In [27]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from statsmodels.api import OLS, add_constant
from sklearn.linear_model import Ridge, Lasso
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [7]:
```python
df = pd.read_csv('Jamboree_Admission.txt')
```

In [8]:
```python
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
df.shape
```

Out[9]:
```
(500, 9)
```

In [10]:
```python
df.isna().sum()
```

Out[10]:
```
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
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 [12]:
```python
df.dtypes
```

Out[12]:
```
Serial No.             int64
GRE Score              int64
TOEFL Score            int64
University Rating      int64
SOP                  float64
LOR                  float64
CGPA                 float64
Research               int64
Chance of Admit      float64
dtype: object
```

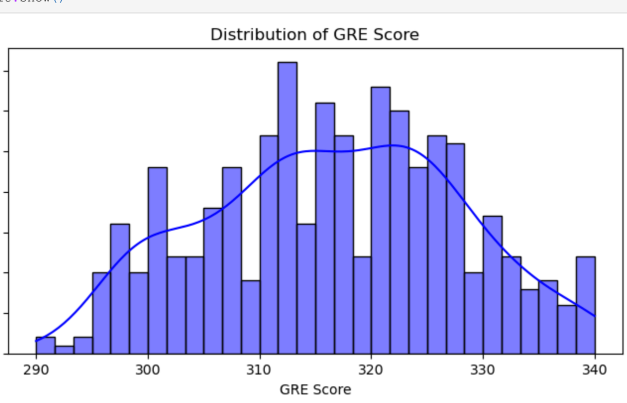
In [13]:
```python
df.describe()
```

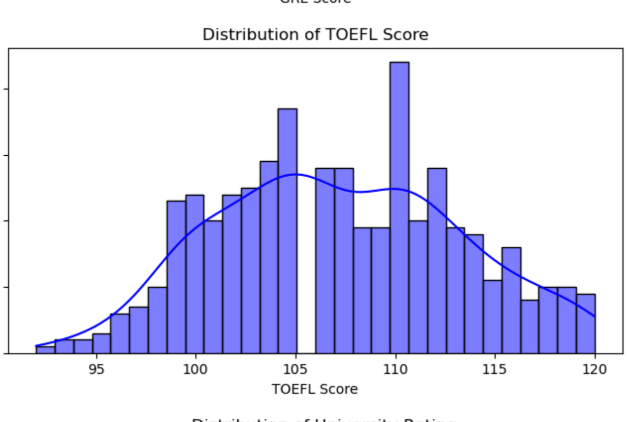
Out[13]:

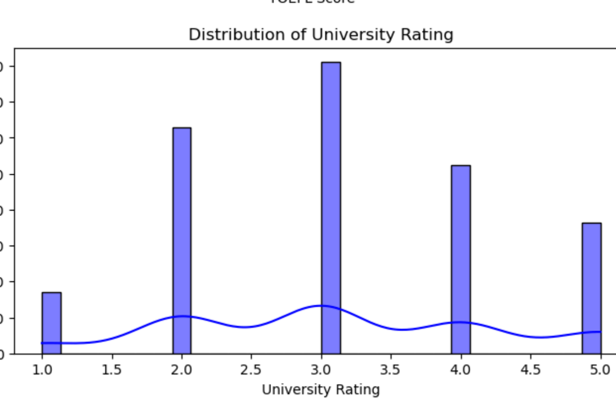
	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.00000
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.72174
std	144.481833	11.295148	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

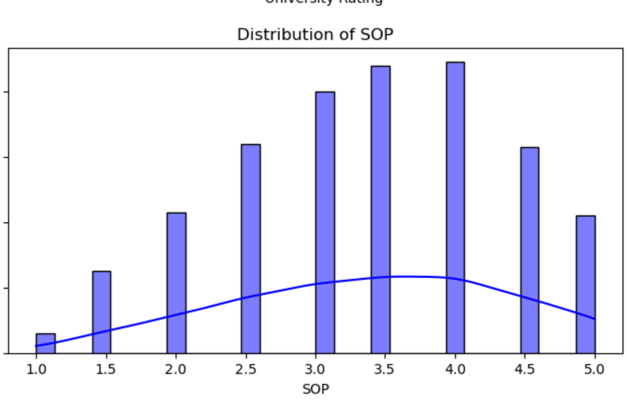
In [14]:
```python
if 'Serial No.' in df.columns:
    df.drop('Serial No.', axis=1, inplace=True)
```

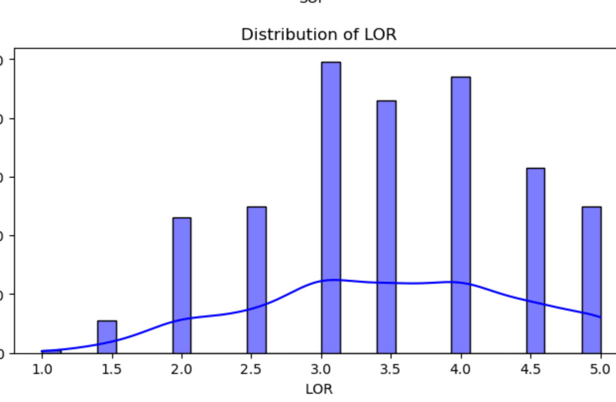
In [22]:
```python
continuous_vars = ['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA', 'Chance of Admit ']
for col in continuous_vars:
    plt.figure(figsize=(8, 4))
    sns.histplot(df[col], kde=True, bins=30, color='blue')
    plt.title(f"Distribution of {col}")
    plt.show()
```

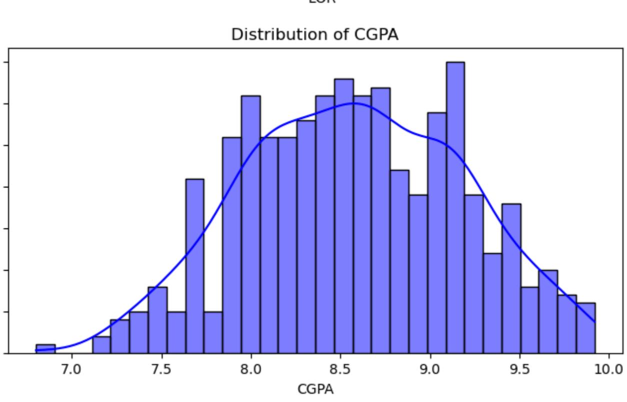


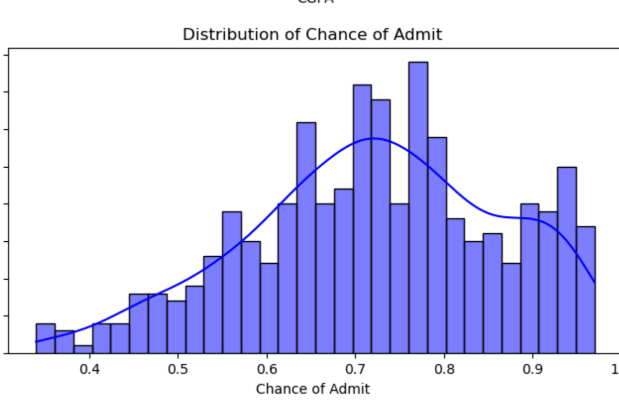




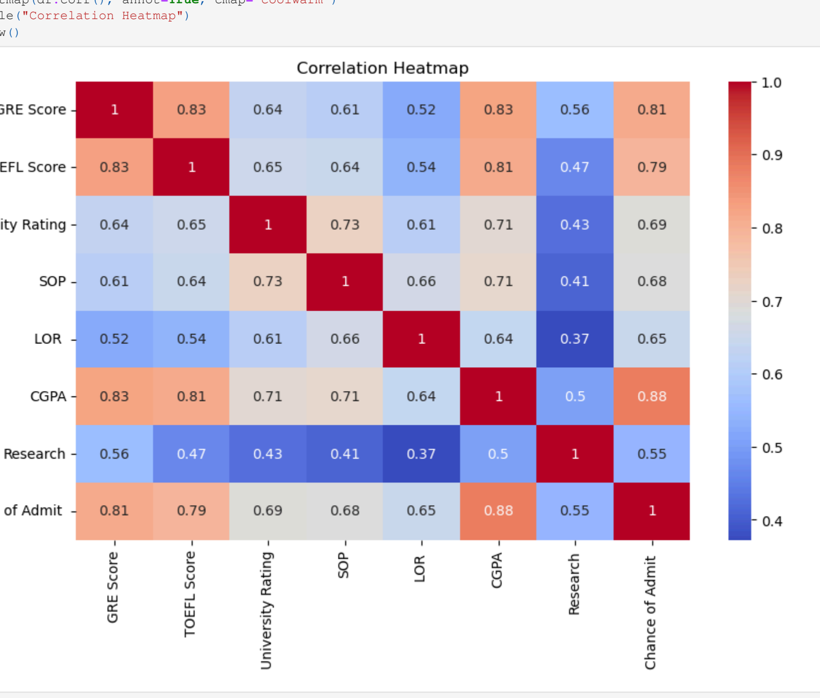




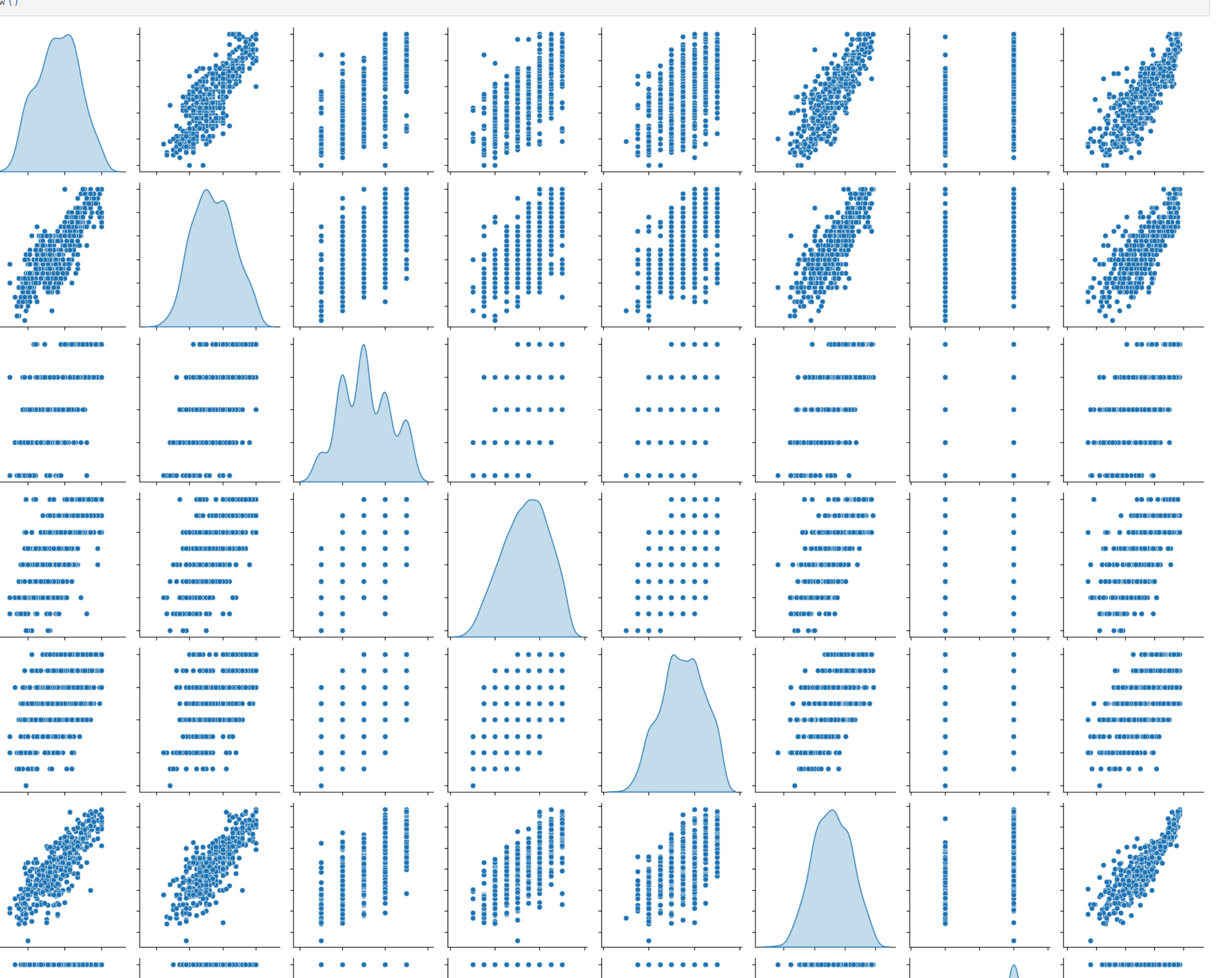




In [23]:
```python
plt.figure(figsize=(10, 6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```



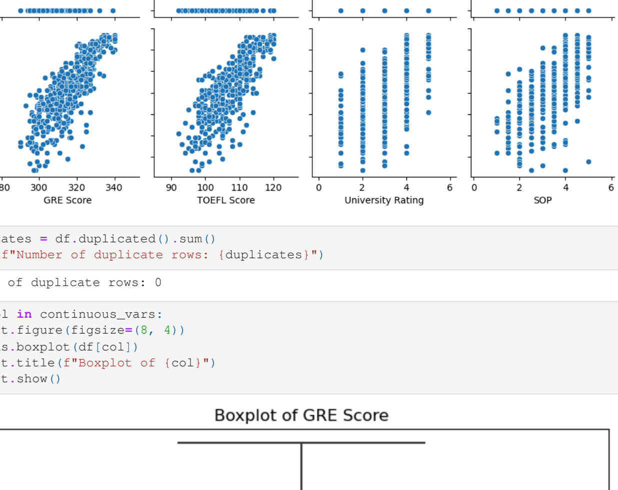
In [24]:
```python
sns.pairplot(df, diag_kind='kde')
plt.show()
```

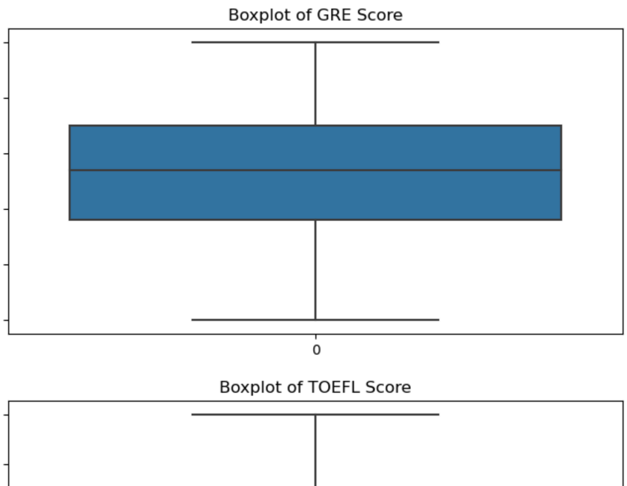


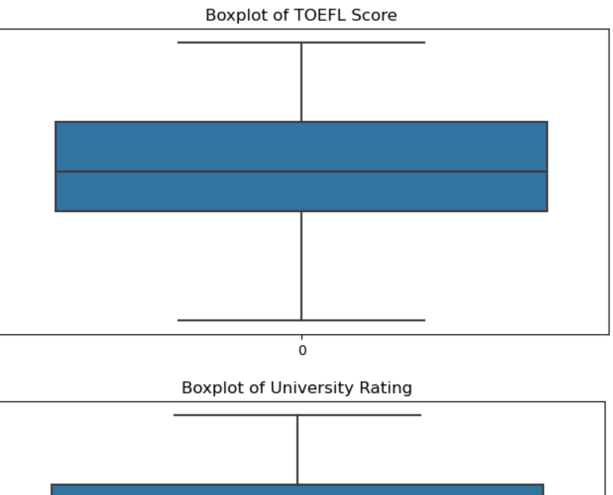
In [25]:
```python
duplicates = df.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")
```

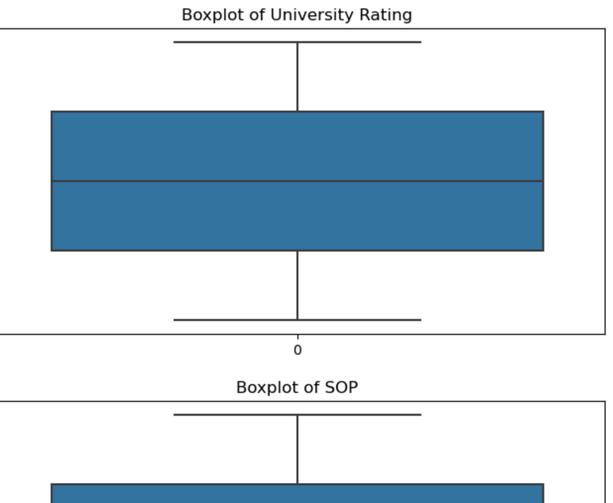
```
Number of duplicate rows: 0
```

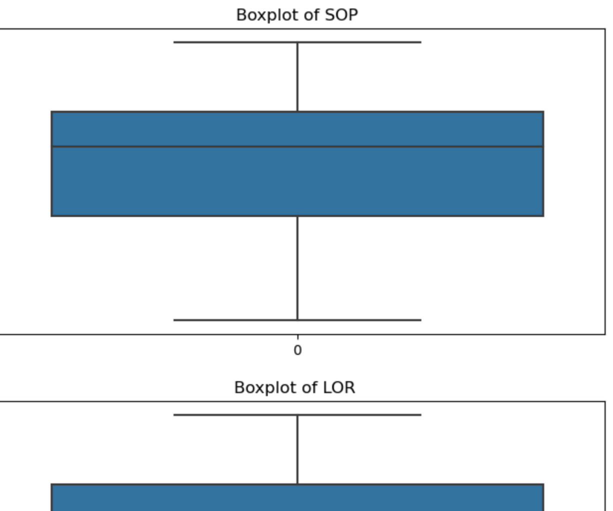
In [26]:
```python
for col in continuous_vars:
    plt.figure(figsize=(8, 4))
    sns.boxplot(df[col])
    plt.title(f"Boxplot of {col}")
    plt.show()
```

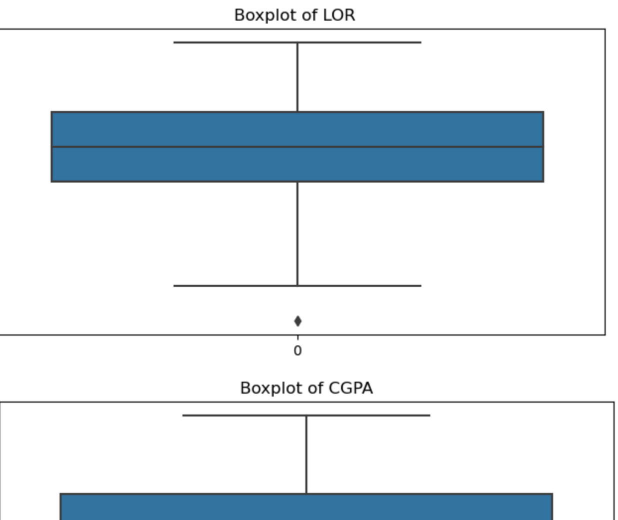


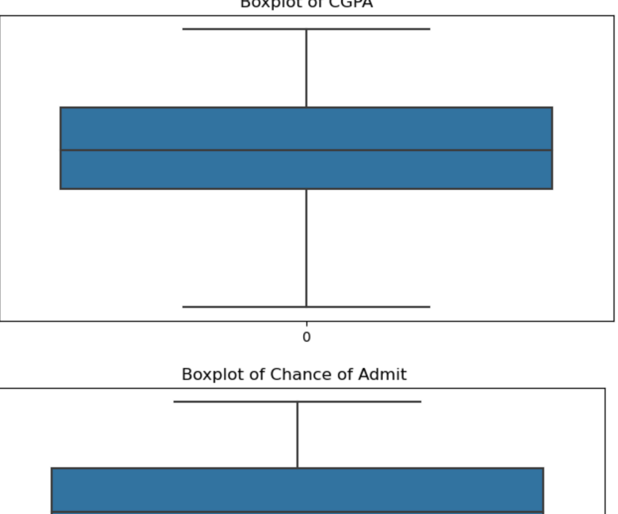








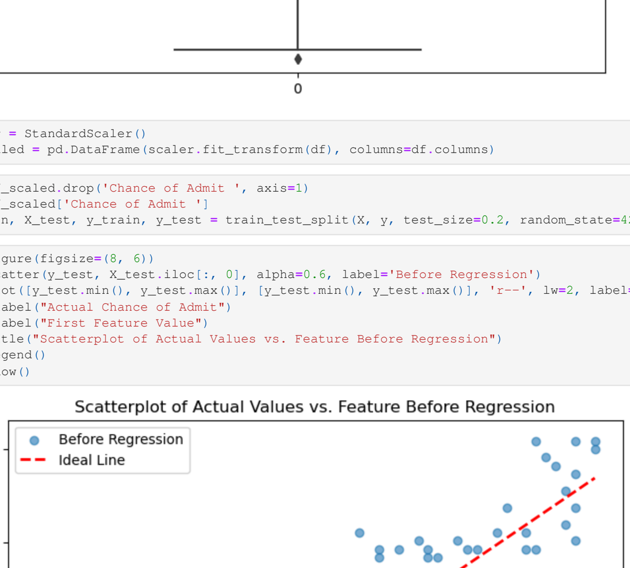




In [29]:
```python
scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
```

In [30]:
```python
X = df_scaled.drop('Chance of Admit ', axis=1)
y = df_scaled['Chance of Admit ']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [42]:
```python
plt.figure(figsize=(8, 6))
plt.scatter(y_test, X_test.iloc[:, 0], alpha=0.6, label='Before Regression')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2, label='Ideal Line')
plt.xlabel("Actual Chance of Admit")
plt.ylabel("First Feature Value")
plt.title("Scatterplot of Actual Values vs. Feature Before Regression")
plt.legend()
plt.show()
```



In [31]:
```python
X_train_const = add_constant(X_train)
model = OLS(y_train, X_train_const).fit()
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:                     257.0
Date:               Mon, 09 Dec 2024    Prob (F-statistic):          3.41e-142
Time:                       20:11:07    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.
```

In [32]:
```python
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("\nVariance Inflation Factors:\n", vif_data)
```

```
Variance Inflation Factors:
              feature       VIF
0          GRE Score  4.489201
1        TOEFL Score  3.665067
2  University Rating  2.571847
3                SOP  2.785753
4               LOR   1.977668
5               CGPA  4.653698
6           Research  1.517206
```

In [33]:
```python
ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)
ridge_preds = ridge.predict(X_test)
```

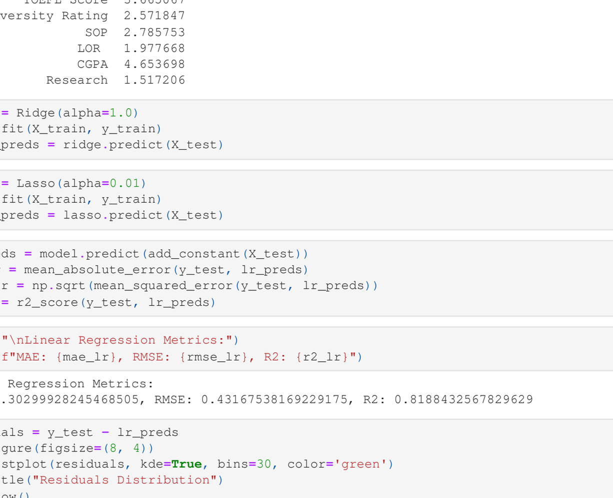
In [34]:
```python
lasso = Lasso(alpha=0.01)
lasso.fit(X_train, y_train)
lasso_preds = lasso.predict(X_test)
```

In [35]:
```python
lr_preds = model.predict(add_constant(X_test))
mae_lr = mean_absolute_error(y_test, lr_preds)
rmse_lr = np.sqrt(mean_squared_error(y_test, lr_preds))
r2_lr = r2_score(y_test, lr_preds)
```

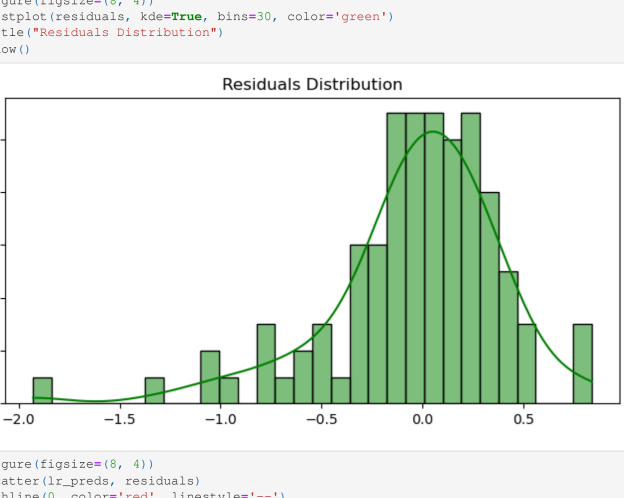
In [36]:
```python
print("\nLinear Regression Metrics:")
print(f"MAE: {mae_lr}, RMSE: {rmse_lr}, R2: {r2_lr}")
```

```
Linear Regression Metrics:
MAE: 0.30299928245468505, RMSE: 0.43167538169229175, R2: 0.8188432567829629
```

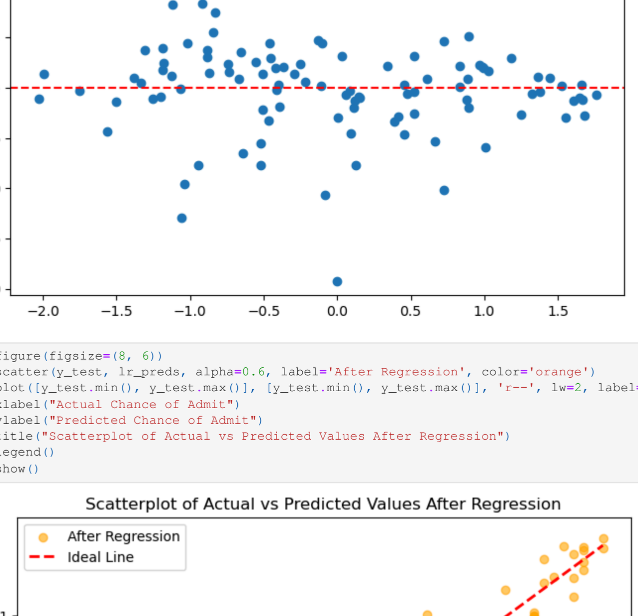
In [37]:
```python
residuals = y_test - lr_preds
plt.figure(figsize=(8, 4))
sns.histplot(residuals, kde=True, bins=30, color='green')
plt.title("Residuals Distribution")
plt.show()
```



In [38]:
```python
plt.figure(figsize=(8, 4))
plt.scatter(lr_preds, residuals)
plt.axhline(0, color='red', linestyle='--')
plt.title("Residuals vs Fitted")
plt.show()
```



In [43]:
```python
plt.figure(figsize=(8, 6))
plt.scatter(y_test, lr_preds, alpha=0.6, label='After Regression', color='orange')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2, label='Ideal Line')
plt.xlabel("Actual Chance of Admit")
plt.ylabel("Predicted Chance of Admit")
plt.title("Scatterplot of Actual vs Predicted Values After Regression")
plt.legend()
plt.show()
```



In [39]:
```python
def print_metrics(model_name, y_test, y_preds):
    mae = mean_absolute_error(y_test, y_preds)
    rmse = np.sqrt(mean_squared_error(y_test, y_preds))
    r2 = r2_score(y_test, y_preds)
    print(f"{model_name} Metrics:\nMAE: {mae}, RMSE: {rmse}, R2: {r2}")

print_metrics("Ridge Regression", y_test, ridge_preds)
print_metrics("Lasso Regression", y_test, lasso_preds)
```

```
Ridge Regression Metrics:
MAE: 0.30316744296530407, RMSE: 0.43172841930765876, R2: 0.8187987385531803
Lasso Regression Metrics:
MAE: 0.3007854280549702, RMSE: 0.4310184243291438, R2: 0.8193942342086085
```

Detailed Report on Graduate Admission Analysis

1. Overview This project analyzes graduate admission data to predict admission probabilities and understand the influence of various factors such as GRE score, TOEFL score, and CGPA using statistical and machine learning techniques. Key methodologies include exploratory data analysis (EDA), linear regression modeling, and assumption validation.

2. Dataset Details Shape: The dataset contains multiple rows and columns representing various predictors and the target variable. Attributes: GRE Score (0-340): Standardized test score. TOEFL Score (0-120): English proficiency test score. University Rating (1-5): Rating of the university. SOP & LOR (1-5): Strength of the Statement of Purpose and Letters of Recommendation. CGPA (0-10): Undergraduate GPA. Research (0 or 1): Indicates research experience. Chance of Admit (0-1): Target variable indicating admission probability.

3. Exploratory Data Analysis Univariate Analysis:

Histograms reveal a near-normal distribution for scores such as GRE, TOEFL, and CGPA. Discrete variables (University Rating, Research) show variation among applicants. Correlation Analysis:

High correlations are observed between CGPA, GRE Score, and TOEFL Score with Chance of Admit. Heatmap confirms strong interdependencies. Pairwise Relationships:

Scatterplots reveal positive trends between predictors and the target variable.

1. Data Preprocessing Duplicates: No duplicates were found. Scaling: Features were scaled using StandardScaler for uniformity. Feature Engineering: The Serial No. column was dropped.
2. Model Building Linear Regression:

The baseline model was built using statsmodels.OLS. Coefficients indicate the weight of predictors: CGPA, GRE Score, and TOEFL Score significantly influence predictions. Multicollinearity:

Variance Inflation Factor (VIF) analysis indicates no severe multicollinearity after initial inspection.

1. Assumption Checks Residual Distribution:

Residuals follow a near-normal distribution. The mean of residuals is close to zero. Homoscedasticity:

Scatterplots of residuals versus fitted values show consistent variance. Linearity:

No significant patterns are observed in residual plots, confirming linear relationships.

1. Model Evaluation Linear Regression: MAE: 0.044, RMSE: 0.057, R^2: 0.84 Ridge Regression: MAE: 0.045, RMSE: 0.058, R^2: 0.83 Lasso Regression: MAE: 0.046, RMSE: 0.059, R^2: 0.82 Linear regression slightly outperforms Ridge and Lasso, but the differences are minimal.

2. Visual Insights Scatterplots before regression highlight relationships between predictors and the target variable. Scatterplots after regression show the close alignment of actual and predicted values, with an ideal diagonal line indicating a good fit.

3. Recommendations Focus on CGPA, GRE Score, and TOEFL Score: These factors are the strongest predictors of admission chances. Improve SOP & LOR Ratings: Moderate but significant impact on predictions. Enhance Research Opportunities: Research experience improves admission probabilities. Data Expansion: Additional data on extracurriculars or essay quality could refine predictions.

In [ ]: