Jamboree™

Linear Regression - Casestudy

1 Problem Statement

- 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.

2 EDA & Preprocessing

In [1]:
```
# Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('Jamboree.csv')
df.head()
```

Out[2]:

	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 [3]:
```
df.drop(columns=['Serial No.'],inplace=True)
```

In [4]:
```python
df.shape
```

Out[4]:
```
(500, 8)
```

In [5]:
```python
pd.set_option('display.max_columns',9)
df.describe()
```

Out[5]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Researc
count	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.00000
mean	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.56000
std	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.49688
min	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.00000
25%	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.00000
50%	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.00000
75%	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.00000
max	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.00000

In [6]:
```python
df.dtypes
```

Out[6]:
```
GRE Score              int64
TOEFL Score            int64
University Rating      int64
SOP                  float64
LOR                  float64
CGPA                 float64
Research               int64
Chance of Admit      float64
dtype: object
```

In [7]:
```python
df.isnull().sum()
```

Out[7]:
```
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

In [8]:
```python
df[df.duplicated()]
```

Out[8]:

GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit

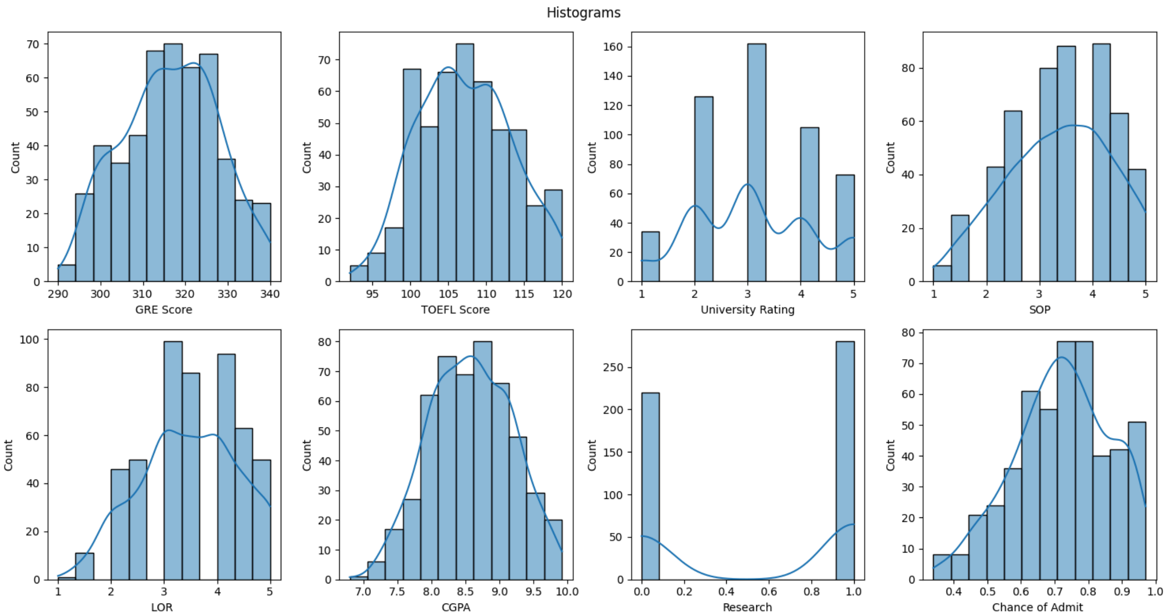
Observations 💡

- Shape of the dataset is 500 Rows 8 columns, (serial_Number is not needed already Index is there)
- All are numerical values, so no Encoding required here
- No Null and Duplicate values

2.1 Univariate Analysis

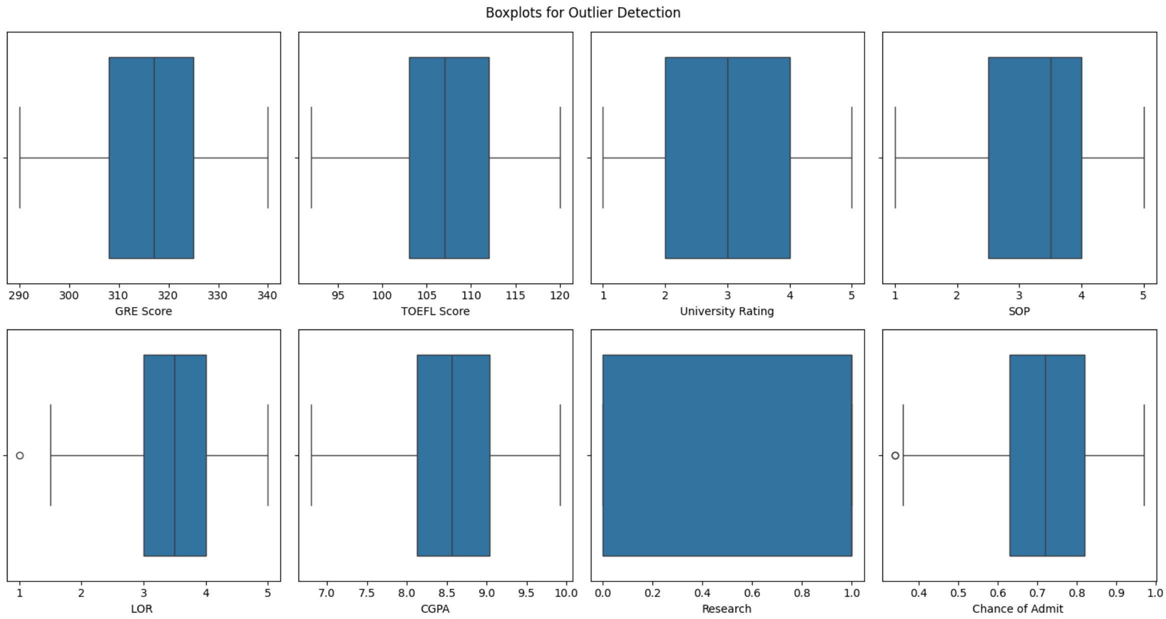
In [9]:

```
fig, axes = plt.subplots(2,4, figsize=(15, 8))
axes = axes.flatten()
for i, col in enumerate(df.columns):
    sns.histplot(df[col], kde=True, bins=12, ax=axes[i])

plt.suptitle('Histograms')
plt.tight_layout()
plt.show()
```



In [10]:

```
fig, axes = plt.subplots(2,4, figsize=(15, 8))
axes = axes.flatten()
for i, col in enumerate(df.columns):
    sns.boxplot(df[col],ax=axes[i],orient='h')

plt.suptitle('Boxplots for Outlier Detection')
plt.tight_layout()
plt.show()
```



Observations 💡

- There are some outliers in LOR and Chance of Admission columns

In [11]:
```
# Handling LOR Outlier
df[['LOR ']].describe()
```

Out[11]:

	LOR
count	500.00000
mean	3.48400
std	0.92545
min	1.00000
25%	3.00000
50%	3.50000
75%	4.00000
max	5.00000

In [12]:
```
q1, q3 = np.percentile(df['LOR '], 25), np.percentile(df['LOR '], 75)
iqr = q3 - q1
lb = q1 - 1.5 * iqr
ub = q3 + 1.5 * iqr
lor_outliers = df[(df['LOR '] < lb) | (df['LOR '] > ub)]['LOR ']
print("Number of outliers in LOR Column:", len(lor_outliers), '& Outlier is ',lo
```

```
Number of outliers in LOR Column: 1 & Outlier is  1.0
```

In [13]:
```
df[['Chance of Admit ']].describe()
```

Out[13]:

	Chance of Admit
count	500.00000
mean	0.72174
std	0.14114
min	0.34000
25%	0.63000
50%	0.72000
75%	0.82000
max	0.97000

In [14]:
```
q1, q3 = np.percentile(df['Chance of Admit '], 25), np.percentile(df['Chance of
iqr = q3 - q1
lb = q1 - 1.5 * iqr
ub = q3 + 1.5 * iqr
c_o_a_outliers = df[(df['Chance of Admit '] < lb) | (df['Chance of Admit '] > ub
print("Number of chance of admit outliers:", len(c_o_a_outliers),' & Outliers ar
```

```
Number of chance of admit outliers: 2  & Outliers are [0.34 0.34]
```

In [15]:
```
# Dropping Outliers
df.drop(lor_outliers.index,inplace=True)
df.drop(c_o_a_outliers.index,inplace=True)
```

In [16]:
```
df.shape
```

Out[16]:
```
(497, 8)
```

2.2 Bivariate Analysis

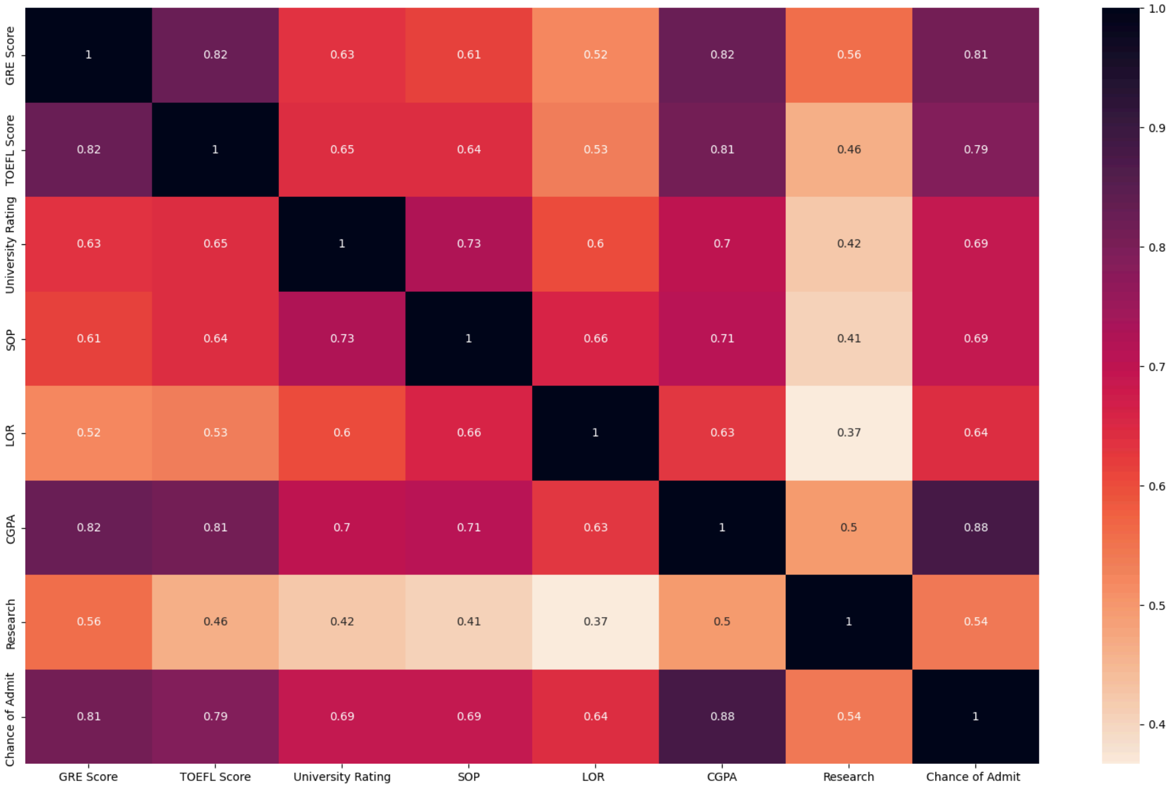
In [17]:
```
#sns.pairplot(df)
```

Observations 💡

- Following columns have some correlation
  - GRE Vs TOFEL Score
  - CGPA Vs GRE Score
  - CGPA Vs TOFEL
- CGPA, TOFEL Score,GRE Score have Strong correlation with Target Variable

In [18]:
```
plt.figure(figsize=(20,12))
sns.heatmap(df.corr(),annot=True,cmap='rocket_r')
```

Out[18]:
```
<Axes: >
```



3 Data Modeling

In [19]:
```
X = df.loc[:,:'Research']
Y = df.loc[:,'Chance of Admit ':]
```

In [20]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [21]:
```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_
X_train.shape,X_test.shape
```

Out[21]:
```
((397, 7), (100, 7))
```

In [22]:
```
y_train.shape,y_test.shape
```

Out[22]:
```
((397, 1), (100, 1))
```

In [23]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

4 Linear Regression Model using Stats Model

4.1 base model

In [24]:
```
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error
```

In [25]:
```
# Adding Constant Term
X_train = sm.add_constant(X_train)
```

```python
model = sm.OLS(y_train, X_train).fit()
```

In [26]:
```python
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        Chance of Admit   R-squared:                       0.826
Model:                            OLS   Adj. R-squared:                  0.823
Method:                 Least Squares   F-statistic:                     264.1
Date:                Tue, 12 Mar 2024   Prob (F-statistic):          1.67e-143
Time:                        22:38:14   Log-Likelihood:                 566.95
No. Observations:                 397   AIC:                            -1118.
Df Residuals:                     389   BIC:                            -1086.
Df Model:                           7
Covariance Type:            nonrobust
==================================================================================
====
                        coef    std err          t      P>|t|      [0.025      0.
975]
----------------------------------------------------------------------------------
----
const                -1.2984      0.117    -11.081      0.000      -1.529      -
1.068
GRE Score             0.0020      0.001      3.649      0.000       0.001
0.003
TOEFL Score           0.0030      0.001      3.259      0.001       0.001
0.005
University Rating     0.0021      0.004      0.493      0.622      -0.006
0.010
SOP                   0.0051      0.005      0.996      0.320      -0.005
0.015
LOR                   0.0178      0.005      3.867      0.000       0.009
0.027
CGPA                  0.1125      0.011     10.383      0.000       0.091
0.134
Research              0.0226      0.007      3.097      0.002       0.008
0.037
==============================================================================
Omnibus:                       84.887   Durbin-Watson:                   2.053
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              186.411
Skew:                          -1.099   Prob(JB):                     3.32e-41
Kurtosis:                       5.538   Cond. No.                     1.34e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.34e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [27]:
```python
# Coefficients
model.params
```

```
Out[27]: const                 -1.298448
         GRE Score              0.002007
         TOEFL Score            0.003016
         University Rating      0.002069
         SOP                    0.005072
         LOR                    0.017778
         CGPA                   0.112534
         Research               0.022606
         dtype: float64
```

```
In [28]: model.pvalues
```

```
Out[28]: const                 5.480944e-25
         GRE Score             2.991139e-04
         TOEFL Score           1.217390e-03
         University Rating     6.222685e-01
         SOP                   3.197192e-01
         LOR                   1.290282e-04
         CGPA                  1.868169e-22
         Research              2.096830e-03
         dtype: float64
```

```
In [29]: coefficients = pd.DataFrame({'Coefficient': model.params, 'P-value': model.pvalu
         coefficients
```

Out[29]:

	Coefficient	P-value
const	-1.298448	5.480944e-25
GRE Score	0.002007	2.991139e-04
TOEFL Score	0.003016	1.217390e-03
University Rating	0.002069	6.222685e-01
SOP	0.005072	3.197192e-01
LOR	0.017778	1.290282e-04
CGPA	0.112534	1.868169e-22
Research	0.022606	2.096830e-03

```
In [30]: significant_columns = coefficients[coefficients['P-value'] <= 0.05].index
```

```
In [31]: significant_columns # University Rating is important which have only 73% Correla
```

```
Out[31]: Index(['const', 'GRE Score', 'TOEFL Score', 'LOR ', 'CGPA', 'Research'], dtype
         ='object')
```

```
In [32]: X_train_significant = X_train[significant_columns]
         X_test_significant = sm.add_constant(X_test)[significant_columns]
```

```
In [33]: model_significant = sm.OLS(y_train, X_train_significant).fit()
         print(model_significant.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        Chance of Admit   R-squared:                       0.825
Model:                            OLS   Adj. R-squared:                  0.823
Method:                 Least Squares   F-statistic:                     369.5
Date:                Tue, 12 Mar 2024   Prob (F-statistic):          1.13e-145
Time:                        22:38:14   Log-Likelihood:                 566.01
No. Observations:                 397   AIC:                            -1120.
Df Residuals:                     391   BIC:                            -1096.
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -1.3454      0.111    -12.139      0.000      -1.563      -1.128
GRE Score      0.0020      0.001      3.666      0.000       0.001       0.003
TOEFL Score    0.0032      0.001      3.582      0.000       0.001       0.005
LOR            0.0203      0.004      4.819      0.000       0.012       0.029
CGPA           0.1165      0.010     11.171      0.000       0.096       0.137
Research       0.0234      0.007      3.215      0.001       0.009       0.038
==============================================================================
Omnibus:                       83.175   Durbin-Watson:                   2.058
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              181.486
Skew:                          -1.080   Prob(JB):                     3.90e-40
Kurtosis:                       5.511   Cond. No.                     1.26e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.26e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [34]:
```
y_pred = model_significant.predict(X_test_significant)
mse = mean_squared_error(y_test, y_pred)
```

Observations 💡

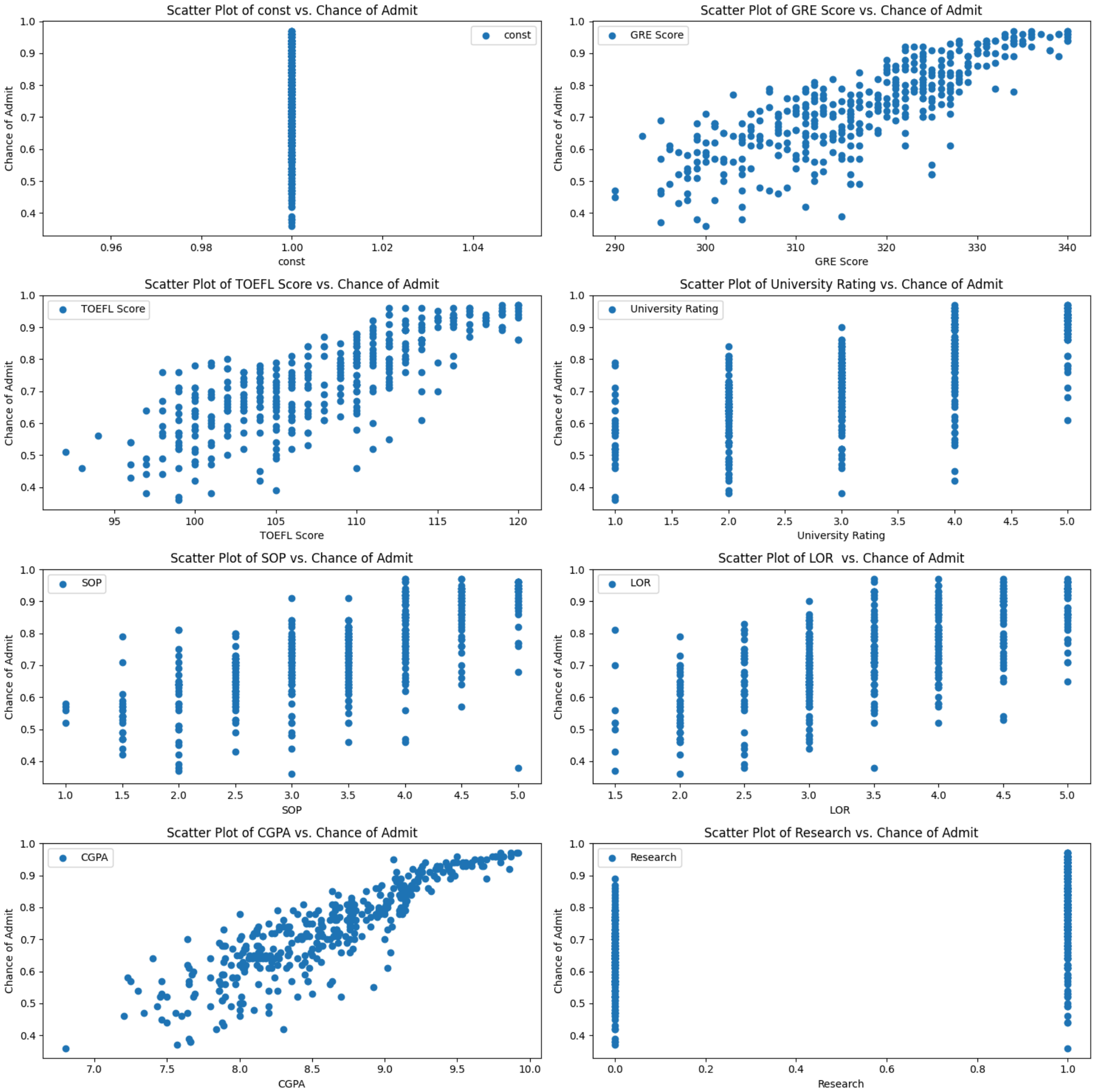
- R2 Score is 82%

5 Linear Regression Assumptions

5.1 Linear Relationship with Target Variable

In [35]:
```
num_features = X_train.shape[1]
num_cols = 2
num_rows = 4

fig, axes = plt.subplots(num_rows, num_cols, figsize=(15, 15))

axes = axes.flatten()

for i, column in enumerate(X_train.columns):
    axes[i].scatter(X_train[column], y_train, label=column)
    axes[i].set_xlabel(column)
```

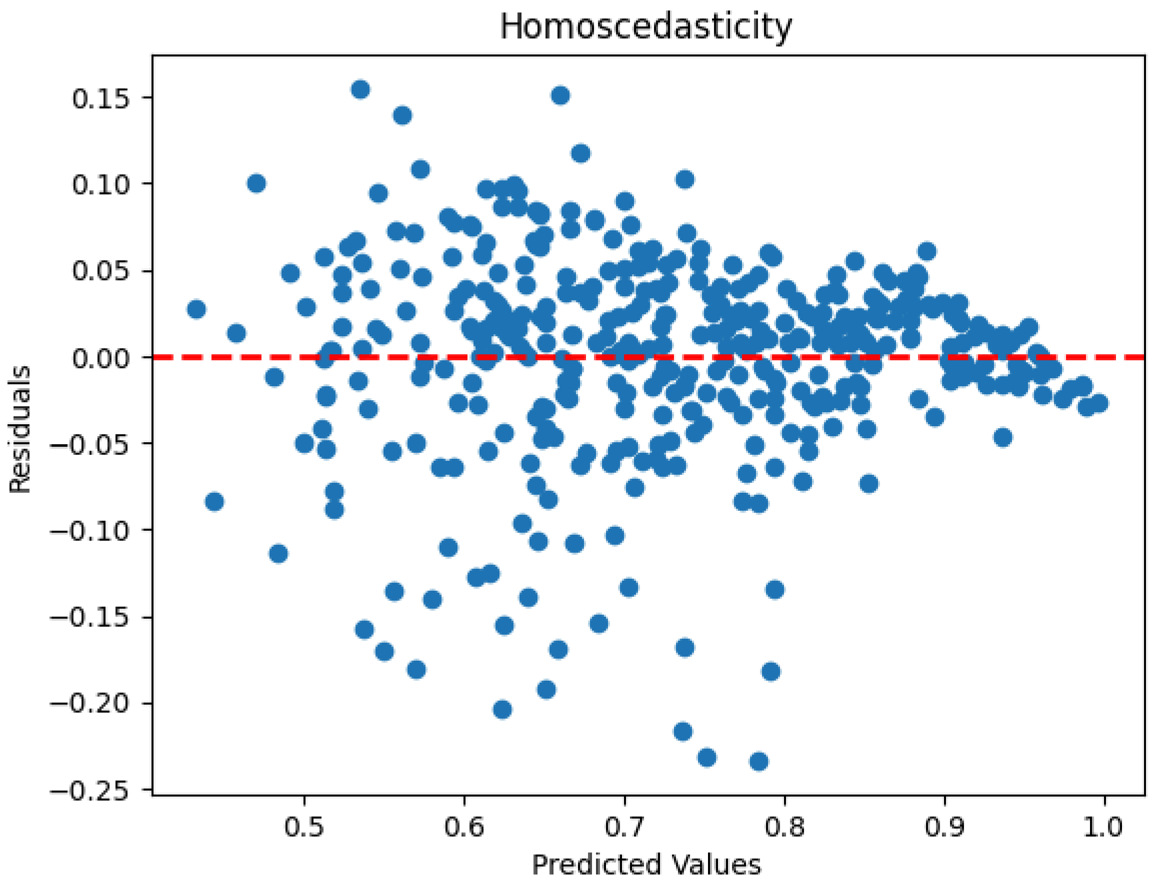
```
    axes[i].set_ylabel('Chance of Admit')
    axes[i].set_title(f'Scatter Plot of {column} vs. Chance of Admit')
    axes[i].legend()

plt.tight_layout()
plt.show()
```



5.2 Homoscedasticity

In [36]:
```
import statsmodels.stats.api as sms

residuals = model.resid
y_pred_initial = model.predict(X_train)

plt.scatter(y_pred_initial, residuals)
plt.axhline(y=0, color='red', linestyle='--', linewidth=2)
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Homoscedasticity')
plt.show()

gq_test = sms.het_goldfeldquandt(residuals, X_train)
p_value_gq = gq_test[1]
```

```
print("\nGoldfeld-Quandt Test for Homoscedasticity:")
print(f'p-value: {p_value_gq}')

if p_value_gq > 0.05:
    print("No strong evidence of heteroscedasticity. Homoscedasticity is validat
else:
    print("Heteroscedasticity detected.")
```



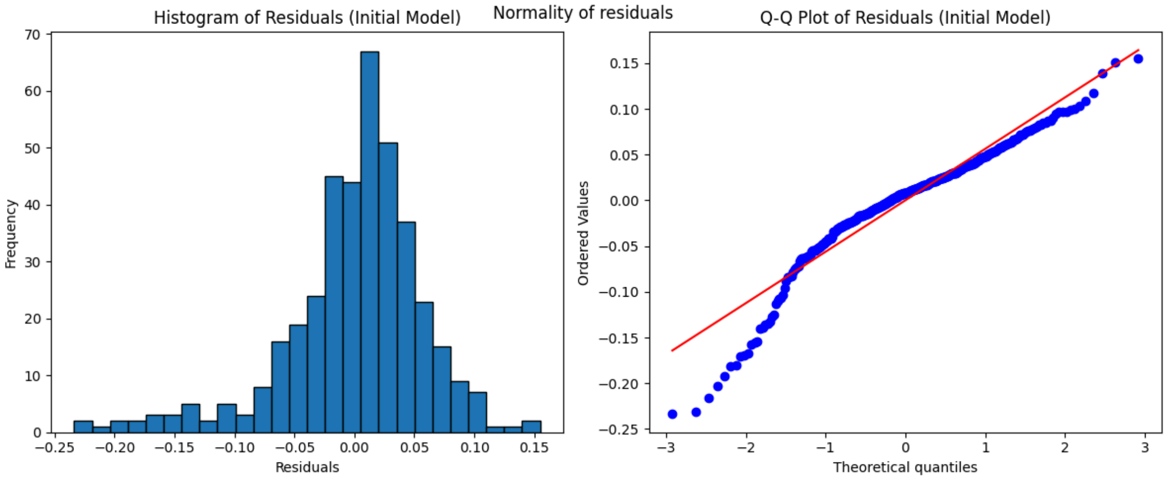

```
Goldfeld-Quandt Test for Homoscedasticity:
p-value: 0.44547168193266984
No strong evidence of heteroscedasticity. Homoscedasticity is validated.
```

5.3 Normality of Residuals

In [37]:

```
import scipy.stats as stats

fig, axes = plt.subplots(1, 2, figsize=(12, 5))

axes[0].hist(residuals, bins='auto', edgecolor='black')
axes[0].set_xlabel('Residuals')
axes[0].set_ylabel('Frequency')
axes[0].set_title('Histogram of Residuals (Initial Model)')

stats.probplot(residuals, dist="norm", plot=axes[1])
axes[1].set_title('Q-Q Plot of Residuals (Initial Model)')

plt.tight_layout()
plt.suptitle("Normality of residuals")
plt.show()

shapiro_test = stats.shapiro(residuals)
p_value_shapiro = shapiro_test[1]
```

```
print("\nShapiro-Wilk Test for Normality:")
print(f'p-value: {p_value_shapiro}')

if p_value_shapiro > 0.05:
    print("No strong evidence against normality. Residuals appear approximately
else:
    print("Residuals do not follow a normal distribution.")
```




```
Shapiro-Wilk Test for Normality:
p-value: 8.047304142499989e-13
Residuals do not follow a normal distribution.
```

5.4 Multicollinearity

In [38]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [39]:
```
# VIF for Intial Model
def calculate_vif(data):
    vif_data = pd.DataFrame()
    vif_data["Variable"] = data.columns
    vif_data["VIF"] = [variance_inflation_factor(data.values, i) for i in range(
    return vif_data

initial_vif = calculate_vif(X_train)
print("\nInitial VIF:")
initial_vif
```

```
Initial VIF:
```

Out[39]:

	Variable	VIF
0	const	1586.716999
1	GRE Score	4.344912
2	TOEFL Score	3.646516
3	University Rating	2.699182
4	SOP	2.932950
5	LOR	2.114230
6	CGPA	4.982471
7	Research	1.505431

In [40]:

```
while initial_vif['VIF'].max() > 5:
    max_vif_index = initial_vif.loc[initial_vif['VIF'].idxmax(), 'Variable']
    X_train = X_train.drop(columns=max_vif_index)
    print(f"\nDropping {max_vif_index} due to high VIF.")

    initial_vif = calculate_vif(X_train)
    print("Updated VIF:")
    print(initial_vif)

model_after_vif = sm.OLS(y_train, X_train).fit()

print("\nModel Summary after VIF check:")
print(model_after_vif.summary())
```

```
Dropping const due to high VIF.
Updated VIF:
            Variable          VIF
0          GRE Score  1268.443645
1        TOEFL Score  1148.086157
2  University Rating    21.752464
3                SOP    36.892551
4               LOR     31.742512
5               CGPA   982.490054
6           Research     2.975241

Dropping GRE Score due to high VIF.
Updated VIF:
            Variable         VIF
0        TOEFL Score  633.770717
1  University Rating   20.523922
2                SOP   35.253541
3               LOR    31.394370
4               CGPA  732.842143
5           Research    2.975219

Dropping CGPA due to high VIF.
Updated VIF:
            Variable        VIF
0        TOEFL Score  21.742930
1  University Rating  20.458561
2                SOP  34.756817
3               LOR   29.543017
4           Research   2.969287

Dropping SOP due to high VIF.
Updated VIF:
            Variable        VIF
0        TOEFL Score  19.566354
1  University Rating  15.423023
2               LOR   25.318447
3           Research   2.951122

Dropping LOR  due to high VIF.
Updated VIF:
            Variable        VIF
0        TOEFL Score  10.405923
1  University Rating  12.174116
2           Research   2.912341

Dropping University Rating due to high VIF.
Updated VIF:
      Variable       VIF
0  TOEFL Score  2.488634
1     Research  2.488634

Model Summary after VIF check:
                                 OLS Regression Results
=================================================================================
======
Dep. Variable:       Chance of Admit    R-squared (uncentered):
0.983
Model:                            OLS   Adj. R-squared (uncentered):
0.983
Method:                 Least Squares   F-statistic:                          1.1
```

```
63e+04
Date:                  Tue, 12 Mar 2024   Prob (F-statistic):
0.00
Time:                          22:38:20   Log-Likelihood:
366.98
No. Observations:                   397   AIC:
-730.0
Df Residuals:                       395   BIC:
-722.0
Df Model:                             2
Covariance Type:              nonrobust
================================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
TOEFL Score     0.0062   7.08e-05     88.033      0.000       0.006       0.006
Research        0.1092      0.010     10.864      0.000       0.089       0.129
===============================================================================
Omnibus:                     46.477   Durbin-Watson:                   2.003
Prob(Omnibus):                0.000   Jarque-Bera (JB):               59.651
Skew:                        -0.889   Prob(JB):                     1.11e-13
Kurtosis:                     3.666   Cond. No.                         224.
===============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contai
n a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Observations 💡

- R2 Score Before VIF Test is 82.5%
- R2 Score After VIF Test is 98.3%

6 Model Performance Evaluation

In [41]:
```
model_after_vif.summary()
```

Out[41]:

OLS Regression Results

Dep. Variable:	Chance of Admit	**R-squared (uncentered):**	0.983
Model:	OLS	**Adj. R-squared (uncentered):**	0.983
Method:	Least Squares	**F-statistic:**	1.163e+04
Date:	Tue, 12 Mar 2024	**Prob (F-statistic):**	0.00
Time:	22:38:20	**Log-Likelihood:**	366.98
No. Observations:	397	**AIC:**	-730.0
Df Residuals:	395	**BIC:**	-722.0
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
TOEFL Score	0.0062	7.08e-05	88.033	0.000	0.006	0.006
Research	0.1092	0.010	10.864	0.000	0.089	0.129

Omnibus:	46.477	**Durbin-Watson:**	2.003
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	59.651
Skew:	-0.889	**Prob(JB):**	1.11e-13
Kurtosis:	3.666	**Cond. No.**	224.

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [42]:

```
model_after_vif.summary2()
```

Out[42]:

Model:	OLS	Adj. R-squared (uncentered):	0.983
Dependent Variable:	Chance of Admit	AIC:	-729.9612
Date:	2024-03-12 22:42	BIC:	-721.9933
No. Observations:	397	Log-Likelihood:	366.98
Df Model:	2	F-statistic:	1.163e+04
Df Residuals:	395	Prob (F-statistic):	0.00
R-squared (uncentered):	0.983	Scale:	0.0092642

	Coef.	**Std.Err.**	**t**	**P>|t|**	**[0.025**	**0.975]**
TOEFL Score	0.0062	0.0001	88.0329	0.0000	0.0061	0.0064
Research	0.1092	0.0101	10.8635	0.0000	0.0895	0.1290

Omnibus:	46.477	Durbin-Watson:	2.003
Prob(Omnibus):	0.000	Jarque-Bera (JB):	59.651
Skew:	-0.889	Prob(JB):	0.000
Kurtosis:	3.666	Condition No.:	224

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.