1) tend to have higher admission chances than those without.

Even though it's binary, its presence provides a competitive edge.

Recommendation: Encourage students to participate in research internships or publish papers, even at the undergraduate level.

3. SOP and LOR Scores Have Moderate Influence The Statement of Purpose (SOP) and Letter of Recommendation (LOR) strength (rated 1–5) also positively affect admission chances.

However, their impact is less than GPA, GRE, and TOEFL.

Recommendation: Offer personalized workshops on writing effective SOPs and getting impactful LORs.

4. University Rating Correlation Is Moderate Higher University Ratings improve admission chances but are not the strongest predictor.

This might reflect the subjective nature or variability across rating systems.

Recommendation: While targeting higher-rated universities is beneficial, students shouldn't ignore mid-tier schools with flexible admission policies.

5. No Strong Multicollinearity Detected VIF analysis shows no features are highly collinear (VIF < 5), so the model is stable.