Context

- Jamboree has helped thousands of students like you make it to top colleges abroad. Be it GMAT, GRE or SAT, their unique problem-solving methods ensure maximum scores with minimum effort.
- They recently launched a feature where students/learners can come to their website and check their probability of getting into the IVY league college. This feature estimates the chances of graduate admission from an Indian perspective.

Problem Statement :

- Your analysis will help Jamboree in understanding what factors are important in graduate admissions and how these factors are interrelated among themselves. It will also help predict one's chances of admission given the rest of the variables.

Column Profiling:

```
Serial No. (Unique row ID)
GRE Scores (out of 340)
TOEFL Scores (out of 120)
University Rating (out of 5)
Statement of Purpose and Letter of Recommendation Strength (out of 5)
Undergraduate GPA (out of 10)
Research Experience (either 0 or 1)
Chance of Admit (ranging from 0 to 1)
```

- Exploratory Data Analysis
- Linear Regression

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib import figure

import warnings
warnings.filterwarnings('ignore')

import statsmodels.api as sm
```

In [ ]:
```
```

In [2]:
```
df = pd.read_csv("Jamboree_Admission.csv")
```

In [3]:
```
df
```

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
...	...	...	...	...	...	...	...	...	...
495	496	332	108	5	4.5	4.0	9.02	1	0.87
496	497	337	117	5	5.0	5.0	9.87	1	0.96
497	498	330	120	5	4.5	5.0	9.56	1	0.93
498	499	312	103	4	4.0	5.0	8.43	0	0.73
499	500	327	113	4	4.5	4.5	9.04	0	0.84

500 rows × 9 columns

In [4]:
```
data = df.copy()
```

In [ ]:

In [5]:
```
# shape of the data
data.shape
```

Out[5]:
```
(500, 9)
```

In [6]:
```
data.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 [7]:
```
data.drop(["Serial No."],axis = 1, inplace = True)
```

In [8]:
```
data.sample(5)
```

Out[8]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
109	304	103	5	5.0	4.0	8.64	0	0.68
185	327	113	4	4.5	4.5	9.11	1	0.89
264	325	110	2	3.0	2.5	8.76	1	0.75
413	317	101	3	3.0	2.0	7.94	1	0.49
33	340	114	5	4.0	4.0	9.60	1	0.90

In [ ]:

In [9]:
```
# isnull ?

data.isna().sum()
```

Out[9]:
```
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

In [10]:
```
# no null values found in data
```

In [ ]:

In [11]:
```
data.columns
```

Out[11]:
```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',
       'Research', 'Chance of Admit '],
      dtype='object')
```

In [12]:
```
data.nunique()
```

Out[12]:
```
GRE Score             49
TOEFL Score           29
University Rating      5
SOP                    9
LOR                    9
CGPA                 184
Research               2
Chance of Admit       61
dtype: int64
```

University Rating,SOP,LOR,Research are categorical variables.

all of the features are numeric , and ordinal . (University Rating,SOP,LOR,Research are discrete) and rest are continuous

In [ ]:

In [ ]:

Overall glance for correlations :

In [14]:
```
data.corr()
```

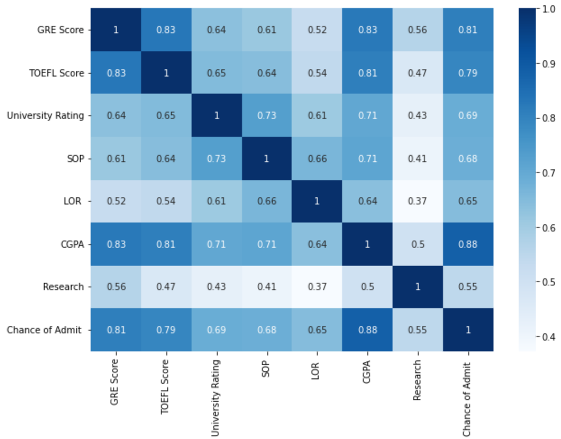
Out[14]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
GRE Score	1.000000	0.827200	0.635376	0.613498	0.524679	0.825878	0.563398	0.810351
TOEFL Score	0.827200	1.000000	0.649799	0.644410	0.541563	0.810574	0.467012	0.792228
University Rating	0.635376	0.649799	1.000000	0.728024	0.608651	0.705254	0.427047	0.690132
SOP	0.613498	0.644410	0.728024	1.000000	0.663707	0.712154	0.408116	0.684137
LOR	0.524679	0.541563	0.608651	0.663707	1.000000	0.637469	0.372526	0.645365
CGPA	0.825878	0.810574	0.705254	0.712154	0.637469	1.000000	0.501311	0.882413
Research	0.563398	0.467012	0.427047	0.408116	0.372526	0.501311	1.000000	0.545871
Chance of Admit	0.810351	0.792228	0.690132	0.684137	0.645365	0.882413	0.545871	1.000000

In [15]:
```
# further correlation check is being done while Multicoliniearity check for independent features and
# correlation between independent and dependent features.
```

In [16]:
```
plt.figure(figsize=(10,7))
sns.heatmap(data.corr(),annot = True,cmap = "Blues")
```

Out[16]: <AxesSubplot:>



In [ ]:

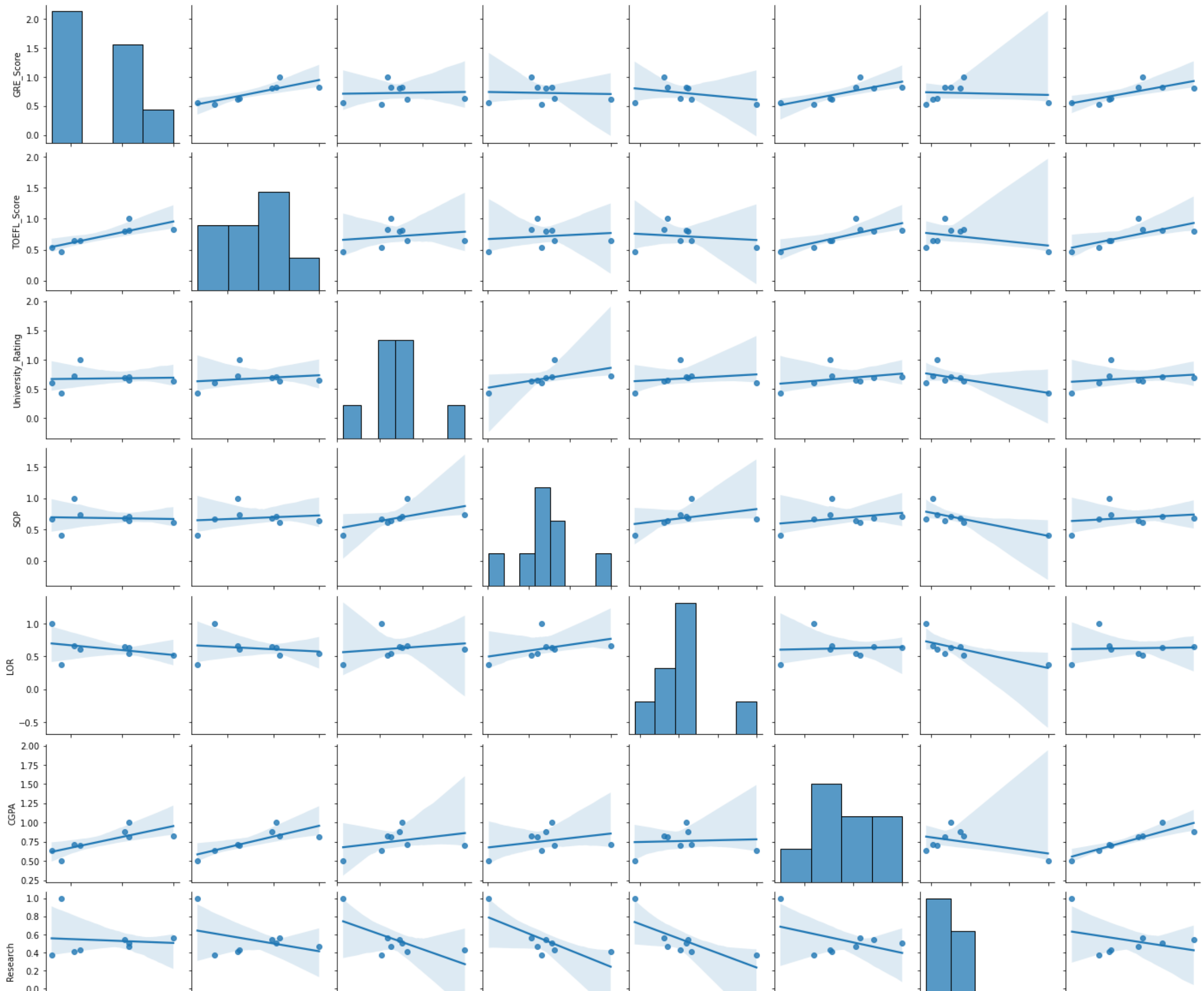
In [17]:
```
data.columns  = ['GRE_Score', 'TOEFL_Score', 'University_Rating', 'SOP', 'LOR', 'CGPA',
       'Research', 'Chance_of_Admit']
```

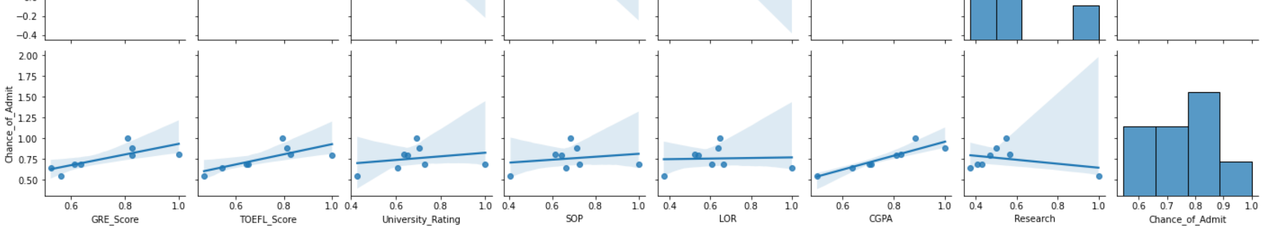
In [ ]:

pairplot , correlation and trend line with each variables:

```
In [18]: sns.pairplot(data.corr(),kind= 'reg',)
```

```
Out[18]: <seaborn.axisgrid.PairGrid at 0x2ac1e717e80>
```





In [ ]:

check for outliers using IQR method

In [19]:
```
def detect_outliers(data):
    length_before = len(data)
    Q1 = np.percentile(data,25)
    Q3 = np.percentile(data,75)
    IQR = Q3-Q1
    upperbound = Q3+1.5*IQR
    lowerbound = Q1-1.5*IQR
    if lowerbound < 0:
        lowerbound = 0

    length_after = len(data[(data>lowerbound)&(data<upperbound)])
    return f"{np.round((length_before-length_after)/length_before,4)} % Outliers data from input data found"
```

In [ ]:

In [ ]:

In [20]:
```
for col in data.columns:
    print(col," : ",detect_outliers(data[col]))
```

```
GRE_Score  :  0.0 % Outliers data from input data found
TOEFL_Score  :  0.0 % Outliers data from input data found
University_Rating  :  0.0 % Outliers data from input data found
SOP  :  0.0 % Outliers data from input data found
LOR  :  0.024 % Outliers data from input data found
CGPA  :  0.0 % Outliers data from input data found
Research  :  0.44 % Outliers data from input data found
Chance_of_Admit  :  0.004 % Outliers data from input data found
```

In [21]:
```
# there are no significant amount of outliers found in the data
```

In [ ]:

In [ ]:

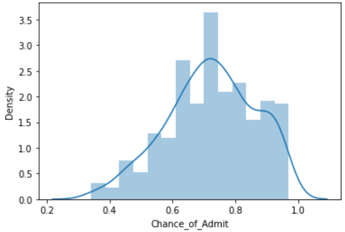
In [ ]:

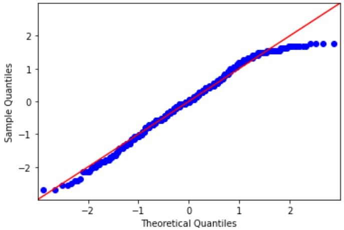
Checking the distributions for Continuous Variables :

In [22]:
```
# Chance_of_Admit
```

In [ ]:

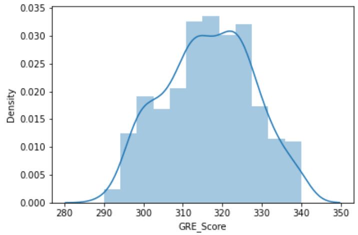
In [23]:
```
sns.distplot(data["Chance_of_Admit"])
sm.qqplot(data["Chance_of_Admit"],fit=True, line="45")
plt.show()
```

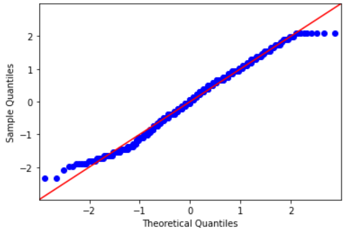




GRE_Score

In [24]:
```
sns.distplot(data["GRE_Score"])
sm.qqplot(data["GRE_Score"],fit=True, line="45")
plt.show()
```





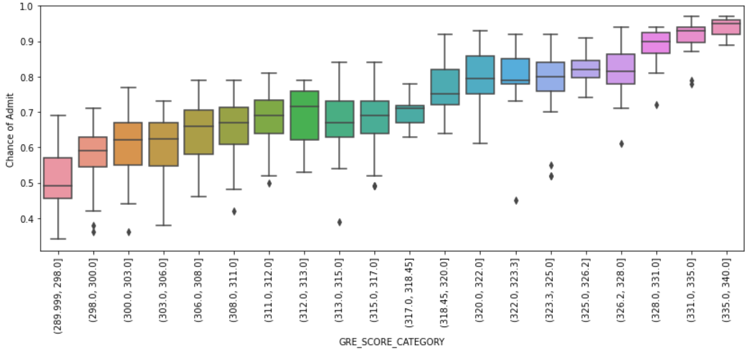
Chance of admit and GRE score are nearly normally distrubted.

for EDA purpose , converting GRE score into bins , to check how distribution of chance of admit across the bins are :

In [28]:
```
df["GRE_SCORE_CATEGORY"]=pd.qcut(df["GRE Score"],20)
```

In [ ]:

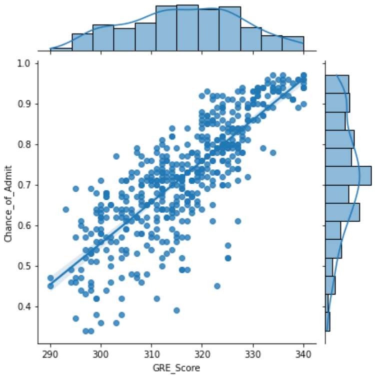
In [29]:
```
plt.figure(figsize=(14,5))
sns.boxplot(y = df["Chance of Admit "], x = df["GRE_SCORE_CATEGORY"])
plt.xticks(rotation = 90)
plt.show()
```



From above boxplot (distribution of chance of admition (probability of getting admition) as per GRE score) : with higher GRE score , there is high probability of getting an admition .

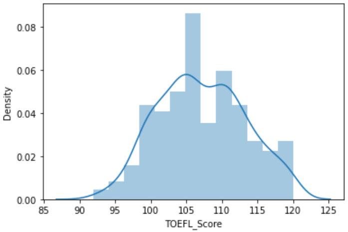
In [30]:
```
sns.jointplot(data["GRE_Score"],data["Chance_of_Admit"], kind = "reg" )
```

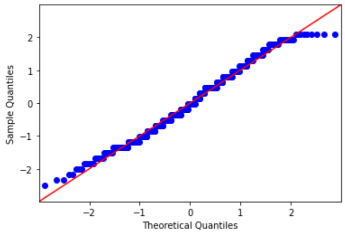
Out[30]: <seaborn.axisgrid.JointGrid at 0x2ac23b33c70>



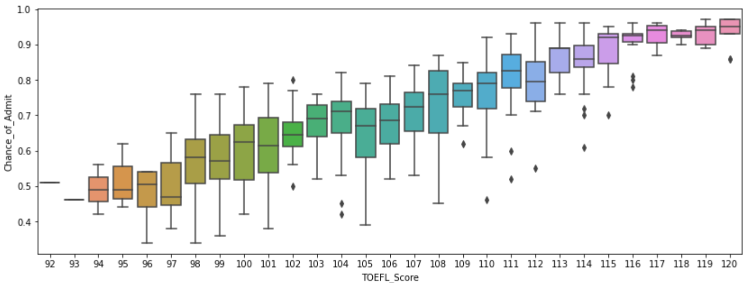
from above regression line| jointplot and boxlot we can observe a strong correlation of GRE score and chance of admit .

In [31]:
```
# TOEFL_Score

sns.distplot(data["TOEFL_Score"])
sm.qqplot(data["TOEFL_Score"],fit=True, line="45")
plt.show()
plt.figure(figsize=(14,5))
sns.boxplot(y = data["Chance_of_Admit"], x = data["TOEFL_Score"])
```





Out[31]: <AxesSubplot:xlabel='TOEFL_Score', ylabel='Chance_of_Admit'>



Students having high toefl score , has higher probability of getting admition .

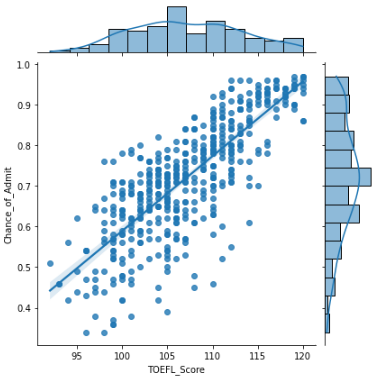
In [32]:
```
data[["GRE_Score","TOEFL_Score","Chance_of_Admit"]].corr()
```

Out[32]:

	GRE_Score	TOEFL_Score	Chance_of_Admit
GRE_Score	1.000000	0.827200	0.810351
TOEFL_Score	0.827200	1.000000	0.792228
Chance_of_Admit	0.810351	0.792228	1.000000

In [33]:
```
sns.jointplot(data["TOEFL_Score"],data["Chance_of_Admit"], kind = "reg" )
```

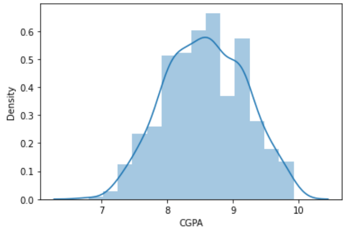
Out[33]: <seaborn.axisgrid.JointGrid at 0x2ac244bc190>

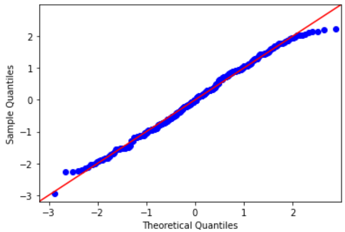


GRE_Score and Toefl_Score have very high correlation with Chance_of_Admit

In [ ]:

In [34]:
```python
# CGPA
sns.distplot(data["CGPA"])
sm.qqplot(data["CGPA"],fit=True, line="45")
plt.show()
```





In [35]:
```
data[["CGPA","Chance_of_Admit"]].corr()
```

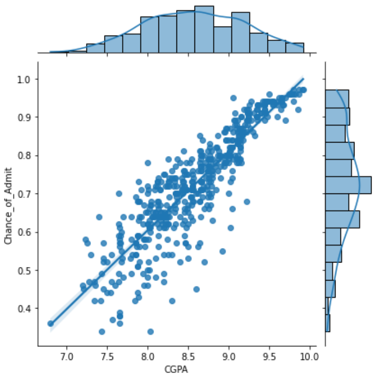
Out[35]:

	CGPA	Chance_of_Admit
CGPA	1.000000	0.882413
Chance_of_Admit	0.882413	1.000000

CGPA also has a very high correlation with Chance of Admition

In [36]:
```
sns.jointplot(data["CGPA"],data["Chance_of_Admit"], kind = "reg" )
```

Out[36]: <seaborn.axisgrid.JointGrid at 0x2ac23aa2bb0>



GRE score, TOEFL score and CGPA has a strong correlation with chance of addmission .

In [ ]:

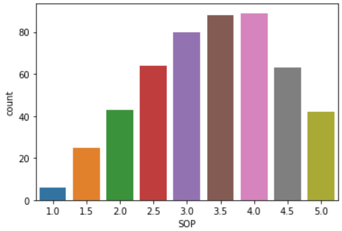
In [ ]:

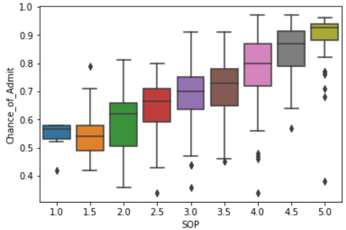
In [ ]:

In [ ]:
```
#CHECKING FOR REST OF THE FEATURES AND THEIR DISTRIBUTION :
```

In [ ]:

In [42]:
```
# SOP strength
sns.countplot(data["SOP"])
plt.show()
sns.boxplot(y = data["Chance_of_Admit"], x = data["SOP"])
plt.show()
```



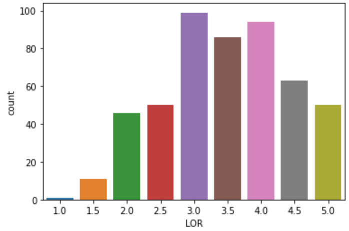


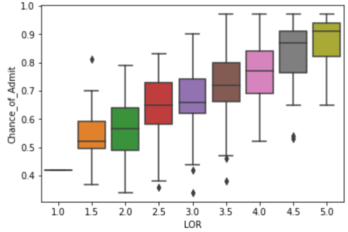
Distribution above shows , most occuring SOP strength us between 2.5 to 4.5

and having higher strength of SOP , bring more chance of getting admission !

In [ ]:

In [41]:
```
sns.countplot(data["LOR"])
plt.show()
sns.boxplot(y = data["Chance_of_Admit"], x = data["LOR"])
plt.show()
```





Statement of Purpose and Letter of Recommendation Strength increases then the chances of admition aslo increases

In [43]:
```
data[["SOP","LOR","Chance_of_Admit"]].corr()
```

Out[43]:

	SOP	LOR	Chance_of_Admit
SOP	1.000000	0.663707	0.684137
LOR	0.663707	1.000000	0.645365
Chance_of_Admit	0.684137	0.645365	1.000000

In [ ]:

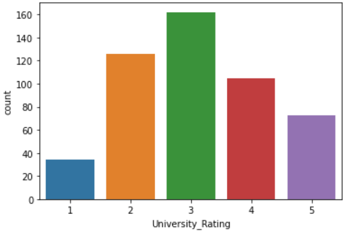
In [ ]:

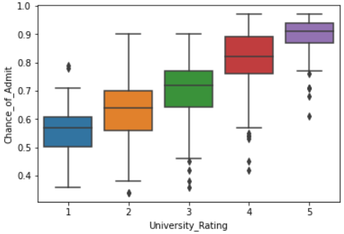
Distribution of Cateogircal variables

In [46]:
```
data["University_Rating"].value_counts()
```

Out[46]:
```
3    162
2    126
4    105
5     73
1     34
Name: University_Rating, dtype: int64
```

In [47]:
```
sns.countplot(data["University_Rating"])
plt.show()
sns.boxplot(y = data["Chance_of_Admit"], x = data["University_Rating"])
plt.show()
```

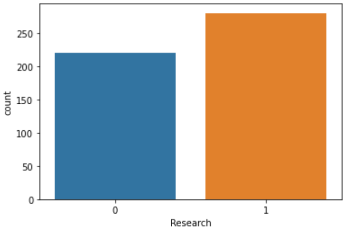


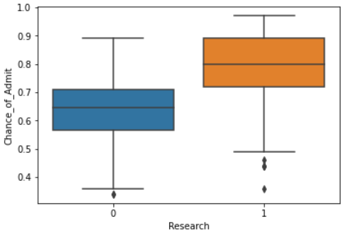


higher the university rating , increase the chance of getting admission .

In [ ]:
```
#Research
```

In [49]:
```
sns.countplot(data["Research"])
plt.show()
sns.boxplot(y = data["Chance_of_Admit"], x = data["Research"])
plt.show()
```





for research student has higher chance of getting the admission.

In [ ]:

In [ ]:

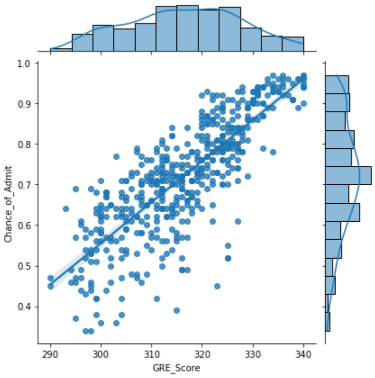
Assumption check for Linear Regression :

In [ ]:

In [50]:
```
for col in data.columns[:-1]:
    print(col)
    plt.figure(figsize=(3,3))
    sns.jointplot(data[col],data["Chance_of_Admit"],kind="reg")
    plt.show()
```

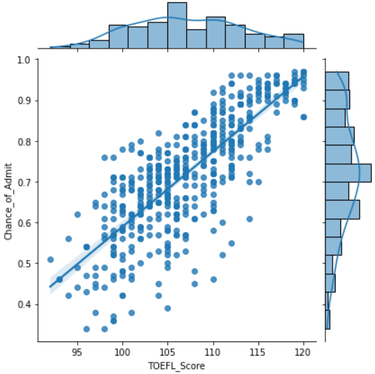
```
GRE_Score
```

```
<Figure size 216x216 with 0 Axes>
```



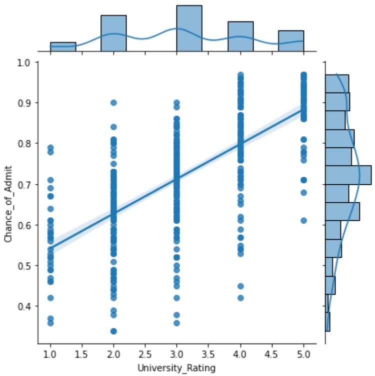

```
TOEFL_Score
```

```
<Figure size 216x216 with 0 Axes>
```



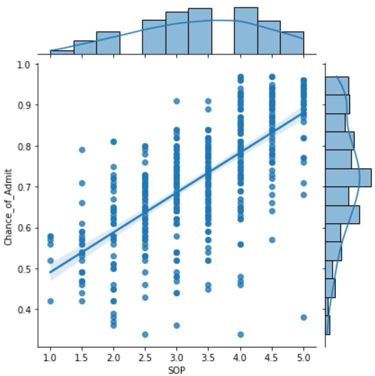
University_Rating

<Figure size 216x216 with 0 Axes>



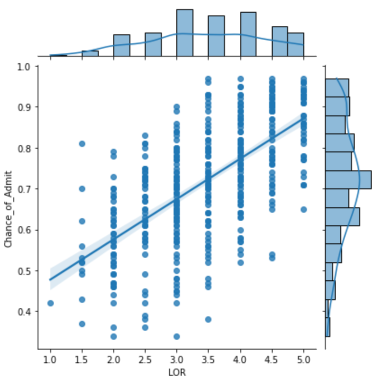
SOP

<Figure size 216x216 with 0 Axes>



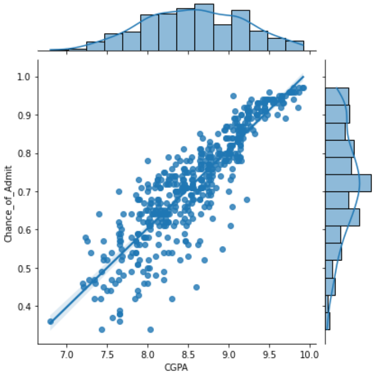
LOR

<Figure size 216x216 with 0 Axes>



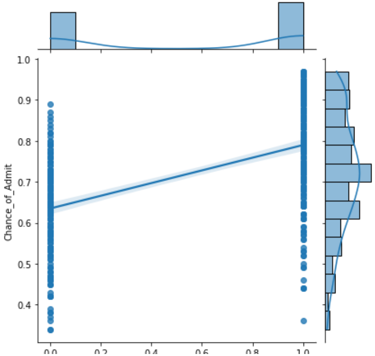
CGPA

<Figure size 216x216 with 0 Axes>



Research

<Figure size 216x216 with 0 Axes>



LOR, SOP , University rating and research are categorical variable, and amonst them chances of admits varies a lot.

In [ ]:
```
# further assumption checks are done while building and testing model .
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Regression using Sklearn library

Closed form solution technique for Linear Regression | OLS:

In [ ]:

In [ ]:

In [51]:
```
X = data.drop(["Chance_of_Admit"],axis = 1)
y = data["Chance_of_Admit"]
```

In [ ]:

In [55]:
```
from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [56]:
```
model = LinearRegression()
```

In [57]:
```
# train test spliting :
```

In [58]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [59]:
```
model.fit(X_train,y_train)
```

Out[59]:
```
LinearRegression()
```

In [61]:
```
for idx, col in enumerate(X_train.columns):
    print("Coefficient for {} is {}".format(col,model.coef_[idx]))
```

```
Coefficient for GRE_Score is 0.002134116998958902
Coefficient for TOEFL_Score is 0.0029507946431573742
Coefficient for University_Rating is 0.004842411688671617
Coefficient for SOP is 0.002095555922376041
Coefficient for LOR is 0.018600202256919177
Coefficient for CGPA is 0.11336157243184922
Coefficient for Research is 0.024713311522787978
```

In [62]:
```
intercept = model.intercept_
intercept
```

Out[62]:
```
-1.341760629850921
```

In [63]:
```
# r2_score
model.score(X_test,y_test)
```

Out[63]:
```
0.7927524897595928
```

In [64]:
```
# testing model on testing splited data.
```

In [65]:
```
y_pred = model.predict(X_test)
```

In [66]:
```
print("MSE:",mean_squared_error(y_test,y_pred)) # MSE
print("RMSE:",np.sqrt(mean_squared_error(y_test,y_pred))) #RMSE
print("MAE :",mean_absolute_error(y_test,y_pred) ) # MAE
print("r2_score:",r2_score(y_test,y_pred)) # r2score
```

```
MSE: 0.004429285498957574
RMSE: 0.06655287746564813
MAE : 0.04730057428620611
r2_score: 0.7927524897595928
```

since all the data is numeric and ordinal, keeping all the features , r_2 score is observed as 0.79 on test data .

In [ ]:

In [ ]:

Using Sklearn | Stochastic Gradient Descent Aalgorithm"

In [131]:
```
X = data.drop(["Chance_of_Admit"],axis = 1)
y = data["Chance_of_Admit"]
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [132]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [133]:
```
scaler.fit(X_train)
```

Out[133]:
```
StandardScaler()
```

In [134]:
```
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)  # apply same transformation to test data
```

In [135]:
```
from sklearn.linear_model import SGDRegressor

from sklearn.pipeline import make_pipeline
sgd = make_pipeline(StandardScaler(), SGDRegressor(max_iter=1000, tol=1e-3))
```

In [136]:
```
sgd.fit(X_train, y_train)
```

Out[136]:
```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('sgdregressor', SGDRegressor())])
```

In [137]:
```
y_pred = sgd.predict(X_test)
```

In [138]:
```
y_test = y_test.values
```

In [139]:
```
r2_score(y_test,y_pred)
```

Out[139]:
```
0.7903760694738095
```

In [ ]:
```
# overserving very similar result as OLS .
# trying different algorithms and different variations with features.
```

In [ ]:

Linear Regression using Statsmodel library

In [164]:
```
import statsmodels.api as sm
```

In [165]:
```
X = data.drop(["Chance_of_Admit"],axis = 1)
y = data["Chance_of_Admit"]
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [166]:
```
X_train_sm = X_train
X_test_sm = X_test
```

In [167]:
```
X_train_sm = sm.add_constant(X_train_sm)
X_test_sm = sm.add_constant(X_test_sm)
```

In [168]:
```
# added a constant in x_train , as stats model regression doent account for intercept separately
```

Multicolinearity check and further re-training model and testing :

In [169]:
```
data.drop(["Chance_of_Admit"],axis = 1).corr()
```

Out[169]:

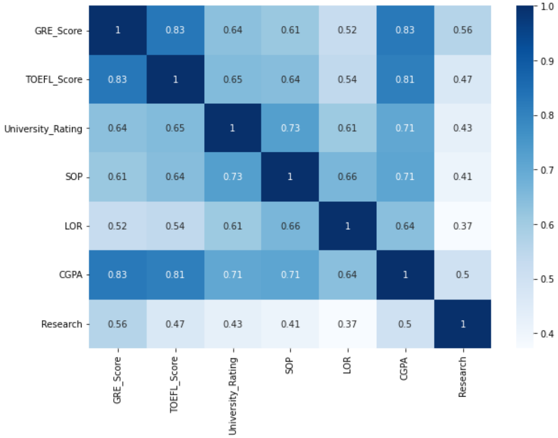
	GRE_Score	TOEFL_Score	University_Rating	SOP	LOR	CGPA	Research
GRE_Score	1.000000	0.827200	0.635376	0.613498	0.524679	0.825878	0.563398
TOEFL_Score	0.827200	1.000000	0.649799	0.644410	0.541563	0.810574	0.467012
University_Rating	0.635376	0.649799	1.000000	0.728024	0.608651	0.705254	0.427047
SOP	0.613498	0.644410	0.728024	1.000000	0.663707	0.712154	0.408116
LOR	0.524679	0.541563	0.608651	0.663707	1.000000	0.637469	0.372526
CGPA	0.825878	0.810574	0.705254	0.712154	0.637469	1.000000	0.501311
Research	0.563398	0.467012	0.427047	0.408116	0.372526	0.501311	1.000000

In [ ]:

In [ ]:

In [170]:
```
plt.figure(figsize=(10,7))
sns.heatmap(data.drop(["Chance_of_Admit"],axis = 1).corr(),annot = True,cmap = "Blues")
```

Out[170]: <AxesSubplot:>



In [171]:
```
# GRE score and Toefel score have a very high correlation with CGPA
# GRE score and TOEFL score also have a very hight correlation
# CGPA and University Rating , SOP stength and CGPA, have a high correlation .
```

In [172]:
```
# checking for Multicolinearity using vif score :
```

Variance Inflation Factor:

In [173]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [174]:
```
vifs = []

for i in range(X_train_sm.shape[1]):

    vifs.append((variance_inflation_factor(exog = X_train_sm.values,
                                           exog_idx=i)))
pd.DataFrame({ "coef_name : " : X_train_sm.columns ,
              "vif : ": np.around(vifs,2)})
```

Out[174]:

	coef_name :	vif :
0	const	1571.81
1	GRE_Score	4.24
2	TOEFL_Score	4.06
3	University_Rating	2.59
4	SOP	2.71
5	LOR	1.98
6	CGPA	4.77
7	Research	1.47

In [175]:
```
# VIF score are all below 5 , look good , there doesnt seem significant multicolinearity.
```

In [176]:
```
# model building
```

In [177]:
```
olsres = sm.OLS(y_train,X_train_sm).fit()
```

In [178]:
```
print(olsres.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        Chance_of_Admit   R-squared:                       0.829
Model:                            OLS   Adj. R-squared:                  0.826
Method:                 Least Squares   F-statistic:                     272.1
Date:                Tue, 04 Oct 2022   Prob (F-statistic):          3.33e-146
Time:                        10:20:32   Log-Likelihood:                 573.41
No. Observations:                 400   AIC:                            -1131.
Df Residuals:                     392   BIC:                            -1099.
Df Model:                           7
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                -1.3418      0.116    -11.613      0.000      -1.569      -1.115
GRE_Score             0.0021      0.001      3.893      0.000       0.001       0.003
TOEFL_Score           0.0030      0.001      3.024      0.003       0.001       0.005
University_Rating     0.0048      0.004      1.185      0.237      -0.003       0.013
SOP                   0.0021      0.005      0.428      0.669      -0.008       0.012
LOR                   0.0186      0.005      4.131      0.000       0.010       0.027
CGPA                  0.1134      0.011     10.633      0.000       0.092       0.134
Research              0.0247      0.007      3.476      0.001       0.011       0.039
==============================================================================
Omnibus:                       94.166   Durbin-Watson:                   1.943
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              231.309
Skew:                          -1.158   Prob(JB):                     5.92e-51
Kurtosis:                       5.918   Cond. No.                     1.33e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.33e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [ ]:

In [180]:
```
r2_score(y_test,olsres.predict(X_test_sm))
```

Out[180]:
```
0.7927524897595936
```

In [ ]:
```
# same result of r2 value , as sklearn OLS regressor. ,
```

In [ ]:

Residual analysis :

In [181]:
```
ypred = olsres.predict(X_train_sm)
```

In [182]:
```
print("Mean of residuals : ",np.mean(y_train - ypred))
```

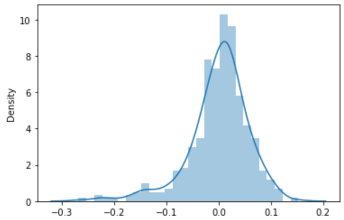
```
Mean of residuals :  1.1572687252936476e-15
```

In [183]:
```
# distribution plot of all residuals
```

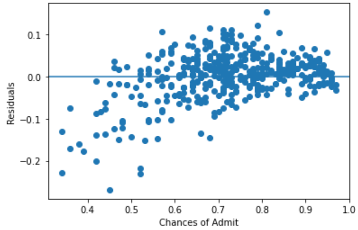
In [184]:
```
Residuals = (y_train-ypred)
```

In [185]:
```
sns.distplot(Residuals)
```

Out[185]: <AxesSubplot:ylabel='Density'>



In [186]:
```
plt.scatter(y_train,Residuals)
plt.xlabel("Chances of Admit")
plt.ylabel("Residuals")
plt.axhline(y= 0)
plt.show()
```



Homoscedasticity

from above residual plot , we can observe the varinace is not so constant .

all residuals are not evenly distributed.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

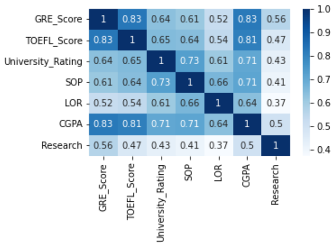
In [ ]:

In [ ]:

In [ ]:

In [187]:
```
plt.figure(figsize=(5,3))
sns.heatmap(data.drop(["Chance_of_Admit"],axis = 1).corr(),annot = True,cmap = "Blues")
```

Out[187]: <AxesSubplot:>



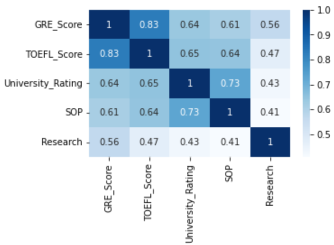

```
based on above heatmap ,
highly correlated independent features are
GRE and Toefl score
CGPA and GRE score
CGPA and TOEFL score
SOP and University_rating

we can get rid of CGPA and LOR, which can help model become better and reduce multicolinearity
```

In [192]:
```
X = data.drop(["Chance_of_Admit"],axis = 1)
y = data["Chance_of_Admit"]
```

In [193]:
```
X = X.drop(["CGPA","LOR"],axis = 1)
```

In [194]:
```
plt.figure(figsize=(5,3))
sns.heatmap(X.corr(),annot = True,cmap = "Blues")
```

Out[194]: <AxesSubplot:>



In [195]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [196]:
```
X_train_sm = X_train
X_train_sm = sm.add_constant(X_train_sm)
```

In [197]:
```
vifs = []

for i in range(X_train_sm.shape[1]):

    vifs.append((variance_inflation_factor(exog = X_train_sm.values,
                               exog_idx=i)))
```

In [198]:
```
pd.DataFrame({ "coef_name : " : X_train_sm.columns ,
             "vif : ": np.around(vifs,2)})
```

Out[198]:

	coef_name :	vif :
0	const	1551.09
1	GRE_Score	3.68
2	TOEFL_Score	3.63
3	University_Rating	2.44
4	SOP	2.35
5	Research	1.45

compare to previous model , VIF score has improved

In [199]:
```
olsres = sm.OLS(y_train,X_train_sm).fit()

print(olsres.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        Chance_of_Admit   R-squared:                       0.761
Model:                            OLS   Adj. R-squared:                  0.758
Method:                 Least Squares   F-statistic:                     251.6
Date:                Tue, 04 Oct 2022   Prob (F-statistic):          3.30e-120
Time:                        10:24:40   Log-Likelihood:                 506.48
No. Observations:                 400   AIC:                            -1001.
Df Residuals:                     394   BIC:                            -977.0
Df Model:                           5
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                -1.4911      0.135    -11.017      0.000      -1.757      -1.225
GRE_Score             0.0043      0.001      7.102      0.000       0.003       0.005
TOEFL_Score           0.0067      0.001      6.187      0.000       0.005       0.009
University_Rating     0.0168      0.005      3.597      0.000       0.008       0.026
SOP                   0.0206      0.005      3.830      0.000       0.010       0.031
Research              0.0326      0.008      3.901      0.000       0.016       0.049
==============================================================================
Omnibus:                       77.416   Durbin-Watson:                   1.864
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              138.987
Skew:                          -1.094   Prob(JB):                     6.60e-31
Kurtosis:                       4.884   Cond. No.                     1.32e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.32e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [ ]:

In [ ]:

In [200]:
```
# re-training model with sklearn library after dropping multicorrelated columns
```

In [201]:
```
model = LinearRegression()
model.fit(X_train,y_train)
```

Out[201]:
```
LinearRegression()
```

In [202]:
```
for idx, col in enumerate(X_train.columns):
    print("Coefficient for {} is {}".format(col,model.coef_[idx]))
```

```
Coefficient for GRE_Score is 0.004275146280824592
Coefficient for TOEFL_Score is 0.006727403356408807
Coefficient for University_Rating is 0.01680793854723885
Coefficient for SOP is 0.020618502555251567
Coefficient for Research is 0.03256855858438903
```

In [203]:
```
intercept = model.intercept_
intercept
```

Out[203]:
```
-1.4910580304577392
```

In [204]:
```
model.score(X_test,y_test)
```

Out[204]:
```
0.7122332491254559
```

In [205]:
```
mean_squared_error(y_test,y_pred) # MSE
```

Out[205]:
```
0.004480074258248521
```

In [206]:
```
y_pred = model.predict(X_test)
```

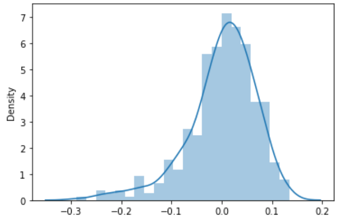
In [ ]:

In [207]:
```
r2_score(y_test,y_pred) # r2score
```

Out[207]:
```
0.7122332491254559
```

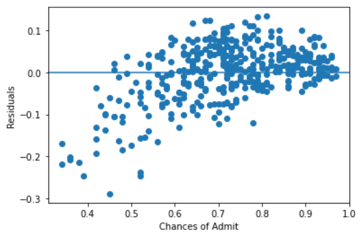
In [208]:
```
sns.distplot((y_train.values-model.predict(X_train)))
```

Out[208]: <AxesSubplot:ylabel='Density'>



In [ ]:

In [210]:
```
Residuals = (y_train - model.predict(X_train))
plt.scatter(y_train,Residuals)
plt.xlabel("Chances of Admit")
plt.ylabel("Residuals")
plt.axhline(y= 0)
plt.show()
```



removing LOR and CGPA , retrained model gives R-2 values as 71% , which decresed compared to previous model .

In [ ]:

University rating, research are categorical data

trying one hot encoding on categorical data

In [214]:
```python
X = data.drop(["Chance_of_Admit"],axis = 1)
y = data["Chance_of_Admit"]
```

In [215]:
```python
X["University_Rating"] = X["University_Rating"].astype("str")
# X["SOP"] = X["SOP"].astype("str")
# X["LOR"] = X["LOR"].astype("str")
```

In [ ]:

In [216]:
```python
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 7 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    object 
 3   SOP                500 non-null    float64
 4   LOR                500 non-null    float64
 5   CGPA               500 non-null    float64
 6   Research           500 non-null    int64  
dtypes: float64(3), int64(3), object(1)
memory usage: 27.5+ KB
```

In [217]:
```python
X = pd.get_dummies(X,columns=["University_Rating"], drop_first=True)
```

In [218]:
```python
X.sample(3)
```

Out[218]:

	GRE_Score	TOEFL_Score	SOP	LOR	CGPA	Research	University_Rating_2	University_Rating_3	University_Rating_4	University_Rating_5
181	305	107	2.5	2.5	8.42	0	1	0	0	0
36	299	106	4.0	4.0	8.40	0	1	0	0	0
484	317	106	3.5	3.0	7.89	1	0	1	0	0

In [219]:
```python
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [220]:
```python
model = LinearRegression()
model.fit(X_train,y_train)
for idx, col in enumerate(X_train.columns):
    print("Coefficient for {} is {}".format(col,model.coef_[idx]))
```

```
Coefficient for GRE_Score is 0.0020879109282400535
Coefficient for TOEFL_Score is 0.0030751293420423244
Coefficient for SOP is 0.0022557230696422905
Coefficient for LOR is 0.018809911491126007
Coefficient for CGPA is 0.1129492493754535
Coefficient for Research is 0.024624133952513998
Coefficient for University_Rating_2 is -0.00743543800264374
Coefficient for University_Rating_3 is -0.008524187583490217
Coefficient for University_Rating_4 is -0.002728961462945642
Coefficient for University_Rating_5 is 0.014023745896441092
```

In [221]:
```python
intercept = model.intercept_
intercept
```

Out[221]: -1.3199903841650387

In [222]:
```
y_predicted = model.predict(X_test)
r_2 = r2_score(y_test,y_predicted)
```

In [223]:
```
print("MSE:",mean_squared_error(y_test,y_pred)) # MSE
print("RMSE:",np.sqrt(mean_squared_error(y_test,y_pred))) #RMSE
print("MAE:",mean_absolute_error(y_test,y_pred))  # MAE
print("r2_score : ",r2_score(y_test,y_predicted)) # r2score
```

```
MSE: 0.006150139489020719
RMSE: 0.07842282505126118
MAE: 0.058159525845528276
r2_score :  0.7925241207599244
```

In [224]:
```
r_2
```

Out[224]:
```
0.7925241207599244
```

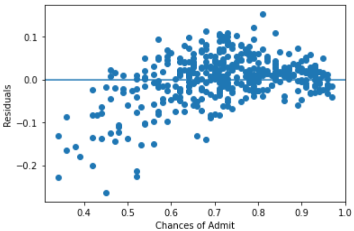
In [225]:
```
model.score(X_test,y_test)
```

Out[225]:
```
0.7925241207599244
```

In [ ]:

In [226]:
```
X_train_sm = X_train
X_train_sm = sm.add_constant(X_train_sm)
olsres = sm.OLS(y_train,X_train_sm).fit()

print(olsres.summary())
ypred = olsres.predict(X_train_sm)
Residuals = (y_train-ypred)
plt.scatter(y_train,Residuals)
plt.xlabel("Chances of Admit")
plt.ylabel("Residuals")
plt.axhline(y= 0)
plt.show()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        Chance_of_Admit   R-squared:                       0.831
Model:                            OLS   Adj. R-squared:                  0.827
Method:                 Least Squares   F-statistic:                     191.2
Date:                Tue, 04 Oct 2022   Prob (F-statistic):          2.13e-143
Time:                        10:27:08   Log-Likelihood:                 575.33
No. Observations:                 400   AIC:                            -1129.
Df Residuals:                     389   BIC:                            -1085.
Df Model:                          10
Covariance Type:            nonrobust
=======================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------
const                  -1.3200      0.118    -11.186      0.000      -1.552      -1.088
GRE_Score               0.0021      0.001      3.793      0.000       0.001       0.003
TOEFL_Score             0.0031      0.001      3.142      0.002       0.001       0.005
SOP                     0.0023      0.005      0.460      0.646      -0.007       0.012
LOR                     0.0188      0.005      4.178      0.000       0.010       0.028
CGPA                    0.1129      0.011     10.577      0.000       0.092       0.134
Research                0.0246      0.007      3.452      0.001       0.011       0.039
University_Rating_2    -0.0074      0.013     -0.554      0.580      -0.034       0.019
University_Rating_3    -0.0085      0.014     -0.599      0.550      -0.037       0.019
University_Rating_4    -0.0027      0.017     -0.163      0.871      -0.036       0.030
University_Rating_5     0.0140      0.019      0.752      0.452      -0.023       0.051
==============================================================================
Omnibus:                       89.185   Durbin-Watson:                   1.944
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              210.672
Skew:                          -1.113   Prob(JB):                     1.79e-46
Kurtosis:                       5.772   Cond. No.                     1.36e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.36e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```



In [227]:
```
vifs = []

for i in range(X_train_sm.shape[1]):

    vifs.append((variance_inflation_factor(exog = X_train_sm.values,
                               exog_idx=i)))


pd.DataFrame({ "coef_name : " : X_train_sm.columns ,
             "vif : ": np.around(vifs,2)})
```

Out[227]:

	coef_name :	vif :
0	const	1642.70
1	GRE_Score	4.29
2	TOEFL_Score	4.10
3	SOP	2.71
4	LOR	1.98
5	CGPA	4.79
6	Research	1.48
7	University_Rating_2	4.16
8	University_Rating_3	5.23
9	University_Rating_4	5.15
10	University_Rating_5	5.37

In [ ]:
```
# Converting University rating into category |and applying one hot encoding ,
# Multicolinearity seems to be increasing.
# though r__2 value is increased.
```

In [ ]:

In [ ]:

In [228]:
```
# retraining after RFE :
```

recursive feature elimination (RFE) to select features :

In [229]:
```
data
```

Out[229]:

	GRE_Score	TOEFL_Score	University_Rating	SOP	LOR	CGPA	Research	Chance_of_Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65
...	...	...	...	...	...	...	...	...
495	332	108	5	4.5	4.0	9.02	1	0.87
496	337	117	5	5.0	5.0	9.87	1	0.96
497	330	120	5	4.5	5.0	9.56	1	0.93
498	312	103	4	4.0	5.0	8.43	0	0.73
499	327	113	4	4.5	4.5	9.04	0	0.84

500 rows × 8 columns

In [241]:
```
from sklearn.feature_selection import RFE
```

In [242]:
```
LRm = LinearRegression()
```

In [243]:
```
rfe = RFE(LRm,n_features_to_select=5)
```

In [244]:
```
rfe
```

Out[244]:
```
RFE(estimator=LinearRegression(), n_features_to_select=5)
```

In [245]:
```
X = data.drop(["Chance_of_Admit"],axis = 1)
y = data["Chance_of_Admit"]
```

In [246]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [247]:
```
rfe = rfe.fit(X_train,y_train)
```

In [248]:
```
rfe.support_
```

Out[248]:
```
array([False,  True,  True, False,  True,  True,  True])
```

In [249]:
```
data.columns
```

Out[249]:
```
Index(['GRE_Score', 'TOEFL_Score', 'University_Rating', 'SOP', 'LOR', 'CGPA',
       'Research', 'Chance_of_Admit'],
      dtype='object')
```

In [250]:
```
rfe.ranking_
```

Out[250]:
```
array([2, 1, 1, 3, 1, 1, 1])
```

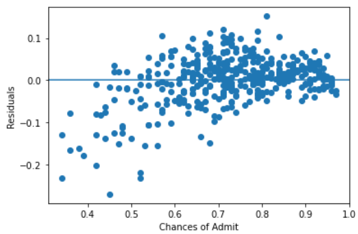
In [251]:
```
X_train.columns
```

Out[251]:
```
Index(['GRE_Score', 'TOEFL_Score', 'University_Rating', 'SOP', 'LOR', 'CGPA',
       'Research'],
      dtype='object')
```

In [252]:
```
X = data.drop(["Chance_of_Admit","University_Rating"],axis = 1)
y = data["Chance_of_Admit"]
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)

X_train_sm = X_train
X_test_sm = X_test

X_train_sm = sm.add_constant(X_train_sm)
X_test_sm = sm.add_constant(X_test_sm)
```

In [253]:
```
X_train_sm = X_train
X_train_sm = sm.add_constant(X_train_sm)
olsres = sm.OLS(y_train,X_train_sm).fit()

print(olsres.summary())
ypred = olsres.predict(X_train_sm)
Residuals = (y_train-ypred)
plt.scatter(y_train,Residuals)
plt.xlabel("Chances of Admit")
plt.ylabel("Residuals")
plt.axhline(y= 0)
plt.show()
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:        Chance_of_Admit   R-squared:                       0.829
Model:                            OLS   Adj. R-squared:                  0.826
Method:                 Least Squares   F-statistic:                     316.9
Date:                Tue, 04 Oct 2022   Prob (F-statistic):          3.61e-147
Time:                        10:31:08   Log-Likelihood:                 572.70
No. Observations:                 400   AIC:                            -1131.
Df Residuals:                     393   BIC:                            -1103.
Df Model:                           6                                         
Covariance Type:            nonrobust                                         
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
const          -1.3759      0.112    -12.291      0.000      -1.596      -1.156
GRE_Score       0.0022      0.001      3.985      0.000       0.001       0.003
TOEFL_Score     0.0030      0.001      3.099      0.002       0.001       0.005
SOP             0.0042      0.005      0.906      0.366      -0.005       0.013
LOR             0.0194      0.004      4.353      0.000       0.011       0.028
CGPA            0.1154      0.011     10.960      0.000       0.095       0.136
Research        0.0252      0.007      3.544      0.000       0.011       0.039
==============================================================================
Omnibus:                       94.122   Durbin-Watson:                   1.948
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              231.319
Skew:                          -1.157   Prob(JB):                     5.88e-51
Kurtosis:                       5.920   Cond. No.                     1.28e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.28e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```



In [254]:
```
r2_score(y_test,olsres.predict(X_test_sm))
```

Out[254]: 0.7909249818462333

In [ ]:

In [ ]:

Inferences and Recommendations :

In [ ]:

Basic EDA and structure of data :

- Fist column was observed as unique row identier which was dropped and wasnt required for neither EDA or modeling in our case.
- University Rating,SOP,LOR,Research are categorical variables. (still ordinal , have used as it is for model training.)
- all of the features are numeric , and ordinal . (University Rating,SOP,LOR,Research are discrete) and rest are continuous

- further correlation check is being done while Multicoliniearity check for independent features and correlation between independent and dependent features.
- There were no significant amount of outliers found in the data.

Feature Importance and correlations :

- Chance of admit and GRE score are nearly normally distrubted.
- for EDA purpose , converting GRE score into bins , to check how distribution of chance of admit across the bins are.
- From boxplots (distribution of chance of admition (probability of getting admition) as per GRE score) : with higher GRE score , there is high probability of getting an admition

- From regression line| jointplot and boxlot we can observe a strong correlation of GRE score and chance of admit

- GRE score, TOEFL score and CGPA has a strong correlation with chance of addmission .
- Distribution shows , most occuring SOP strength us between 2.5 to 4.5 and having higher strength of SOP , bring more chance of getting admission !
- Statement of Purpose and Letter of Recommendation Strength increases then the chances of admition aslo increases.
- higher the university rating , increase the chance of getting admission
- for research student has higher chance of getting the admission.

- sklearn OLS: since all the data is numeric and ordinal, keeping all the features , r_2 score is observed as 0.79 on test data .
- overserving very similar result as OLS .
- VIF score are all below 5 , look good , there doesnt seem significant multicolinearity.
- same result of r2 value , as sklearn OLS regressor as statsmodel regressoion model.
- Homoscedasticity : from residual plot , we can observe the varinace is not so constant .
- all residuals are not evenly distributed.
- tried removing LOR and CGPA , retrained model gives R-2 values as 71% , which decresed compared to previous model .
- Converting University rating into category |and applying one hot encoding , Multicolinearity seems to be increasing. though r__2 value is increased.
- It is recommended to use all the given data as it is which is numerical also ordinal , giving higher value of r^2. which says the model is performing better in that scenario.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: