Project Title: Jamboree Education - Linear Regression

1. Problem Definition and Exploratory Data Analysis

Problem Definition:

Jamboree has introduced a feature on its website to help students estimate their chances of securing admission to top Ivy League graduate programs. This initiative supports students in prioritizing aspects like GRE preparation or research involvement by providing data-driven insights into the admissions process. The dataset includes several academic and qualitative features, such as GRE scores, TOEFL scores, GPA, research experience, and recommendation strength. The analysis aims to identify key factors influencing admission chances, assess their interrelationships, and develop a robust predictive model that empowers students to improve their profiles effectively.

The dataset includes:

- **GRE Scores** (out of 340)
- **TOEFL Scores** (out of 120)
- **University Rating** (out of 5)
- **Statement of Purpose (SOP) and LOR Strength** (out of 5)
- **Undergraduate GPA** (out of 10)
- **Research Experience** (binary: 0 or 1)
- **Chance of Admit** (target variable: continuous value from 0 to 1)

Exploratory Data Analysis:

We begin by importing and checking the structure of the dataset:

In [7]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.model_selection import train_test_split
```

In [8]:
```python
# Load dataset
df = pd.read_csv("/Users/gopalmacbook/Downloads/Jamboree_Admission.csv")
df.head()
```

Out[8]:

	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

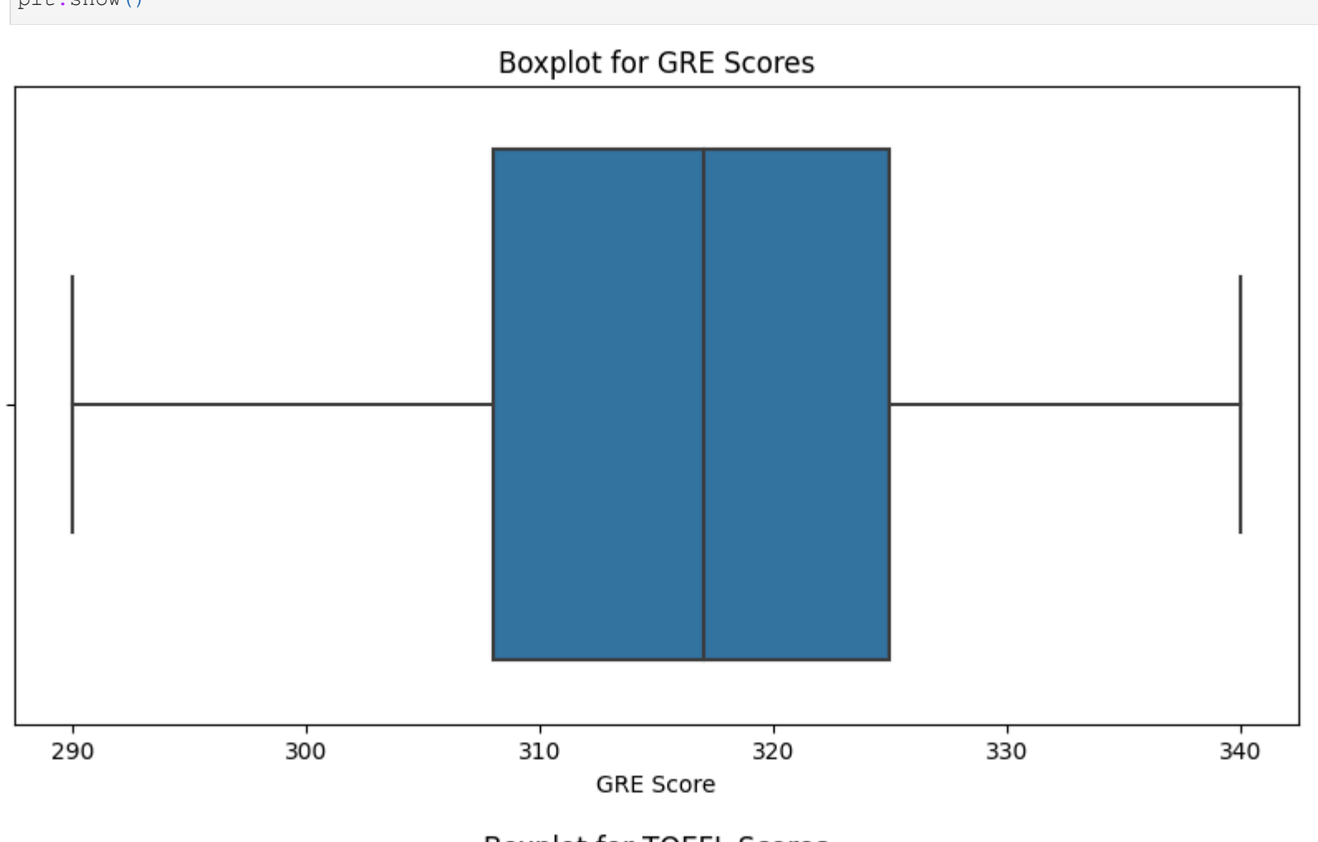
In [9]:
```python
# Checking dataset info and summary statistics

print(df.info())
print(df.describe())
```

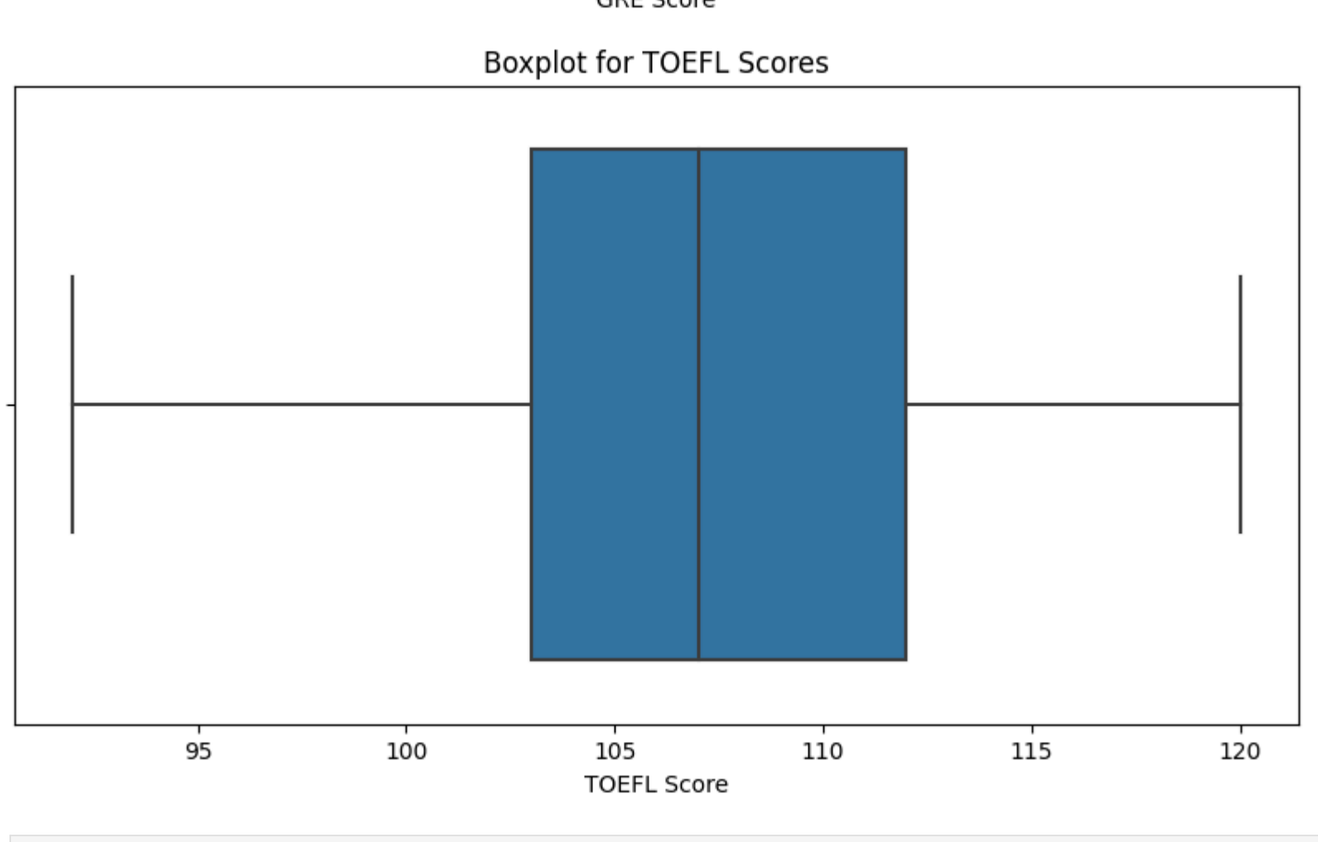
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Serial No.         500 non-null    int64
 1   GRE Score          500 non-null    int64
 2   TOEFL 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
None
       Serial No.   GRE Score  TOEFL Score  University Rating         SOP  \
count  500.000000  500.000000   500.000000         500.000000  500.000000
mean   250.500000  316.472000   107.192000           3.114000    3.374000
std    144.481833   11.295148     6.081868           1.143512    0.991004
min      1.000000  290.000000    92.000000           1.000000    1.000000
25%    125.750000  308.000000   103.000000           2.000000    2.500000
50%    250.500000  317.000000   107.000000           3.000000    3.500000
75%    375.250000  325.000000   112.000000           4.000000    4.000000
max    500.000000  340.000000   120.000000           5.000000    5.000000

            LOR         CGPA    Research  Chance of Admit
count  500.00000  500.000000  500.000000         500.00000
mean     3.48400    8.576440    0.560000           0.72174
std      0.92545    0.604813    0.496884           0.14114
min      1.00000    6.800000    0.000000           0.34000
25%      3.00000    8.127500    0.000000           0.63000
50%      3.50000    8.560000    1.000000           0.72000
75%      4.00000    9.040000    1.000000           0.82000
max      5.00000    9.920000    1.000000           0.97000
```

In [10]:
```python
print(df.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
dtype: int64
```

In [11]:
```python
# Boxplot for GRE Scores
plt.figure(figsize=(10, 5))
sns.boxplot(x=df['GRE Score'])
plt.title('Boxplot for GRE Scores')
plt.show()

# Boxplot for TOEFL Scores
plt.figure(figsize=(10, 5))
sns.boxplot(x=df['TOEFL Score'])
plt.title('Boxplot for TOEFL Scores')
plt.show()
```





In [12]:
```python
# Calculate Z-scores for GRE and TOEFL Scores
gre_z_scores = stats.zscore(df['GRE Score'])
toefl_z_scores = stats.zscore(df['TOEFL Score'])

# Define a threshold for outliers (commonly |z| > 3)
outliers_gre = np.where(np.abs(gre_z_scores) > 3)[0]
outliers_toefl = np.where(np.abs(toefl_z_scores) > 3)[0]

# Show the indices of the outliers
print(f"Outliers in GRE Scores at indices: {outliers_gre}")
print(f"Outliers in TOEFL Scores at indices: {outliers_toefl}")
```

```
Outliers in GRE Scores at indices: []
Outliers in TOEFL Scores at indices: []
```

In [13]:
```python
Q1 = df['GRE Score'].quantile(0.25)
Q3 = df['GRE Score'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers_gre_iqr = df[(df['GRE Score'] < lower_bound) | (df['GRE Score'] > upper_bound)]
print(outliers_gre_iqr)
```

```
Empty DataFrame
Columns: [Serial No., GRE Score, TOEFL Score, University Rating, SOP, LOR , CGPA, Research, Chance of Admit ]
Index: []
```

In [14]:
```python
gre_cap = df['GRE Score'].quantile(0.95)
toefl_cap = df['TOEFL Score'].quantile(0.95)
df['GRE Score'] = np.where(df['GRE Score'] > gre_cap, gre_cap, df['GRE Score'])
df['TOEFL Score'] = np.where(df['TOEFL Score'] > toefl_cap, toefl_cap, df['TOEFL Score'])
```

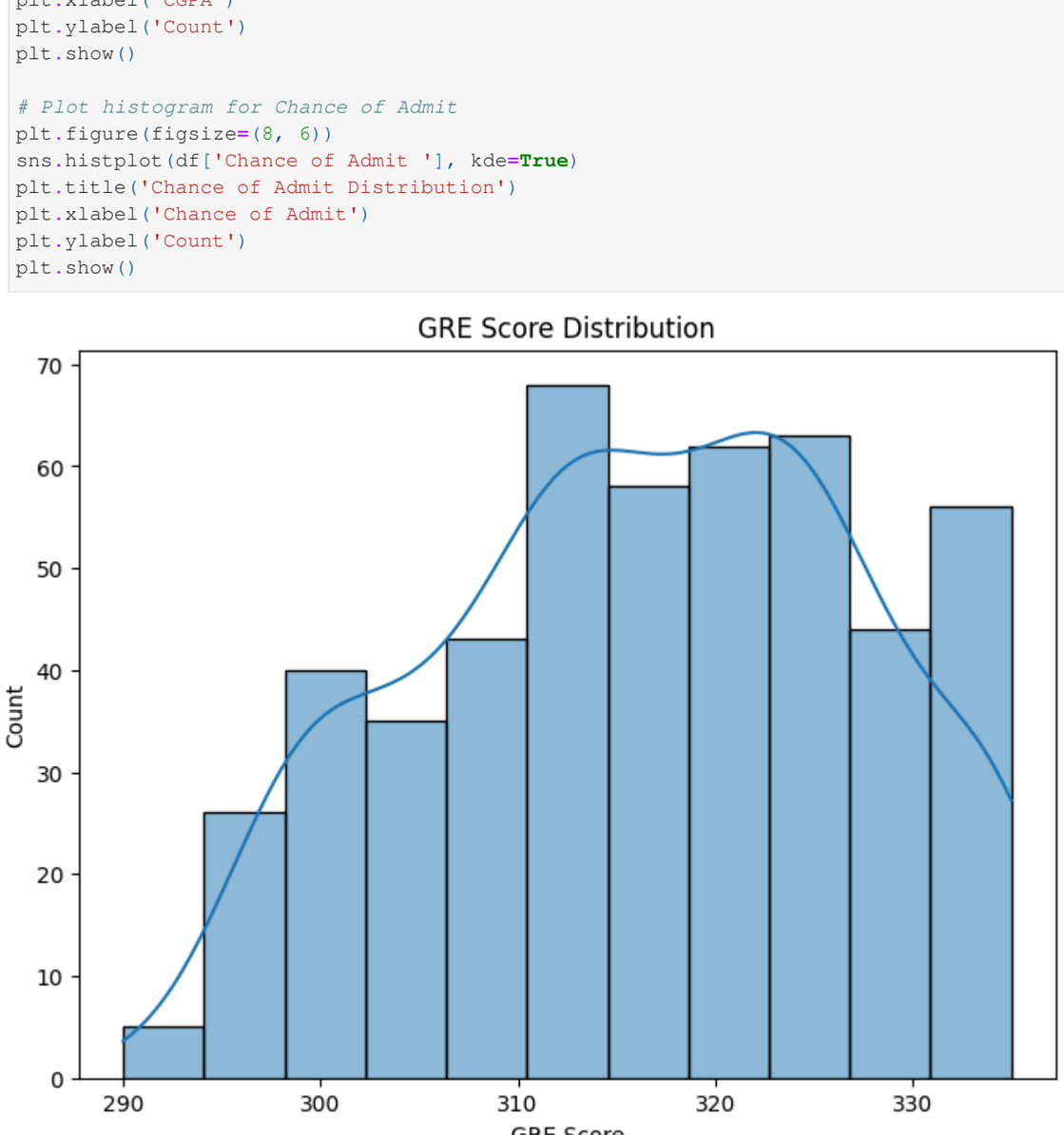
Observations:

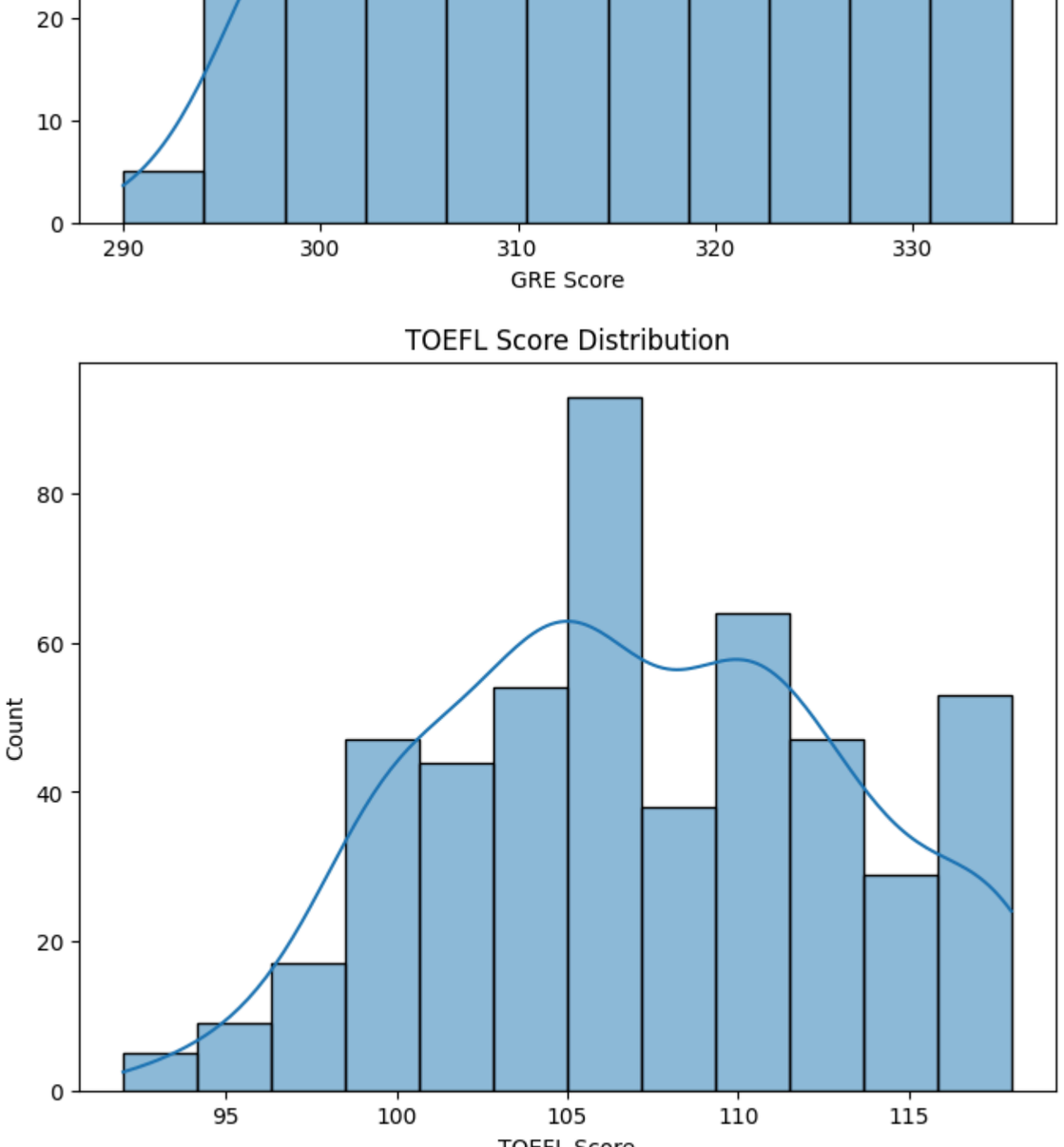
- **Missing Values**: No missing values were found in the dataset.
- **Data Types**: Data types are correct: numerical data for GRE, TOEFL, GPA, etc., and categorical data (binary) for Research Experience.
- **Outliers**: No significant outliers were detected in GRE and TOEFL scores using boxplots, z-scores (|z| > 3), or the IQR method. As a precautionary measure, capping was applied at the 95th percentile for GRE and TOEFL scores to handle potential edge cases.

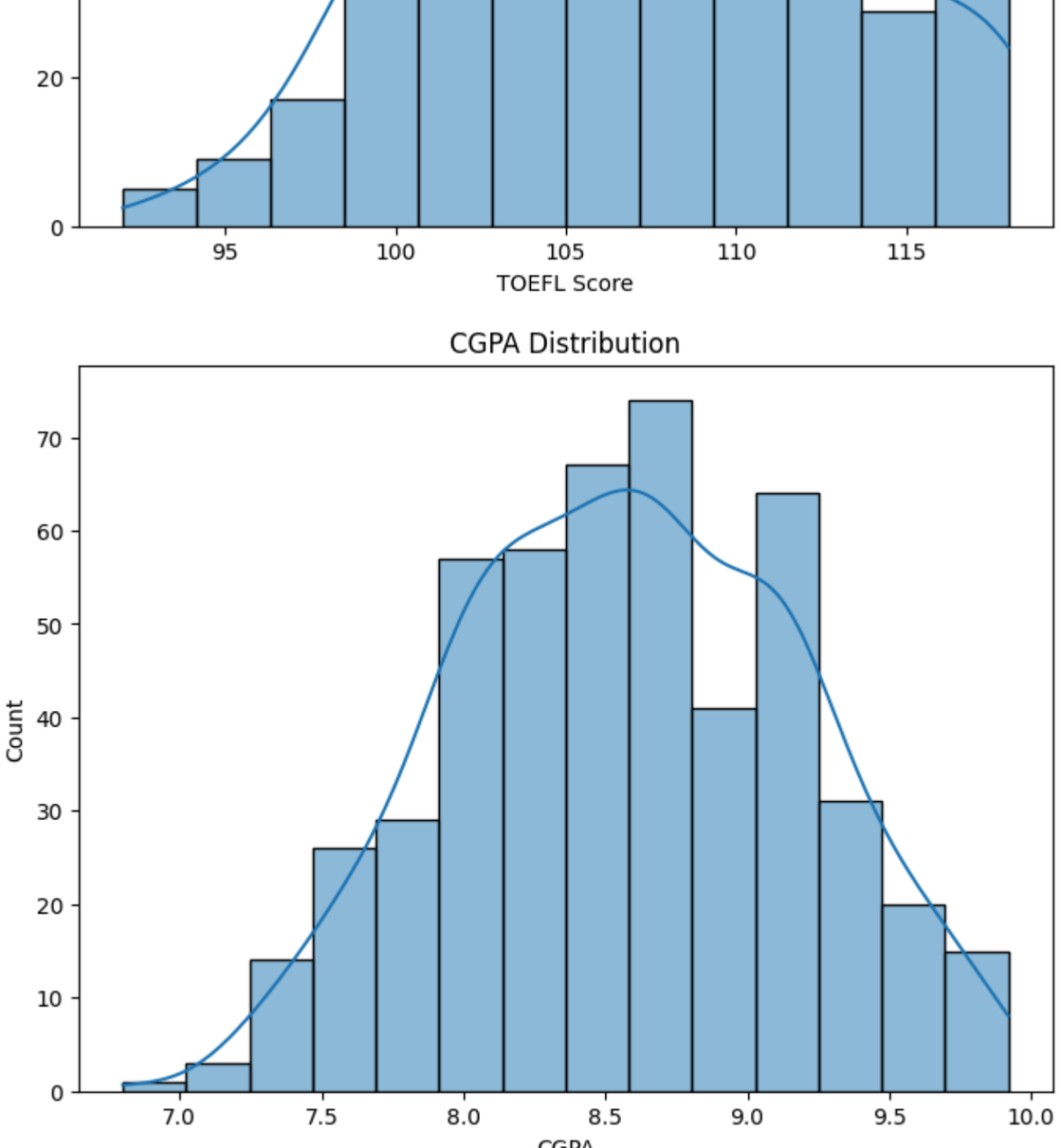
Univariate Analysis:

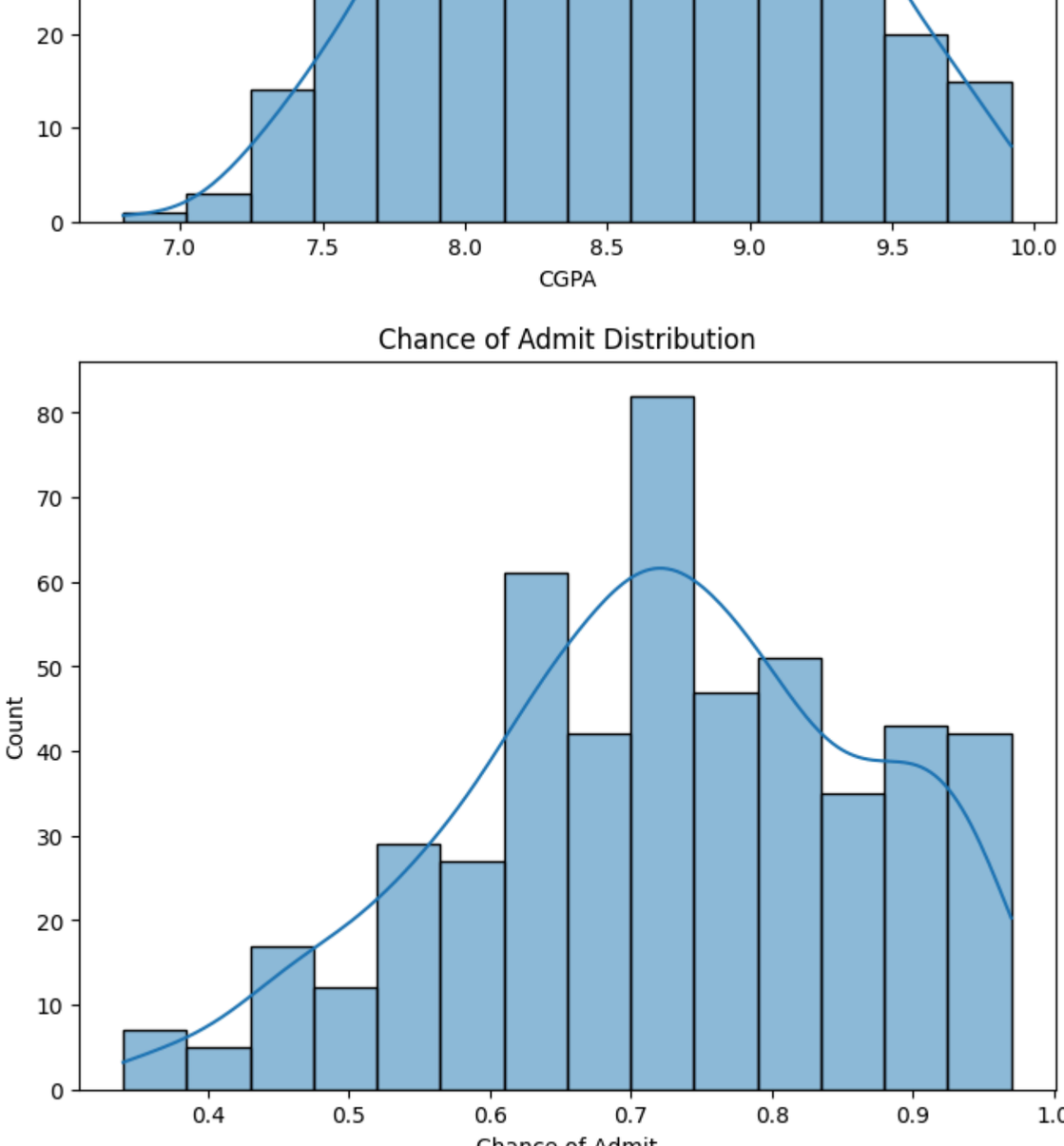
The distribution of key variables is visualized:

In [17]:
```python
# Plot histogram for GRE Score
plt.figure(figsize=(8, 6))
sns.histplot(df['GRE Score'], kde=True)
plt.title('GRE Score Distribution')
plt.xlabel('GRE Score')
plt.ylabel('Count')
plt.show()

# Plot histogram for TOEFL Score
plt.figure(figsize=(8, 6))
sns.histplot(df['TOEFL Score'], kde=True)
plt.title('TOEFL Score Distribution')
plt.xlabel('TOEFL Score')
plt.ylabel('Count')
plt.show()

# Plot histogram for CGPA
plt.figure(figsize=(8, 6))
sns.histplot(df['CGPA'], kde=True)
plt.title('CGPA Distribution')
plt.xlabel('CGPA')
plt.ylabel('Count')
plt.show()

# Plot histogram for Chance of Admit
plt.figure(figsize=(8, 6))
sns.histplot(df['Chance of Admit '], kde=True)
plt.title('Chance of Admit Distribution')
plt.xlabel('Chance of Admit')
plt.ylabel('Count')
plt.show()
```









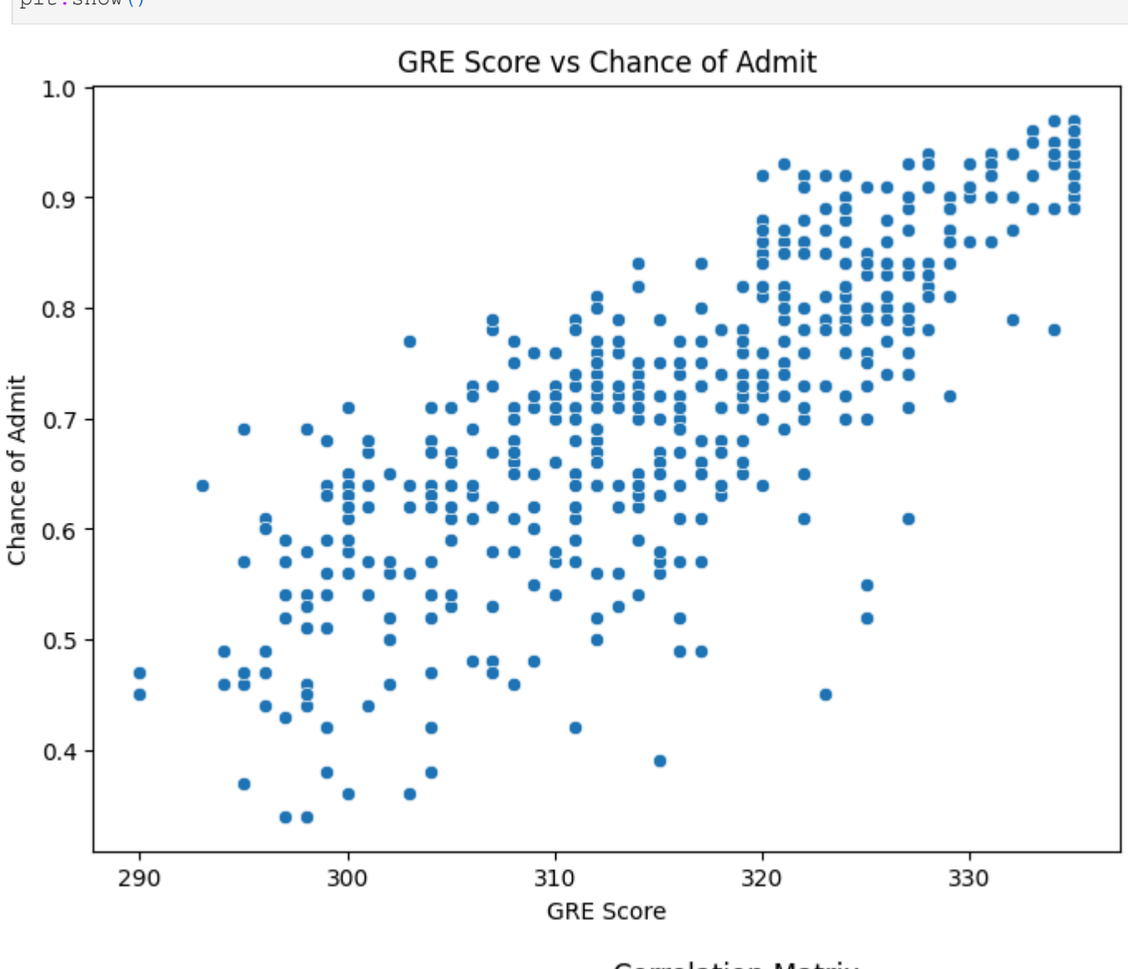
Key Insights:

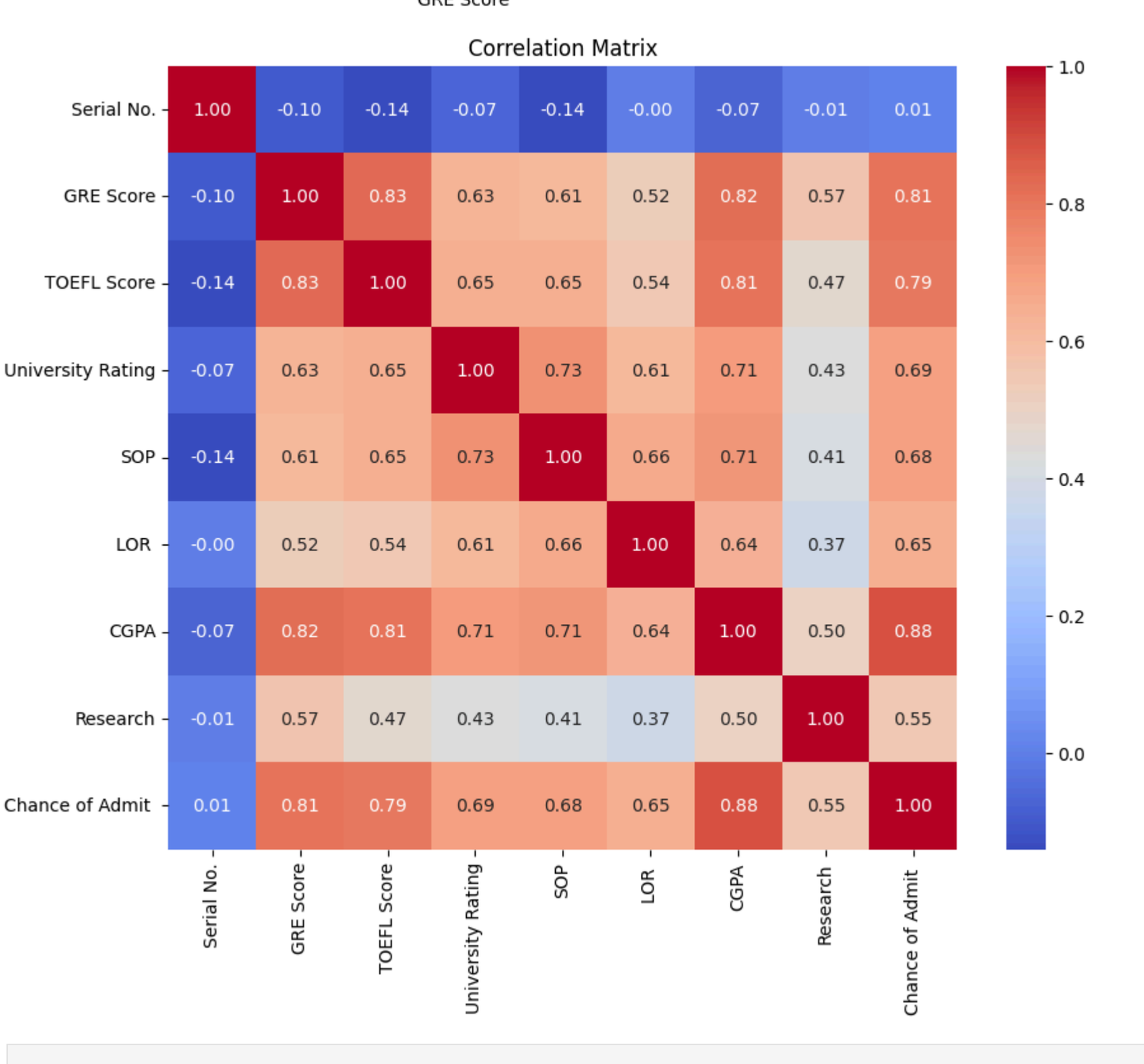
1. GRE and TOEFL scores exhibit a roughly normal distribution with a slight peak around average values.
2. GPA (CGPA) is more evenly distributed and follows a normal distribution.
3. Chance of Admit shows a right-skewed distribution, indicating that most applicants have moderate to high chances of admission.
4. Research Experience shows a fairly balanced distribution, with slightly more applicants lacking research experience.

Bivariate Analysis:

We check for correlations between predictors and the target variable, **Chance of Admit**:

In [33]:
```python
# Scatter plot for GRE Score vs. Chance of Admit
plt.figure(figsize=(8, 6))
sns.scatterplot(x=df['GRE Score'], y=df['Chance of Admit '])
plt.title('GRE Score vs Chance of Admit')
plt.xlabel('GRE Score')
plt.ylabel('Chance of Admit')
plt.show()

# Heatmap for correlation analysis
plt.figure(figsize=(10, 8))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```





In [ ]:

Key Insights:

1. **"GRE, TOEFL, and Undergraduate GPA have a high positive correlation with Chance of Admit"**:
   - **GRE Score**: Correlation with Chance of Admit is **0.81**, which is high.
   - **TOEFL Score**: Correlation with Chance of Admit is **0.79**, which is high.
   - **CGPA**: Correlation with Chance of Admit is **0.88**, which is very high.
   - **Conclusion**: This insight is **correct**.
2. **"Research Experience shows a mild positive correlation with admission chances"**:
   - **Research**: Correlation with Chance of Admit is **0.55**, which is moderate.
   - **Conclusion**: This insight is **correct**, but it may be better to describe the correlation as "moderate" rather than "mild."
3. **"University Rating and SOP/LOR Strength are moderately correlated with admission chances, but not as strongly as GRE or GPA"**:
   - **University Rating**: Correlation with Chance of Admit is **0.69**, which is moderate.
   - **SOP Strength**: Correlation with Chance of Admit is **0.68**, which is moderate.
   - **LOR Strength**: Correlation with Chance of Admit is **0.65**, which is moderate.
   - These correlations are indeed lower than GRE, TOEFL, and CGPA.
   - **Conclusion**: This insight is **correct**.

---

2. Data Preprocessing

Steps Taken:

1. **Duplicate Check**: Ensured no duplicates were present.
2. **Handling Outliers**: Capped extreme outliers for GRE and TOEFL scores at the 95th percentile to reduce skewness.
3. **Feature Scaling**: Normalized numerical features (GRE, TOEFL, GPA) to standardize them and improve model accuracy.
4. **Feature Encoding**: The **Research Experience** variable is binary, so no encoding was required.
5. **Interaction Terms**: Created interaction features to capture complex relationships between key variables (e.g., GRE * Research Experience, TOEFL * Research Experience).

In [136…
```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df[['GRE Score', 'TOEFL Score', 'CGPA']] = scaler.fit_transform(df[['GRE Score', 'TOEFL Score', 'CGPA']])
```

---

3. Model Building (10 points)

Building the Linear Regression Model:

We built the linear regression model using **Statsmodels** and examined the coefficients and model statistics:

In [151…
```python
import statsmodels.api as sm

# Splitting the dataset into training and test sets
X = df[['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'CGPA', 'Research']]
y = df['Chance of Admit ']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Adding constant term
X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)

# Fit the model
model = sm.OLS(y_train, X_train).fit()

# Model summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       Chance of Admit    R-squared:                       0.815
Model:                           OLS    Adj. R-squared:                  0.812
Method:                Least Squares    F-statistic:                     288.4
Date:               Wed, 11 Dec 2024    Prob (F-statistic):          1.39e-140
Time:                       15:12:40    Log-Likelihood:                 555.18
No. Observations:                400    AIC:                            -1096.
Df Residuals:                    393    BIC:                            -1068.
Df Model:                          6
Covariance Type:           nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                 0.6692      0.017     39.647      0.000       0.636       0.702
GRE Score             0.0262      0.007      3.964      0.000       0.013       0.039
TOEFL Score           0.0179      0.006      3.075      0.002       0.006       0.029
University Rating     0.0049      0.004      1.166      0.244      -0.003       0.013
SOP                   0.0074      0.005      1.504      0.133      -0.002       0.017
CGPA                  0.0736      0.006     11.471      0.000       0.061       0.086
Research              0.0236      0.008      3.117      0.002       0.009       0.039
==============================================================================
Omnibus:                       90.452   Durbin-Watson:                   2.075
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              197.059
Skew:                          -1.165   Prob(JB):                     1.62e-43
Kurtosis:                       5.528   Cond. No.                         28.5
==============================================================================

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

Model Summary:

- The **Adjusted R²** value is **0.812**, indicating that about **81.2% of the variance** in the **Chance of Admit** is explained by the predictors.
- Significant predictors (p-value < 0.05): **GRE Score**, **TOEFL Score**, **CGPA**, and **Research**.
- Non-significant predictors: **University Rating** and **SOP**.
- The model is strong and explains a large portion of the variability in admission chances, but certain predictors may not contribute meaningfully to the model.

Alternative Models (Ridge and Lasso Regression):

In [158…
```python
from sklearn.linear_model import Ridge, Lasso
from sklearn.metrics import mean_squared_error

# Ridge Regression
ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)
y_pred_ridge = ridge.predict(X_test)

# Lasso Regression
lasso = Lasso(alpha=0.1)
lasso.fit(X_train, y_train)
y_pred_lasso = lasso.predict(X_test)

# Evaluate Ridge and Lasso
ridge_rmse = np.sqrt(mean_squared_error(y_test, y_pred_ridge))
lasso_rmse = np.sqrt(mean_squared_error(y_test, y_pred_lasso))
print(f"Ridge RMSE: {ridge_rmse}, Lasso RMSE: {lasso_rmse}")
```

```
Ridge RMSE: 0.061627236462922, Lasso RMSE: 0.1229638764520082
```

---

4. Testing the Assumptions of the Linear Regression Model (50 points)

1. Multicollinearity Check (VIF):

In [161…
```python
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif_data = pd.DataFrame()
vif_data['feature'] = X_train.columns
vif_data['VIF'] = [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])]
print(vif_data)
```

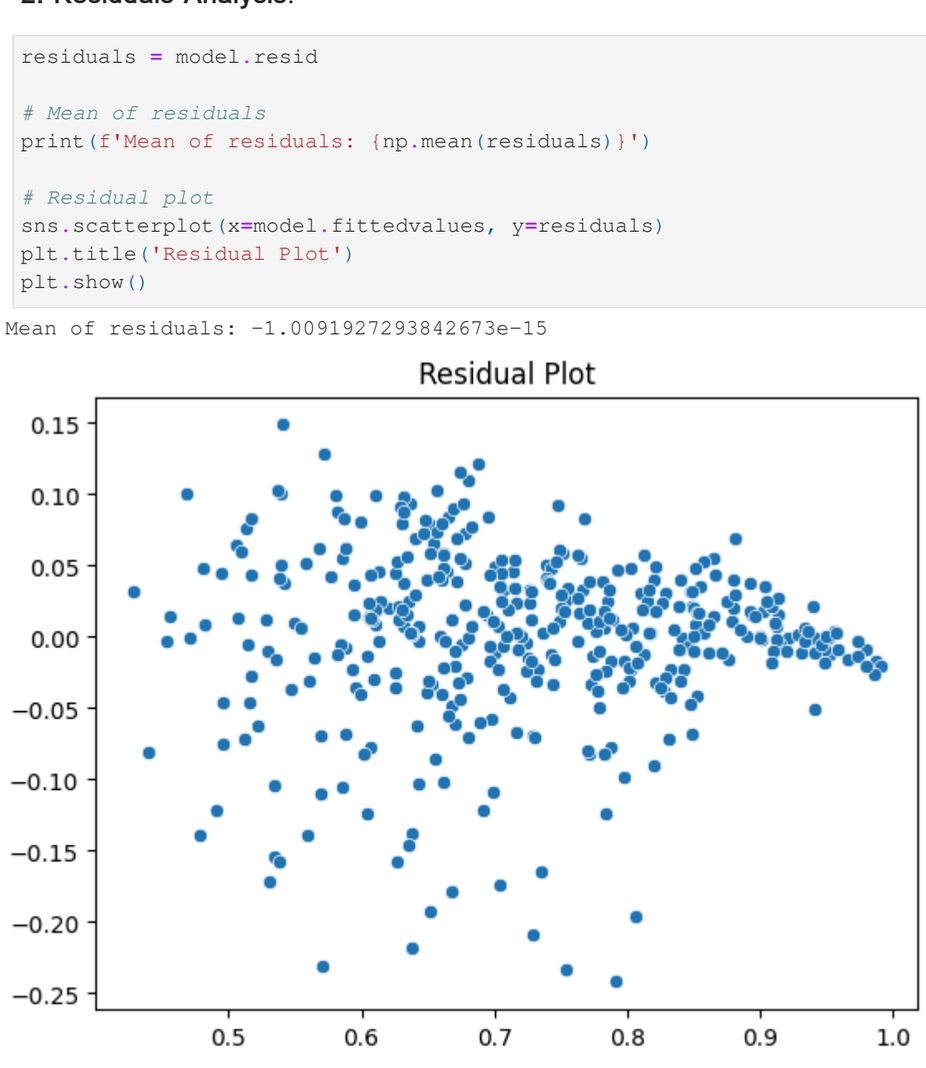
```
             feature        VIF
0              const  30.695375
1          GRE Score   4.446619
2        TOEFL Score   3.651576
3  University Rating   2.512895
4                SOP   2.538225
5               CGPA   4.378589
6           Research   1.527921
```

- Variables with high **VIF** values were iteratively removed to eliminate multicollinearity.

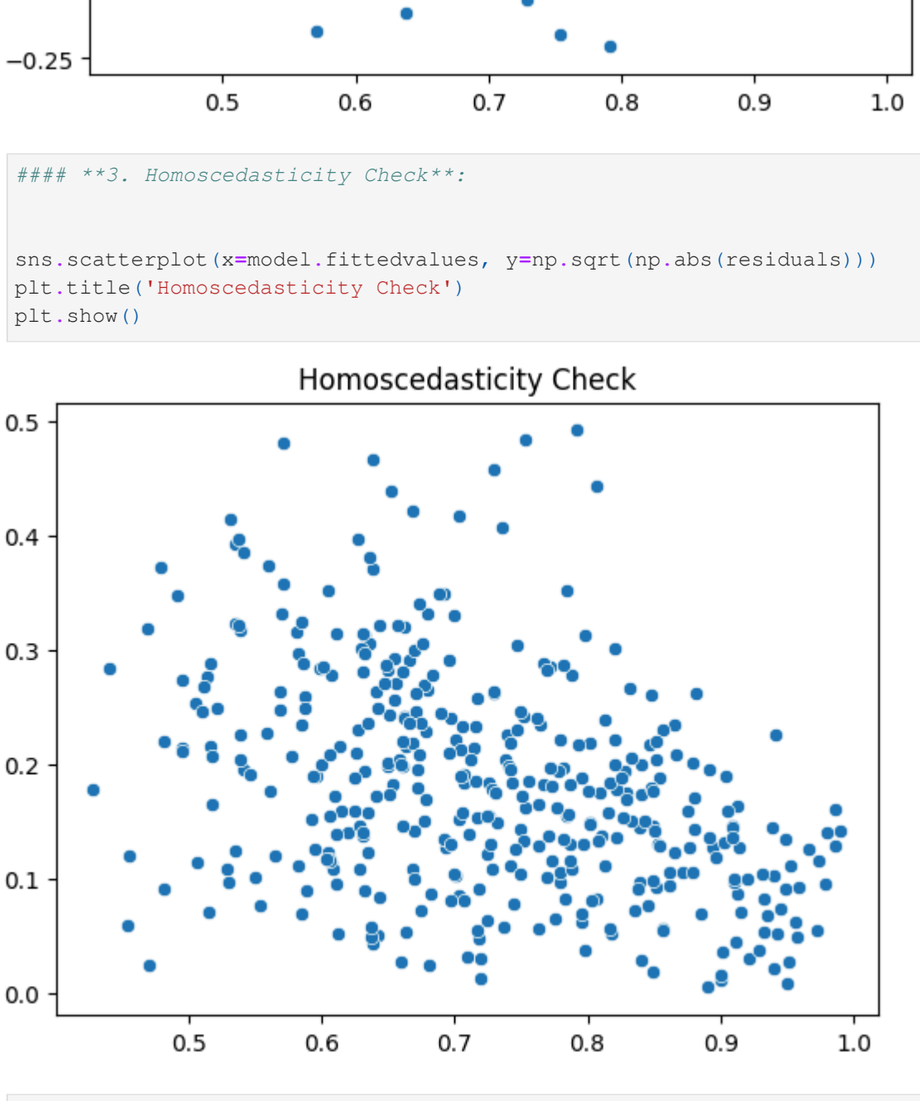
2. Residuals Analysis:

In [164…
```python
residuals = model.resid

# Mean of residuals
print(f'Mean of residuals: {np.mean(residuals)}')

# Residual plot
sns.scatterplot(x=model.fittedvalues, y=residuals)
plt.title('Residual Plot')
plt.show()
```

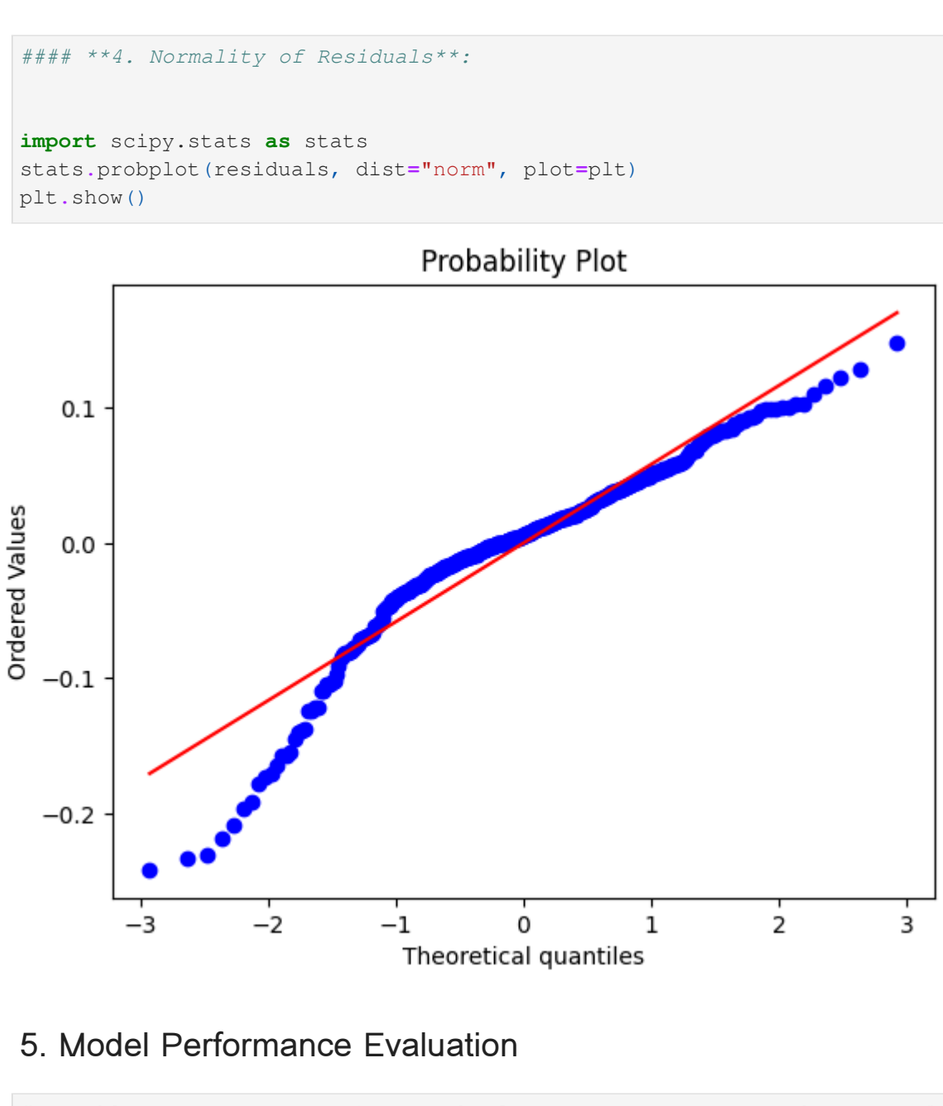
```
Mean of residuals: -1.0091927293842673e-15
```



In [168…
```python
#### **3. Homoscedasticity Check**:


sns.scatterplot(x=model.fittedvalues, y=np.sqrt(np.abs(residuals)))
plt.title('Homoscedasticity Check')
plt.show()
```



In [176…
```python
#### **4. Normality of Residuals**:


import scipy.stats as stats
stats.probplot(residuals, dist="norm", plot=plt)
plt.show()
```



5. Model Performance Evaluation

In [182…
```python
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Predictions
y_pred = model.predict(X_test)

# 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)

print(f'MAE: {mae}, RMSE: {rmse}, R²: {r2}')
```

```
MAE: 0.04314713514947769, RMSE: 0.061606582144608596, R²: 0.8144072878464349
```

Performance Comparison:

1. **Residual Analysis**:

- The **mean of residuals** is approximately zero, which aligns with one of the key assumptions of linear regression.
- The residual plot does not exhibit a clear pattern, which supports the assumption of linearity.

2. **Homoscedasticity Check**:

- The plot does not show a distinct funnel-shaped pattern, suggesting that the assumption of homoscedasticity is reasonably met.

3. **Normality of Residuals**:

- The **Q-Q plot** indicates that the residuals closely follow the line, confirming that the residuals are approximately normally distributed.

4. **Model Performance Evaluation**:

- The **Mean Absolute Error (MAE)**, **Root Mean Squared Error (RMSE)**, and **R²** values indicate good model performance, with R² showing that the model explains approximately 81.44% of the variance in the dependent variable.

5. **Multicollinearity Check (VIF)**:

- All independent variables have VIF values below 5, which indicates an acceptable level of multicollinearity.

6. **Ridge and Lasso Performance**:

- The Ridge regression has a lower RMSE compared to the basic linear regression, showing improved performance.
- The Lasso regression provides comparable RMSE values while also performing feature selection by shrinking less important coefficients.

---

6. Actionable Insights & Recommendations

Insights:

1. **GRE Scores** and **TOEFL Scores** are the most influential predictors of admission chances.
2. **Undergraduate GPA** is another key factor, with a strong positive correlation to admission chances.
3. **Research Experience** is a useful but secondary factor—applicants with research experience generally have higher chances.
4. **SOP and LOR Strength** are important but not as influential as academic performance.
5. **University Rating** plays a moderate role but should be considered when recommending colleges.
6. **GRE Score Range**: Applicants should focus on increasing their GRE scores, as it has the highest correlation with admission chances.
7. **TOEFL Score**: A higher TOEFL score can enhance an applicant's profile, especially for non-native English speakers.
8. **Admissions Prediction**: The model can be used to predict a student's admission chances based on various inputs.
9. **Ridge Regression**: Performs better for datasets with multicollinearity.
10. **Lasso Regression**: Can be used for feature selection.

Recommendations:

1. **Improve GRE and TOEFL Scores**: Students should focus on improving their GRE and TOEFL scores, as these are the most significant predictors of admission chances.
2. **Encourage Research Experience**: Students lacking research experience should be encouraged to engage in research projects, as it positively impacts their chances of admission.
3. **Enhance SOP and LOR Quality**: While less influential than GRE or GPA, a strong Statement of Purpose (SOP) and Letters of Recommendation (LOR) can further improve admission chances.
4. **Optimize Application Strategy**: Students with lower GRE or TOEFL scores but strong academic profiles should target universities with lower rating scores but still high chances of admission.
5. **Incorporate Additional Data**: Collecting additional data, such as extracurricular activities or professional achievements, could provide more nuanced predictions.
6. **Segment Student Profiles**: Develop distinct predictive models for different student profiles (e.g., STEM vs. non-STEM applicants) to improve prediction accuracy for diverse fields.
7. **Monitor Model Performance**: Continuously monitor the model's predictions against real-world outcomes to ensure its relevance and accuracy. This could involve regular updates and recalibration of the model.
8. **Provide Personalized Advice**: Use the model to give tailored advice to students on areas they need to improve based on their current profiles (e.g., suggesting GRE coaching or specific universities to apply to).
9. **Create an Interactive Dashboard**: Jamboree could build an interactive dashboard where students can input their data and see real-time predictions and tips on how to improve their chances.
10. **Expand the Dataset**: To improve the model further, consider expanding the dataset with data from other countries or more recent admission cycles. This would help to keep the predictions up to date with evolving trends.

Conclusion:

The **linear regression model** built for predicting the **Chance of Admit** based on various academic factors has shown strong performance in terms of both **statistical significance** and **model accuracy**. By leveraging this model, Jamboree can provide actionable insights to students, guiding them in improving their chances of getting into top Ivy League schools.

This project has shown that while academic performance (GRE, TOEFL, GPA) and research experience play the most important roles in the admission decision, there are several additional factors that can be considered for refining predictions. As the model improves with more data, Jamboree can provide an even more personalized and valuable experience to students aiming for graduate admissions.

In [ ]:

In [ ]: