# Problem Statement

A higher education consulting firm, has provided a list of students' performance metrics such as GRE test score (out of 340), TOEFL test score (out of 120), university rating (out of 5.0), undergrad GPA (out of 10), score of recommendation letters and Statement of Purpose (out of 5), research experience (0 or 1), and their chance of getting admit (0 to 1 range). We need to analyze which performance metrics were more important in predicting the chances of getting an admit, to help in predicting the admit chances for future students.

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

In [2]:
```
# import warnings
# warnings.filterwarnings("ignore")
```

In [3]:
```
df = pd.read_csv('Jamboree_Admission.csv')
df.head()
```

Out[3]:

| | 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 [4]:
```
df.drop(['Serial No.'],axis=1,inplace=True) # deleting the serial no. column as it is not requir
```

In [5]:
```
df['Research'].replace([0,1],["no","yes"],inplace=True)
df.head()
```

Out[5]:

| | GRE Score | TOEFL Score | University Rating | SOP | LOR | CGPA | Research | Chance of Admit |
|---|---|---|---|---|---|---|---|---|
| **0** | 337 | 118 | 4 | 4.5 | 4.5 | 9.65 | yes | 0.92 |
| **1** | 324 | 107 | 4 | 4.0 | 4.5 | 8.87 | yes | 0.76 |
| **2** | 316 | 104 | 3 | 3.0 | 3.5 | 8.00 | yes | 0.72 |
| **3** | 322 | 110 | 3 | 3.5 | 2.5 | 8.67 | yes | 0.80 |
| **4** | 314 | 103 | 2 | 2.0 | 3.0 | 8.21 | no | 0.65 |

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   GRE Score          500 non-null    int64  
 1   TOEFL Score        500 non-null    int64  
 2   University Rating  500 non-null    int64  
 3   SOP                500 non-null    float64
 4   LOR                500 non-null    float64
 5   CGPA               500 non-null    float64
 6   Research           500 non-null    object 
 7   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(3), object(1)
memory usage: 31.4+ KB
```

Shape is 500 rows and 8 columns all except research experience are numerical data types. There are no missing values in any column.

## checking for null values

In [7]:
```
df.isna().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
```

## checking for duplicated values

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

Out[8]:
```
0
```

In [9]:
```
df.describe() # numerical features
```

Out[9]:

| | GRE Score | TOEFL Score | University Rating | SOP | LOR | CGPA | Chance of Admit |
|---|---|---|---|---|---|---|---|
| **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 | 0.72174 |
| **std** | 11.295148 | 6.081868 | 1.143512 | 0.991004 | 0.92545 | 0.604813 | 0.14114 |
| **min** | 290.000000 | 92.000000 | 1.000000 | 1.000000 | 1.00000 | 6.800000 | 0.34000 |
| **25%** | 308.000000 | 103.000000 | 2.000000 | 2.500000 | 3.00000 | 8.127500 | 0.63000 |
| **50%** | 317.000000 | 107.000000 | 3.000000 | 3.500000 | 3.50000 | 8.560000 | 0.72000 |
| **75%** | 325.000000 | 112.000000 | 4.000000 | 4.000000 | 4.00000 | 9.040000 | 0.82000 |
| **max** | 340.000000 | 120.000000 | 5.000000 | 5.000000 | 5.00000 | 9.920000 | 0.97000 |

Mean GRE score is 316.47. Ranges from 290 to 340. Median is close to mean at 317 so there are no outliers.

Mean TOEFL score is 107.2 and median is 107 so there are no outliers. Ranges from 92 to 120.

Mean university rating is 3.11, median 3 so there are not many outliers and ranges from 1 to 5.

Mean SOP score is 3.37, median 3.5 so there are not many outliers, and scores range from 1 to 5.

Mean LOR score is 3.48, median 3.5 so there are not many outliers, and scores range from 1 to 5.

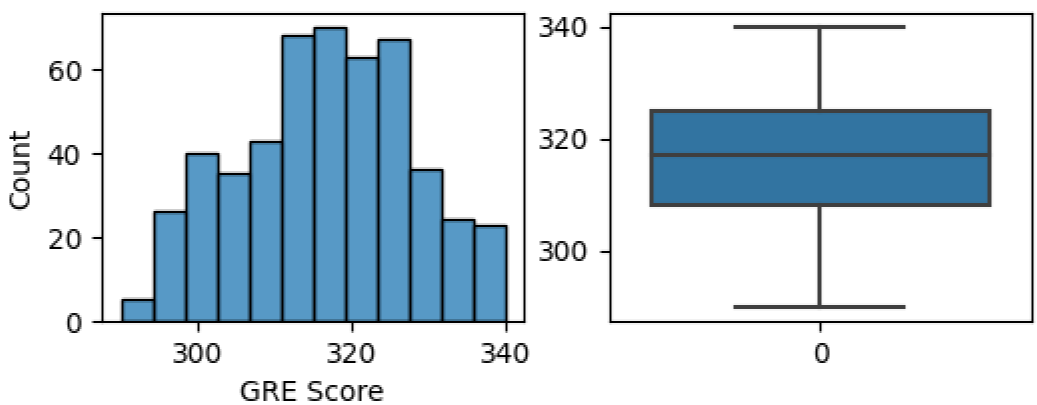
Mean CGPA is 8.58, median is 8.56 so there are not many outliers, and range is from 6.8 to 9.92.

Mean of chances of admit is 0.7217 or 72.17%, median is 72% so there are not many outliers, and range is from 34% to 97%.

## Univariate analysis

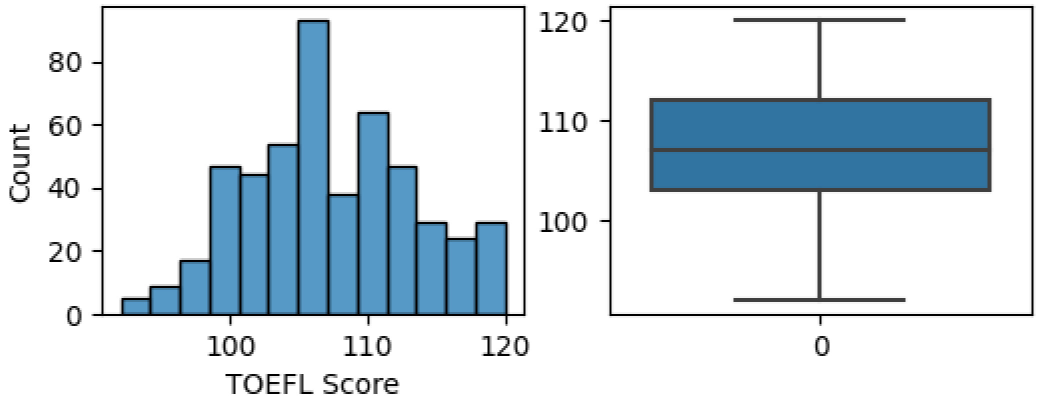
In [10]:

```
f = plt.figure()
f.set_figwidth(6)
f.set_figheight(2)
plt.subplot(121)
sns.histplot(data=df, x='GRE Score') # looks close to a bell-shaped curve.
plt.subplot(122)
sns.boxplot(df['GRE Score'])
plt.show()  # there are no outliers here.
```



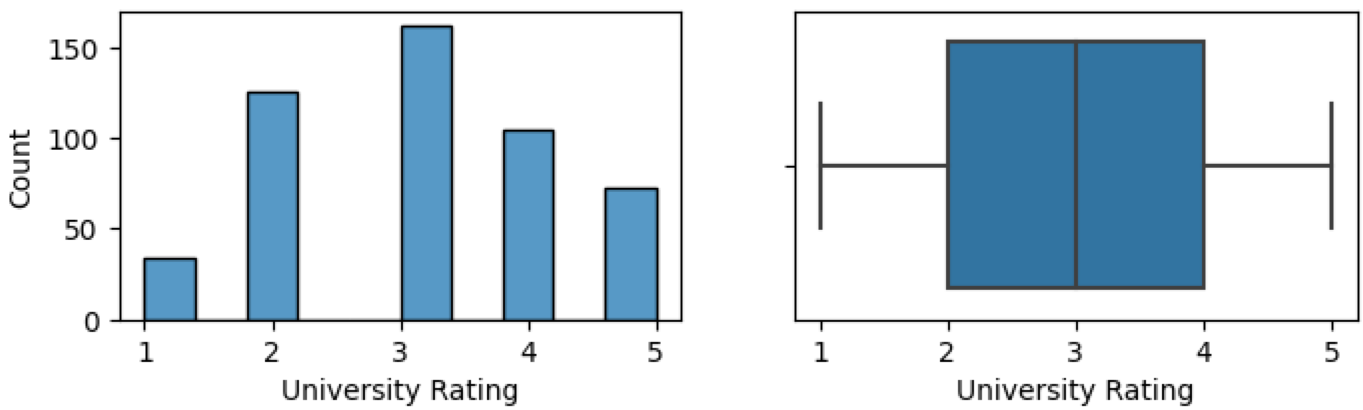

In [11]:

```
f = plt.figure()
f.set_figwidth(6)
f.set_figheight(2)
plt.subplot(121)
sns.histplot(data=df, x='TOEFL Score') # looks close to a bell-shaped curve
plt.subplot(122)
sns.boxplot(df['TOEFL Score'])
plt.show() # no outliers in the boxplot
```



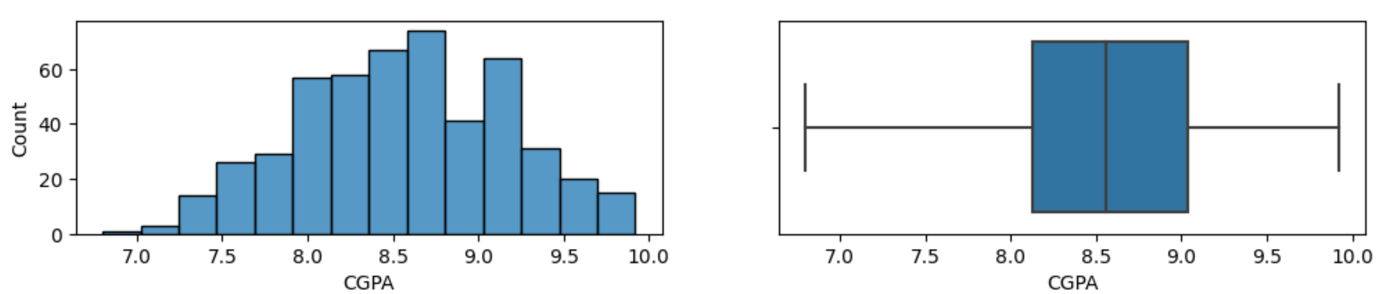

In [12]:
```
f = plt.figure()
f.set_figwidth(8)
f.set_figheight(2)
plt.subplot(121)
sns.histplot(df["University Rating"]) # looks like bell-shaped curve
plt.subplot(122)
sns.boxplot(data=df, x='University Rating') # no outliers
```

Out[12]: <AxesSubplot: xlabel='University Rating'>




In [13]:
```
f = plt.figure()
f.set_figwidth(12)
f.set_figheight(2)
plt.subplot(121)
sns.histplot(df['CGPA']) #looks like a normal distribution
plt.subplot(122)
sns.boxplot(data=df, x='CGPA') # no outliers
```

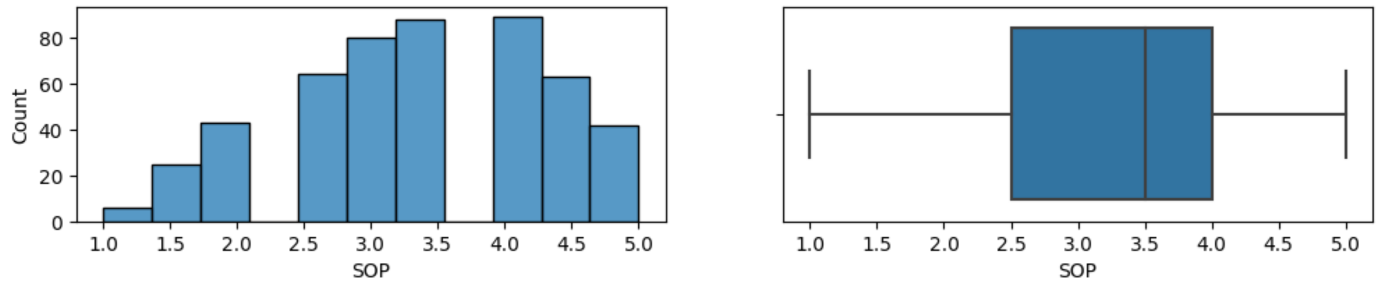
Out[13]: <AxesSubplot: xlabel='CGPA'>




In [14]:
```
f = plt.figure()
f.set_figwidth(12)
f.set_figheight(2)
plt.subplot(121)
sns.histplot(df['SOP']) #looks like a left skewed distribution
```

```
plt.subplot(122)
sns.boxplot(data=df, x='SOP') # no outliers
```

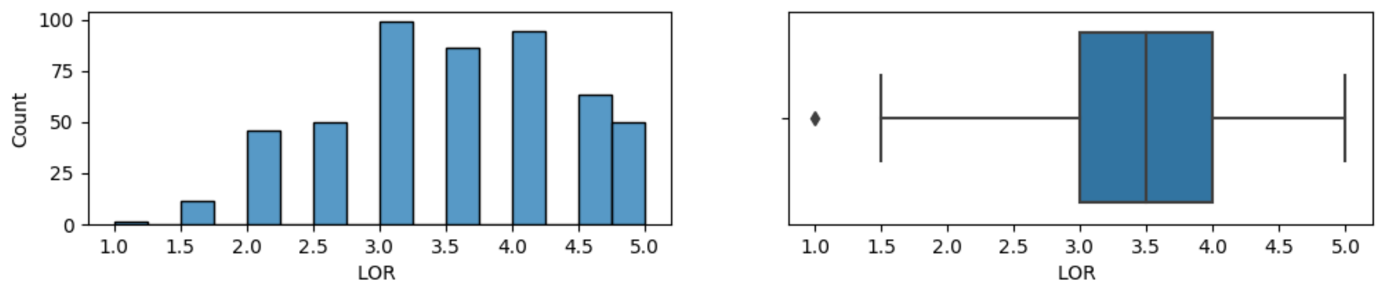
Out[14]: <AxesSubplot: xlabel='SOP'>




In [15]:

```
f = plt.figure()
f.set_figwidth(12)
f.set_figheight(2)
plt.subplot(121)
sns.histplot(df['LOR ']) #looks like a left skewed distribution
plt.subplot(122)
sns.boxplot(data=df, x='LOR ') # there are some outliers
```

Out[15]: <AxesSubplot: xlabel='LOR '>




In [16]:

```
import scipy
iqr = scipy.stats.iqr(df['LOR '])
q1 = np.percentile(df['LOR '],25)
print(q1)
df['LOR '][df['LOR '] < (q1 - iqr*1.5)]
```

```
3.0
```

Out[16]:

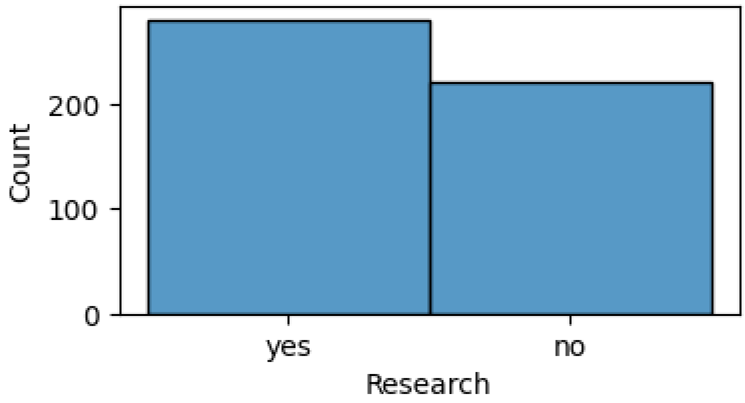
```
347    1.0
Name: LOR , dtype: float64
```

### There is only 1 outlier at 1.0 which is not an error, but a valid score so we will keep it.

In [17]:

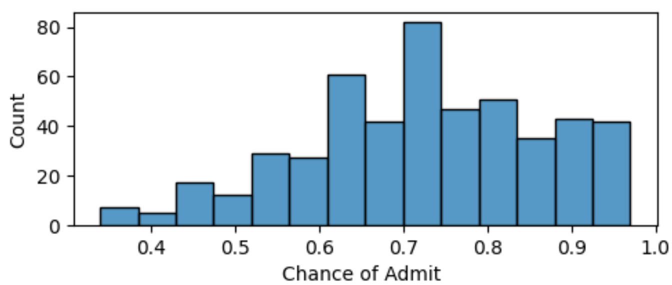
```
f = plt.figure()
f.set_figwidth(4)
f.set_figheight(2)
sns.histplot(df['Research']) #there are more students with research experience that those withou
```

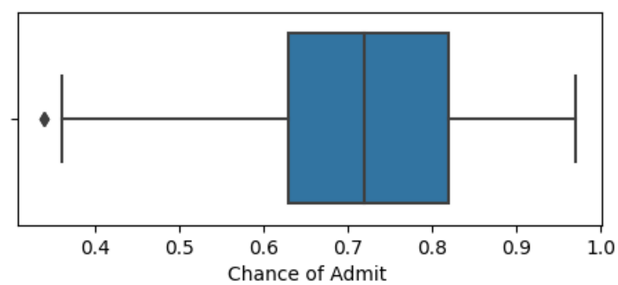
Out[17]: <AxesSubplot: xlabel='Research', ylabel='Count'>




In [18]:
```
f = plt.figure()
f.set_figwidth(12)
f.set_figheight(2)
plt.subplot(121)
sns.histplot(df['Chance of Admit ']) #looks like a left skewed distribution
plt.subplot(122)
sns.boxplot(data=df, x='Chance of Admit ') # there are some outliers
```

Out[18]: <AxesSubplot: xlabel='Chance of Admit '>




In [19]:
```
iqr = scipy.stats.iqr(df['Chance of Admit '])
q1 = np.percentile(df['Chance of Admit '],25)
print(q1)
df['Chance of Admit '][df['Chance of Admit '] < (q1 - iqr*1.5)]
```

```
0.63
```

Out[19]:
```
92     0.34
376    0.34
Name: Chance of Admit , dtype: float64
```

### Since, these outliers have a valid percentage for chance of admit, we will not remove them.

Now that we have checked the various merits with which students have applied for universities, let's look at their relationship with each other.

## Bivariate analysis

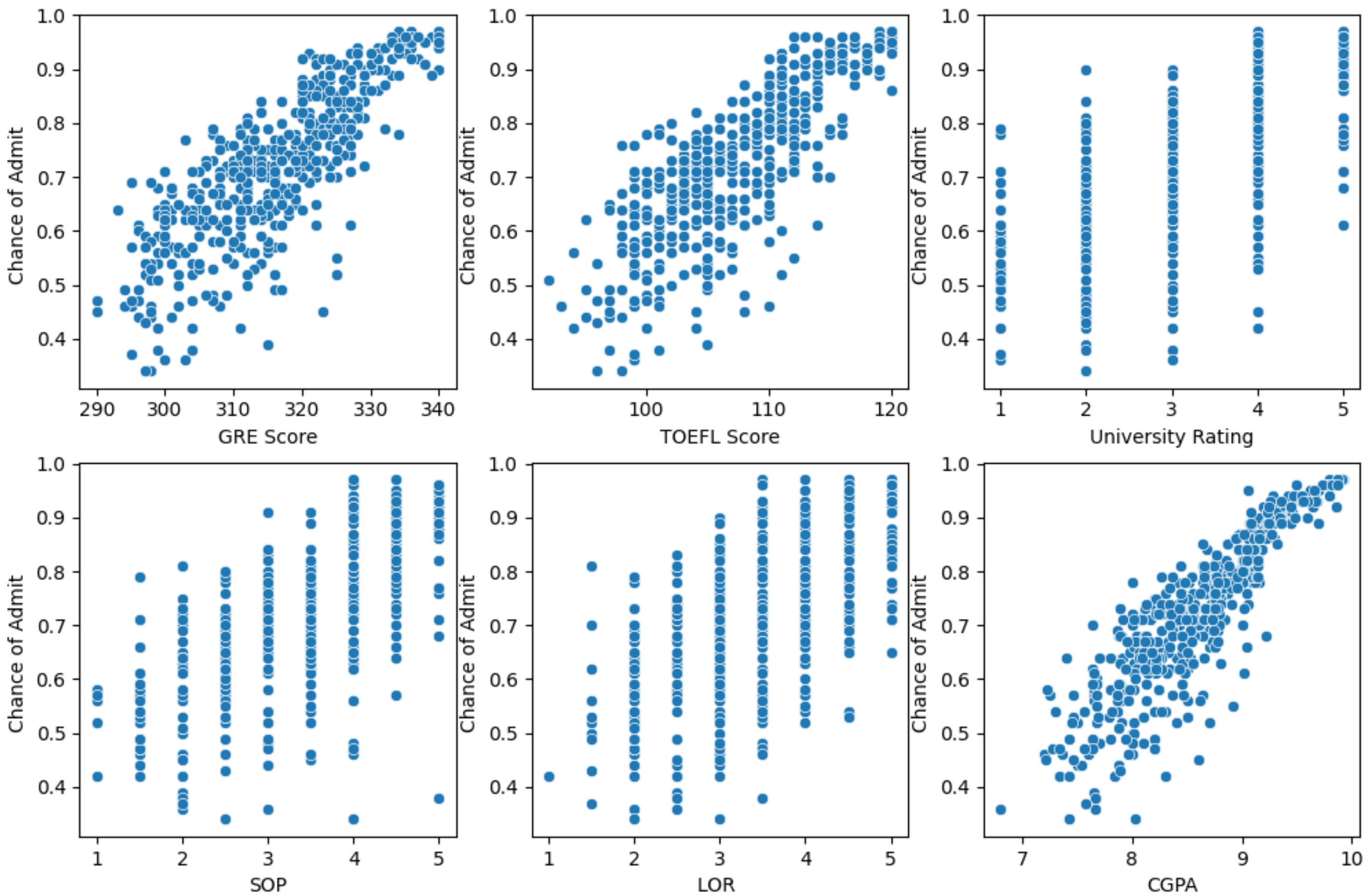
In [20]:
```
continuous_cols = df.columns[df.dtypes != 'object'][:-1]
continuous_cols
```

Out[20]:
```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA'], dtype='object')
```

In [21]:
```
f = plt.figure()
f.set_figwidth(12)
f.set_figheight(12)
```

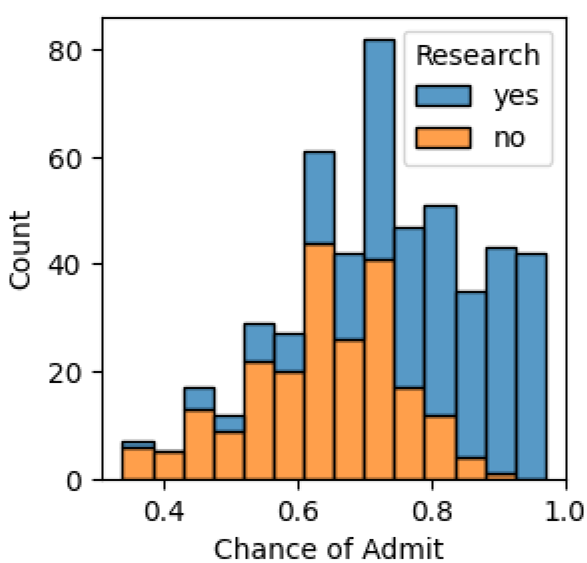
```python
n = len(continuous_cols)

for i in range(n):
    plt.subplot(n//2,n-(n//2),i+1)
    sns.scatterplot(data=df, x=continuous_cols[i],y='Chance of Admit ')
plt.show()
```




All the numerical scores are positively correlated with chance of getting an admit. Most have a linear trend of increase with increase in chance of admit.

In [22]:
```python
fig = plt.figure()
fig.set_figwidth(3)
fig.set_figheight(3)
sns.histplot(data=df,x='Chance of Admit ', hue='Research',multiple='stack')
plt.show()
```



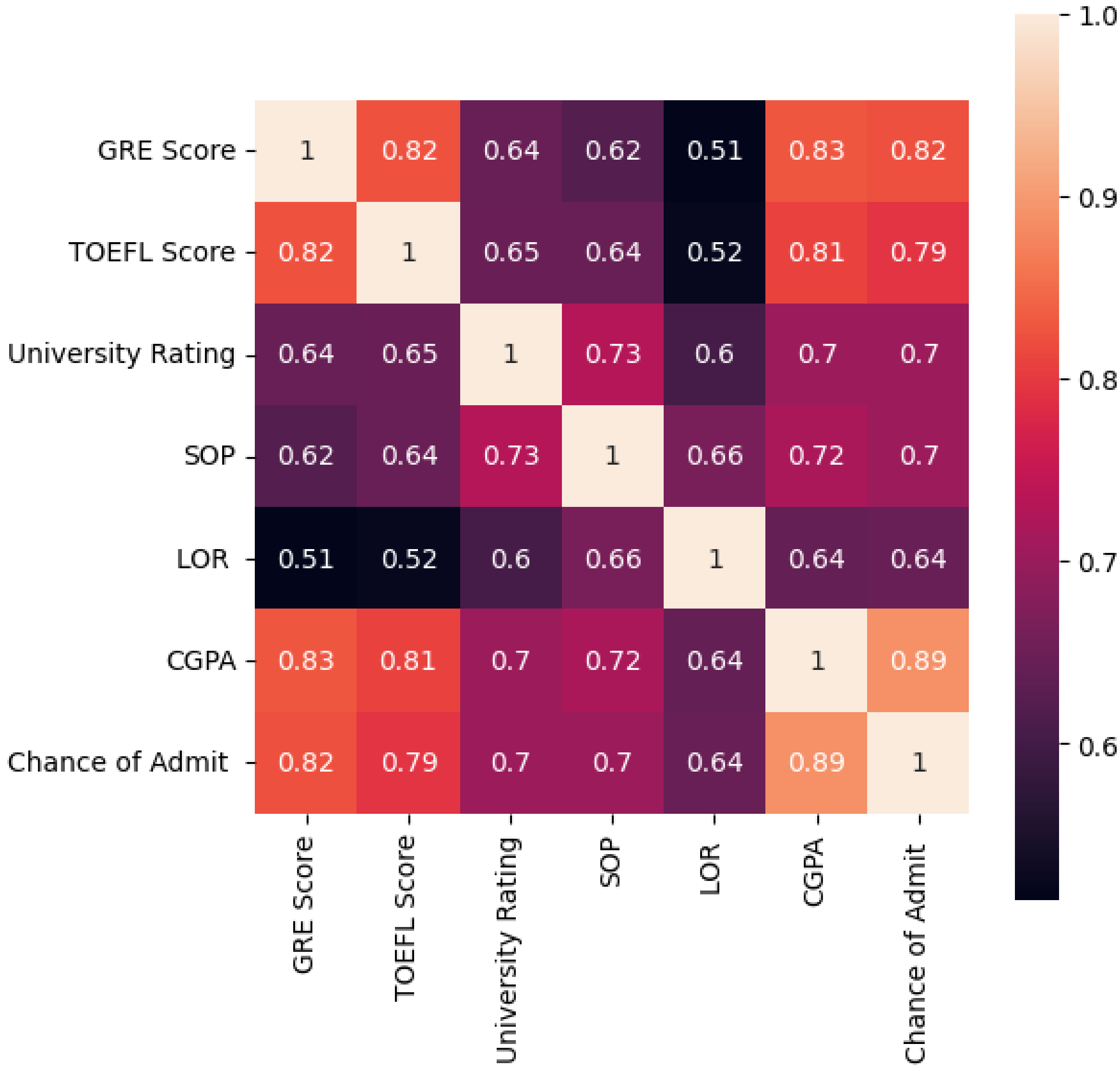
The students with research experience have higher chances of admit. But can we say that for the entire population or just for this sample of data? We would need to get statistical significance for this.

## Multivariate analysis

In [23]:
```
# Spearman's Rank Correlation Coefficient
plt.figure(figsize=(6,6))
sns.heatmap(df.corr(method='spearman'), square=True,annot=True)
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_3720\3748891021.py:3: FutureWarning: The default val
ue of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to Fals
e. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(method='spearman'), square=True,annot=True)
```

Out[23]: <AxesSubplot: >




1. The GRE score and undergrad CGPA are highest indicator of chance of admit with correlation of 0.82 and 0.89.
2. The TOEFL score, university rating, SOP scores, and LOR scores are also correlated with chance of getting an admit with correlation coefficients at 0.79, 0.7, 0.7 and 0.64 respectively.

Let's find out a way to mathematically model this correlation of admit chances with numerical scores, so that we can do future chance of admit prediction.

## Linear Regression

In [24]:
```
Y = np.array(df["Chance of Admit "]).reshape(-1,1)
X = df[continuous_cols]
```

In [25]:
```
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.model_selection import train_test_split
# Create training and test split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
```

### Column Standarization

As the different test scores are in different units, we cannot fairly compare them in terms of importance. We need to scale them to a standard range called standardization.

In [43]:
```
# Mean centering and Variance scaling (Standard Scaling)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
std_data = scaler.fit_transform(X_train)
X_train_transformed = pd.DataFrame(std_data, columns=continuous_cols)
X_test_transformed = scaler.transform(X_test)
X_test_transformed = pd.DataFrame(X_test_transformed, columns=continuous_cols)
X_train_transformed.head()
```

Out[43]:

| | GRE Score | TOEFL Score | University Rating | SOP | LOR | CGPA |
|---|---|---|---|---|---|---|
| **0** | 1.223185 | 1.279809 | 1.647869 | 1.133935 | -0.529123 | 1.285506 |
| **1** | -1.613224 | -0.868155 | -0.084125 | 0.632828 | 0.015562 | 0.073490 |
| **2** | 0.491209 | 0.453669 | 1.647869 | 1.635043 | 0.560248 | 0.881501 |
| **3** | 0.033723 | -0.042015 | -0.084125 | 0.632828 | -0.529123 | 0.208159 |
| **4** | -0.881247 | -0.372471 | 0.781872 | -0.369388 | -0.529123 | -1.071191 |

In [44]:
```
y_train.shape
```

Out[44]: (350, 1)

In [45]:
```
X_train.shape
```

Out[45]: (350, 6)

### Statsmodel OLS implementation of Linear Regression

In [32]:
```
# Statmodels implementation of Linear regression
import statsmodels.api as sm

X_sm = sm.add_constant(X_train_transformed)  #Statmodels default is without intercept, to add in

sm_model = sm.OLS(y_train, X_sm).fit() # fitting the model

print(sm_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.815
Model:                            OLS   Adj. R-squared:                  0.811
Method:                 Least Squares   F-statistic:                     251.1
Date:                Wed, 14 Jun 2023   Prob (F-statistic):          3.10e-122
Time:                        22:50:04   Log-Likelihood:                 487.74
No. Observations:                 350   AIC:                            -961.5
Df Residuals:                     343   BIC:                            -934.5
Df Model:                           6
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                 0.7241      0.003    223.296      0.000       0.718       0.730
GRE Score             0.0301      0.007      4.601      0.000       0.017       0.043
TOEFL Score           0.0200      0.006      3.309      0.001       0.008       0.032
University Rating     0.0048      0.005      0.904      0.366      -0.006       0.015
SOP                   0.0014      0.006      0.246      0.806      -0.010       0.012
LOR                   0.0145      0.005      3.165      0.002       0.005       0.023
CGPA                  0.0686      0.007      9.755      0.000       0.055       0.082
==============================================================================
Omnibus:                       81.292   Durbin-Watson:                   2.052
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              192.545
Skew:                          -1.139   Prob(JB):                     1.55e-42
Kurtosis:                       5.831   Cond. No.                         5.38
==============================================================================

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

The Ordinary Least Squares regression R-squared value is moderately good at 0.817, and adjusted R-squared at 0.815. But the p-values for University Rating and SOP are greater than 0.05 so they are not statistically significant factors for admissions. Let's build the model again without these factors.

In [34]:
```
sm_model = sm.OLS(y_train, X_sm.drop(['SOP', 'University Rating'],axis=1)).fit()
print(sm_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.814
Model:                            OLS   Adj. R-squared:                  0.812
Method:                 Least Squares   F-statistic:                     377.3
Date:                Wed, 14 Jun 2023   Prob (F-statistic):         1.48e-124
Time:                        22:52:13   Log-Likelihood:                 487.14
No. Observations:                 350   AIC:                            -964.3
Df Residuals:                     345   BIC:                            -945.0
Df Model:                           4
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
const           0.7241      0.003    223.560      0.000       0.718       0.730
GRE Score       0.0308      0.006      4.745      0.000       0.018       0.044
TOEFL Score     0.0210      0.006      3.524      0.000       0.009       0.033
LOR             0.0161      0.004      3.875      0.000       0.008       0.024
CGPA            0.0706      0.007     10.575      0.000       0.057       0.084
==============================================================================
Omnibus:                       79.245   Durbin-Watson:                   2.058
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              184.438
Skew:                          -1.118   Prob(JB):                     8.91e-41
Kurtosis:                       5.765   Cond. No.                         4.41
==============================================================================

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

The coefficients are as shown below:

In [35]:
```
sm_model.params
```

Out[35]:
```
const          0.724086
GRE Score      0.030754
TOEFL Score    0.021013
LOR            0.016119
CGPA           0.070619
dtype: float64
```

# Using Scipy package modules for simple Linear regression, Lasso and Ridge regression

In [36]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

In [46]:
```
X_test_transformed.drop(['SOP','University Rating'],axis=1,inplace=True)
```

In [47]:
```
X_train_transformed.drop(['SOP', 'University Rating'],axis=1,inplace=True)
```

In [48]:
```
# Initialize Linear Regression implementation
model = LinearRegression()
model.fit(X_train_transformed, y_train)
y_pred = model.predict(X_test_transformed)
print(model.coef_, model.intercept_)
model.score(X_test_transformed, y_test), model.score(X_train_transformed, y_train)
```

```
[[0.03075408 0.02101284 0.01611915 0.07061864]] [0.72408571]
```

Out[48]:
```
(0.8145572502266275, 0.8139284508791207)
```

In [49]:
```
# adjusted R-squared
1 - (1-model.score(X_test_transformed, y_test))*(len(y_test)-1)/(len(y_test)-X_test_transformed.
```

Out[49]: 
```
0.8094415881639138
```

In [50]:
```
mean_absolute_error(y_test,y_pred), mean_squared_error(y_test, y_pred)
```

Out[50]: 
```
(0.043621196823669536, 0.003865009329436615)
```

In [51]:
```
mean_absolute_error(y_train,model.predict(X_train_transformed)), mean_squared_error(y_train, mode
```

Out[51]: 
```
(0.04305750075820504, 0.0036191753246244207)
```

Since the error for test and training data are closeby, this is a good fit model not underfit or overfit.

The coefficients, intercept and R-squared value at 0.815 and adjusted R-squared value of 0.81 is similar to that of statsmodel library model a section above.

Mean absolute error was 0.04 and mean squared error 0.004.

## Lasso regression

It performs 'L1 regularization', i.e. it adds a factor of absolute value of sum of coefficients in the optimization objective

In [52]:
```
# Initialize Lasso Regression implementation
lasso = Lasso()
lasso.fit(X_train_transformed, y_train)
y_lasso_pred = lasso.predict(X_test_transformed)
print(lasso.coef_, lasso.intercept_)
lasso.score(X_test_transformed, y_test), lasso.score(X_train_transformed, y_train)
```

```
[0. 0. 0. 0.] [0.72408571]
```

Out[52]: 
```
(-0.002933371228675874, 0.0)
```

In [53]:
```
# adjusted R-squared
1 - (1-lasso.score(X_test_transformed, y_test))*(len(y_test)-1)/(len(y_test)-X_test_transformed.
```

Out[53]: 
```
-0.030600498710846358
```

In [54]:
```
mean_absolute_error(y_test,y_lasso_pred), mean_squared_error(y_test, y_lasso_pred)
```

Out[54]: 
```
(0.11470247619047622, 0.020903199727891154)
```

In [55]:
```
mean_absolute_error(y_train,lasso.predict(X_train_transformed)), mean_squared_error(y_train, las
```

Out[55]: 
```
(0.11360277551020409, 0.019450449795918368)
```

Here Lasso regression performed badly as it regularized all the less useful features to 0 value leaving an intercept. And the R-squared value is very low at -0.003. So it means that the Lasso Regression is not suitable for this dataset.

## Ridge regression

It performs 'L2 regularization', i.e. it adds a factor of sum of squares of coefficients in the optimization objective.

In [56]:
```
# Initialize Ridge Regression implementation
ridge = Ridge(alpha=0.1)
ridge.fit(X_train_transformed, y_train)
y_ridge_pred = ridge.predict(X_test_transformed)
print(ridge.coef_, ridge.intercept_)
ridge.score(X_test_transformed, y_test), ridge.score(X_train_transformed, y_train)
```

```
[[0.03076942 0.02103159 0.01613267 0.07056268]] [0.72408571]
```

Out[56]:
```
(0.8145507914795711, 0.8139284088152702)
```

In [57]:
```
# adjusted R-squared
1 - (1-ridge.score(X_test_transformed, y_test))*(len(y_test)-1)/(len(y_test)-X_test_transformed.
```

Out[57]:
```
0.8094349512445248
```

In [58]:
```
mean_absolute_error(y_test,y_ridge_pred), mean_squared_error(y_test, y_ridge_pred)
```

Out[58]:
```
(0.04362235884000494, 0.003865143943044642)
```

In [59]:
```
mean_absolute_error(y_train,ridge.predict(X_train_transformed)), mean_squared_error(y_train, rid
```

Out[59]:
```
(0.04305661019290923, 0.0036191761427852 34)
```

The coefficients, intercept of Ridge regression and R-squared value at 0.815 and adj R2 of 0.81 is similar to that of statsmodel library model and simple linear regression model. It is because it regularized all less useful features close to 0. Mean squared error is 0.004 and mean absolute error is 0.04 which are low and good values.

Since the error for test and training data are closeby, this is a good fit model not underfit or overfit.

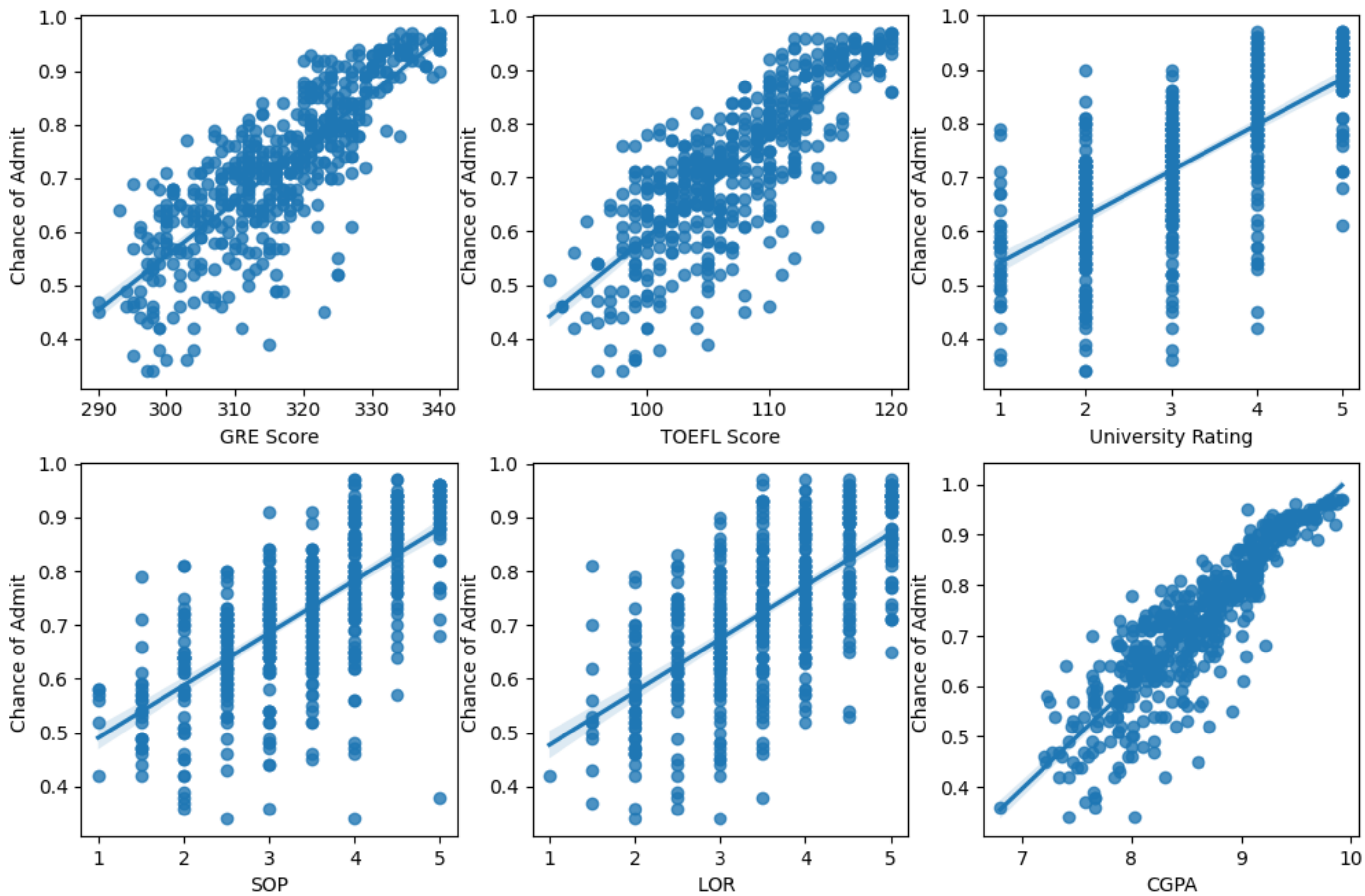
# Assumptions of linear regression:

1. There is linearity between the independent and dependent variables.
2. The residual errors are normally distributed.
3. The mean of residuals is close to 0.
4. There is homoskedasticity.
5. The independent variables are assumed to be independent of each other or have low correlation or are not multi-collinear, i.e. VIF (Variance Inflation Factor) < 10.

In [60]:
```
# 1. Linearity- all variable have a linear trend
# regplot
f = plt.figure()
f.set_figwidth(12)
f.set_figheight(12)
n = len(continuous_cols)

for i in range(n):
    plt.subplot(n//2,n-(n//2),i+1)
```

```
    sns.regplot(x=continuous_cols[i],y='Chance of Admit ', data=df)
plt.show()
```




In [64]:
```
y_pred = model.predict(X_train_transformed)
y_pred.shape, y_train.shape
```

Out[64]: ((350, 1), (350, 1))

In [73]:
```
z = y_train**2 - y_pred**2
z = z.T
z.shape
```

Out[73]: (1, 350)

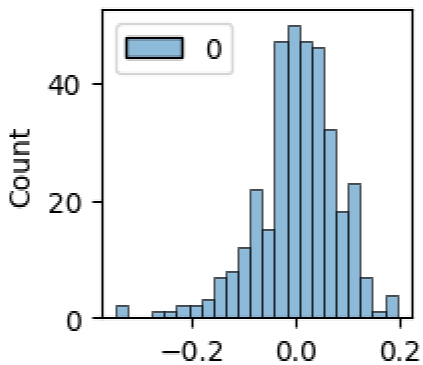
In [84]:
```
# 2. residual errors are normally distributed
# from scipy import stats
# stats.probplot(z, dist="norm", plot= plt)
# plt.title("Residuals Q-Q Plot")
# plt.legend(['Actual','Predicted'])
# plt.show()

# from bokeh.plotting import figure, show
# from scipy.stats import probplot
# # pd_series is the series you want to plot
# series1 = probplot(z, dist="norm")
# p1 = figure(title="Normal QQ-Plot", background_fill_color="#E8DDCB")
# p1.scatter(series1[0][0],series1[0][1], fill_color="red")
# show(p1)

# q = []
# j=0
# for i in range(1,z.shape[0]+1,1):
#     j=i/z.shape[0]
```

```
#     q_temp = np.quantile(z, j)
#     q.append(q_temp)
# fig, ax = plt.subplots(figsize=(10, 8))
# plt.plot(q,sorted(z),'o')
# plt.xlabel("Quantile of standard normal distribution")
# plt.ylabel("Sample Z-score")
# plt.grid()
```

In [81]:
```
y_pred = model.predict(X_train_transformed)
residuals = y_train**2 - y_pred**2
f = plt.figure()
f.set_figwidth(2)
f.set_figheight(2)
sns.histplot(residuals)
plt.show()
```




The residuals look similar to normally distributed data but have left-skew. We can remove these outliers to fulfil this assumption.

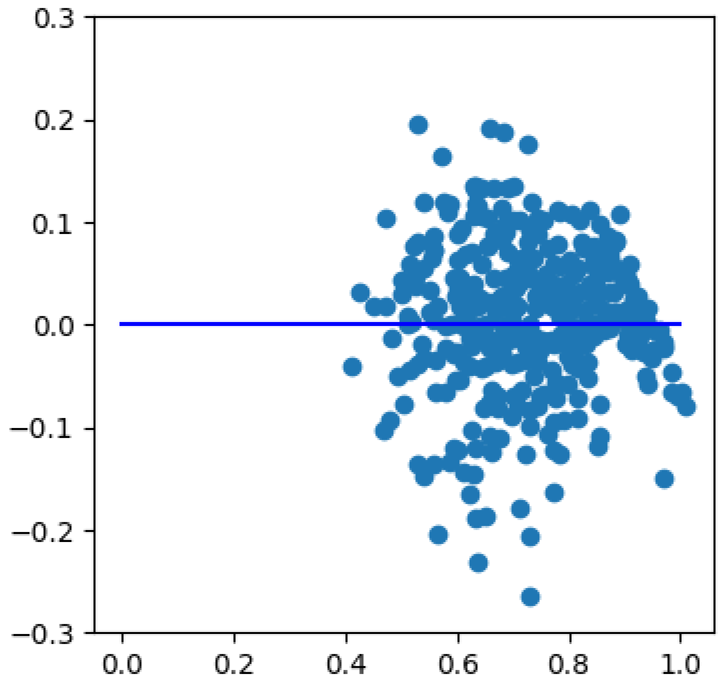
In [82]:
```
# 3. residual errors are close to 0
np.mean(y_train**2 - y_pred**2)
```

Out[82]:
```
0.0036191753246245305
```

As the mean is close to zero, we can see this assumption is met.

In [85]:
```
# 4. Homoskedasticity
f = plt.figure()
f.set_figwidth(4)
f.set_figheight(4)
plt.scatter(y_pred, residuals)
sns.lineplot([0,0],color='blue')
plt.ylim(-0.3,0.3)
```

Out[85]:
```
(-0.3, 0.3)
```




In [ ]:
```
import statsmodels.stats.api as sms
from statsmodels.compat import lzip
name = ['F statistic', 'p-value']
test = sms.het_goldfeldquandt(residuals,X_test)
lzip(name, test)
```

The residuals are having equal variance, so there is no heteroskedasticity.

In [86]:
```
# 5. VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [88]:
```
vif = pd.DataFrame()
# X_t = X
vif['Features'] = X_train_transformed.columns
vif['VIF'] = [variance_inflation_factor(X_train_transformed.values, i) for i in range(X_train_tra
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[88]:

| | Features | VIF |
|---|---|---|
| **3** | CGPA | 4.25 |
| **0** | GRE Score | 4.00 |
| **1** | TOEFL Score | 3.39 |
| **2** | LOR | 1.65 |

Any variable with a VIF of 10 or above is considered strongly correlated with other variables. There are none above 10 so there is no multi-collinearity.

## Compare chances of admit for students with and without research experience

Since we are comparing a numerical score for two categories we will be using Mann-Whitney U test which is a non-parametric alternative for two-tailed independent samples t-test.

Ho : their means are equal

Ha : the mean chances of admit for research experienced students is higher than that of students without research experience.

In [89]:
```
import scipy
from scipy.stats import norm, chi2, chi2_contingency, mannwhitneyu, f_oneway, shapiro, levene

a = df['Chance of Admit '][df['Research'] == 'yes']
b = df['Chance of Admit '][df['Research'] == 'no']

print(len(a),len(b))
print(np.array(a).var(), np.array(b).var())  # variances are unequal
print(shapiro(a)) # test for normality for a , p value is large so a is not normal
print(shapiro(b)) # pvalue is small so b is normal
print(levene(a,b)) # test for equal variances- they are are unequal
mannwhitneyu(a,b,alternative='greater')  # we continue to perform t test as the number of data po
```

```
280 220
0.015126070153061225 0.012468628099173554
ShapiroResult(statistic=0.9504043459892273, pvalue=3.8728980911173494e-08)
ShapiroResult(statistic=0.9822517037391663, pvalue=0.007273990195244551)
LeveneResult(statistic=2.291641098174848, pvalue=0.130706702224074)
```

Out[89]:
```
MannwhitneyuResult(statistic=51060.0, pvalue=6.615275731076634e-37)
```

The p-value is small (6.6e-37) so we reject the null hypothesis and say that students with research experience have significantly greater chance of admit than students without.

# Business Insights

Mean GRE score is 316.47. Ranges from 290 to 340. Median is close to mean at 317.

Mean TOEFL score is 107.2 and median is 107. Ranges from 92 to 120.

Mean university rating is 3.11, median 3 and ranges from 1 to 5.

Mean SOP score is 3.37, median 3.5, and scores range from 1 to 5.

Mean LOR score is 3.48, median 3.5, and scores range from 1 to 5.

Mean CGPA is 8.58, median is 8.56, and range is from 6.8 to 9.92.

Mean of chances of admit is 0.7217 or 72.17%, median is 72%, and range is from 34% to 97%.

1. The GRE score and undergrad CGPA are highest indicator of chance of admit with correlation of 0.82 and 0.89.
2. The TOEFL score, university rating, SOP scores, and LOR scores are also correlated with chance of getting an admit with correlation coefficients at 0.79, 0.7, 0.7 and 0.64 respectively.
3. The students with research experience have higher chances of admit.

For the categorical column on research, the Mann Whitney U p-value is small (6.6e-37) so we reject the null hypothesis and say that students with research experience have significantly greater chance of admit than students without.

For the numerical test scores, the Ordinary Least Squares regression R-squared value is moderately good at 0.817, and asjusted R-squared at 0.815. The coefficients are as shown below:

Here Lasso regression performed badly as it regularized all the less useful features to 0 value leaving an intercept. And the R-squared value is very low at -0.003. So it means that the Lasso Regression is not suitable for this dataset.

The coefficients, intercept of Ridge regression and R-squared value at 0.817 and adj R2 of 0.81 is similar to that of statsmodel library model and simple linear regression model. It is because it regularized all less useful features close to 0. Mean squared error is 0.004 and mean absolute error is 0.04 which are low and good values.

Since the error for test and training data are closeby, this is a good fit model not underfit or overfit.

The residuals have no heteroskedasticity.

Any variable with a VIF of 10 or above is considered strongly correlated with other variables. There are none above 10 so there is no multi-collinearity.

In [92]:

```
sm_model.params
```

Out[92]:

```
const          0.724086
GRE Score      0.030754
TOEFL Score    0.021013
LOR            0.016119
CGPA           0.070619
dtype: float64
```

# Recommendations

The CGPA and GRE scores have higher effect on chance of admit than other scores. So students should focus more on their undergrad CGPA and GRE test performance.

Next come the TOEFL test scores having high significance and LOR scores, so students with good TOEFL and LOR scores have high chance of admit.

University rating and SOP scores are not as important as other scores, nevertheless they have an impact on admit chances. So Jamboree can accordingly advise students from lower university ratings too.

There was no data present about the students' projects scores or the quality of their resume as that also is significant in showing universities about a student's practical knowledge and skills.

There was no data on any prior work experience if some students worked for a few years after undergrad. Work experience can also help us predict chances of admit and improve model performance.

In [ ]: