1. DATA PREPROCESSING

a. IMPORT THE LIBRARIES

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
import scipy
import xgboost
import seaborn as sns
%matplotlib inline
```

b. IMPORT THE DATASET

In [2]:

```


d = pd.read_csv(r"C:\Users\BS663TU\Downloads\Admission_Predict.csv")
```

In [3]:

```
d.head(3)
```

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

In [4]:

```
d.describe()
```

Out[4]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	F
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	40
mean	200.500000	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	

c. TAKING CARE OF MISSING DATA

In [5]:

```
d.isnull().any()
```

Out[5]:

```
Serial No.           False
GRE Score            False
TOEFL Score          False
University Rating    False
SOP                  False
LOR                  False
CGPA                 False
Research             False
Chance of Admit      False
dtype: bool
```

d. NO LABEL ENCODING

e. NO ONEHOT ENCODING

f. DATA VISUSLIZATION

In [6]:

```
x = d.iloc[:,1:8].values
y = d.iloc[:,8].values
```

In [7]:

```
x
```

Out[7]:

```
array([[337.  , 118.  ,   4.  , ...,   4.5 ,   9.65,   1.  ],
       [324.  , 107.  ,   4.  , ...,   4.5 ,   8.87,   1.  ],
       [316.  , 104.  ,   3.  , ...,   3.5 ,   8.  ,   1.  ],
       ...,
       [330.  , 116.  ,   4.  , ...,   4.5 ,   9.45,   1.  ],
       [312.  , 103.  ,   3.  , ...,   4.  ,   8.78,   0.  ],
       [333.  , 117.  ,   4.  , ...,   4.  ,   9.66,   1.  ]])
```

In [8]:

```
y
```

Out[8]:

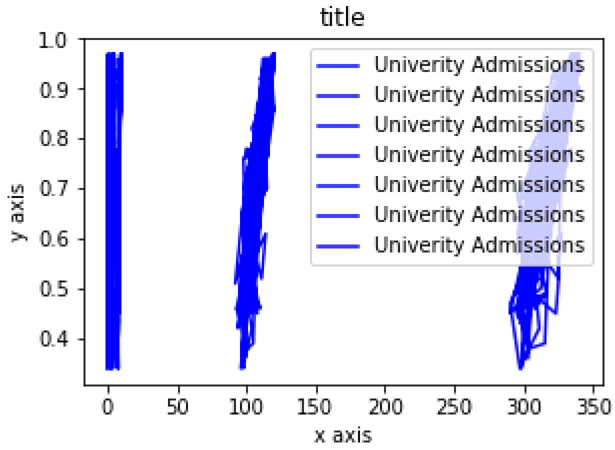
```
array([0.92, 0.76, 0.72, 0.8 , 0.65, 0.9 , 0.75, 0.68, 0.5 , 0.45, 0.52,
       0.84, 0.78, 0.62, 0.61, 0.54, 0.66, 0.65, 0.63, 0.62, 0.64, 0.7 ,
       0.94, 0.95, 0.97, 0.94, 0.76, 0.44, 0.46, 0.54, 0.65, 0.74, 0.91,
       0.9 , 0.94, 0.88, 0.64, 0.58, 0.52, 0.48, 0.46, 0.49, 0.53, 0.87,
       0.91, 0.88, 0.86, 0.89, 0.82, 0.78, 0.76, 0.56, 0.78, 0.72, 0.7 ,
       0.64, 0.64, 0.46, 0.36, 0.42, 0.48, 0.47, 0.54, 0.56, 0.52, 0.55,
       0.61, 0.57, 0.68, 0.78, 0.94, 0.96, 0.93, 0.84, 0.74, 0.72, 0.74,
       0.64, 0.44, 0.46, 0.5 , 0.96, 0.92, 0.92, 0.94, 0.76, 0.72, 0.66,
       0.64, 0.74, 0.64, 0.38, 0.34, 0.44, 0.36, 0.42, 0.48, 0.86, 0.9 ,
       0.79, 0.71, 0.64, 0.62, 0.57, 0.74, 0.69, 0.87, 0.91, 0.93, 0.68,
       0.61, 0.69, 0.62, 0.72, 0.59, 0.66, 0.56, 0.45, 0.47, 0.71, 0.94,
       0.94, 0.57, 0.61, 0.57, 0.64, 0.85, 0.78, 0.84, 0.92, 0.96, 0.77,
       0.71, 0.79, 0.89, 0.82, 0.76, 0.71, 0.8 , 0.78, 0.84, 0.9 , 0.92,
       0.97, 0.8 , 0.81, 0.75, 0.83, 0.96, 0.79, 0.93, 0.94, 0.86, 0.79,
       0.8 , 0.77, 0.7 , 0.65, 0.61, 0.52, 0.57, 0.53, 0.67, 0.68, 0.81,
       0.78, 0.65, 0.64, 0.64, 0.65, 0.68, 0.89, 0.86, 0.89, 0.87, 0.85,
       0.9 , 0.82, 0.72, 0.73, 0.71, 0.71, 0.68, 0.75, 0.72, 0.89, 0.84,
       0.93, 0.93, 0.88, 0.9 , 0.87, 0.86, 0.94, 0.77, 0.78, 0.73, 0.73,
       0.7 , 0.72, 0.73, 0.72, 0.97, 0.97, 0.69, 0.57, 0.63, 0.66, 0.64,
       0.68, 0.79, 0.82, 0.95, 0.96, 0.94, 0.93, 0.91, 0.85, 0.84, 0.74,
       0.76, 0.75, 0.76, 0.71, 0.67, 0.61, 0.63, 0.64, 0.71, 0.82, 0.73,
       0.74, 0.69, 0.64, 0.91, 0.88, 0.85, 0.86, 0.7 , 0.59, 0.6 , 0.65,
       0.7 , 0.76, 0.63, 0.81, 0.72, 0.71, 0.8 , 0.77, 0.74, 0.7 , 0.71,
       0.93, 0.85, 0.79, 0.76, 0.78, 0.77, 0.9 , 0.87, 0.71, 0.7 , 0.7 ,
       0.75, 0.71, 0.72, 0.73, 0.83, 0.77, 0.72, 0.54, 0.49, 0.52, 0.58,
       0.78, 0.89, 0.7 , 0.66, 0.67, 0.68, 0.8 , 0.81, 0.8 , 0.94, 0.93,
       0.92, 0.89, 0.82, 0.79, 0.58, 0.56, 0.56, 0.64, 0.61, 0.68, 0.76,
       0.86, 0.9 , 0.71, 0.62, 0.66, 0.65, 0.73, 0.62, 0.74, 0.79, 0.8 ,
       0.69, 0.7 , 0.76, 0.84, 0.78, 0.67, 0.66, 0.65, 0.54, 0.58, 0.79,
       0.8 , 0.75, 0.73, 0.72, 0.62, 0.67, 0.81, 0.63, 0.69, 0.8 , 0.43,
       0.8 , 0.73, 0.75, 0.71, 0.73, 0.83, 0.72, 0.94, 0.81, 0.81, 0.75,
       0.79, 0.58, 0.59, 0.47, 0.49, 0.47, 0.42, 0.57, 0.62, 0.74, 0.73,
       0.64, 0.63, 0.59, 0.73, 0.79, 0.68, 0.7 , 0.81, 0.85, 0.93, 0.91,
       0.69, 0.77, 0.86, 0.74, 0.57, 0.51, 0.67, 0.72, 0.89, 0.95, 0.79,
       0.39, 0.38, 0.34, 0.47, 0.56, 0.71, 0.78, 0.73, 0.82, 0.62, 0.96,
       0.96, 0.46, 0.53, 0.49, 0.76, 0.64, 0.71, 0.84, 0.77, 0.89, 0.82,
       0.84, 0.91, 0.67, 0.95])
```

In [9]:

```
fig=plt.figure()
axes=fig.add_axes([0.4,0.4,0.6,0.6])
axes.plot(x,y,label="Univerity Admissions",color='blue')
axes.set_xlabel("x axis")
axes.set_ylabel("y axis")
axes.set_title("title")
axes.legend(loc='upper right')
```

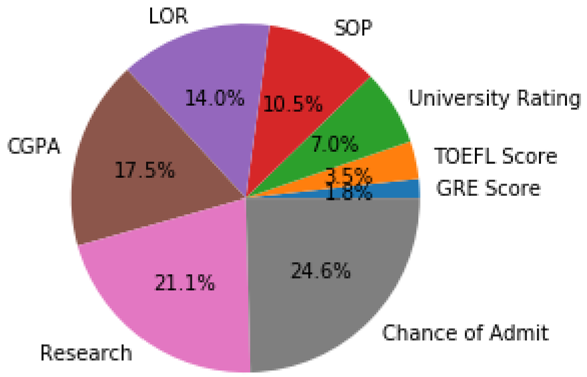
Out[9]:

```
<matplotlib.legend.Legend at 0x12e692d9ec8>
```



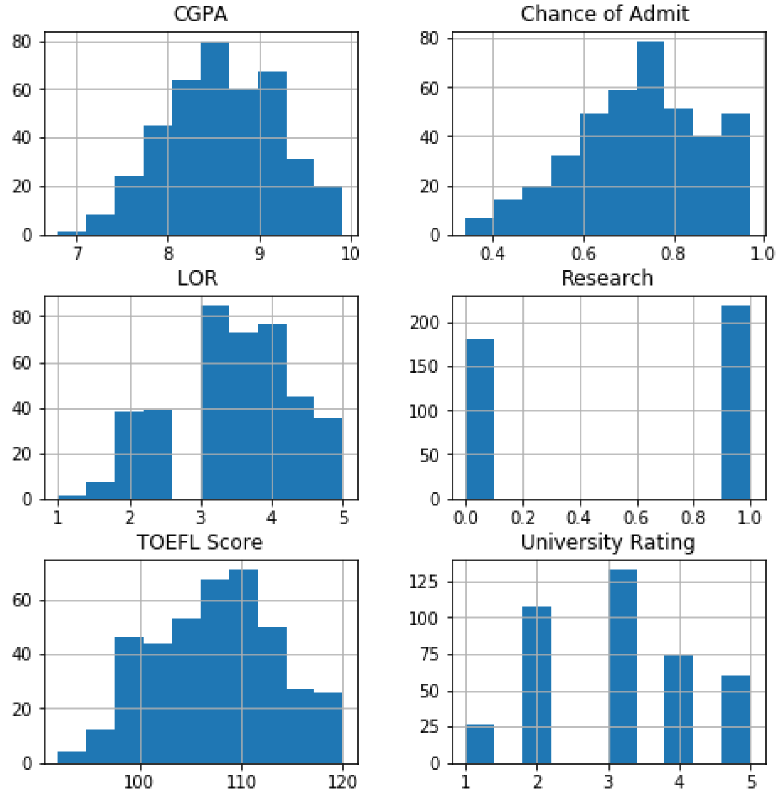
In [10]:

```
details=["GRE Score","TOEFL Score","University Rating","SOP","LOR","CGPA","Research","
student=[25,50,100,150,200,250,300,350]
plt.pie(student,labels=details,autopct="%.1f%%")
plt.show()
```



In [11]:

```
d.pop('Serial No.')
d.hist(figsize=(12,8))
plt.show()
```



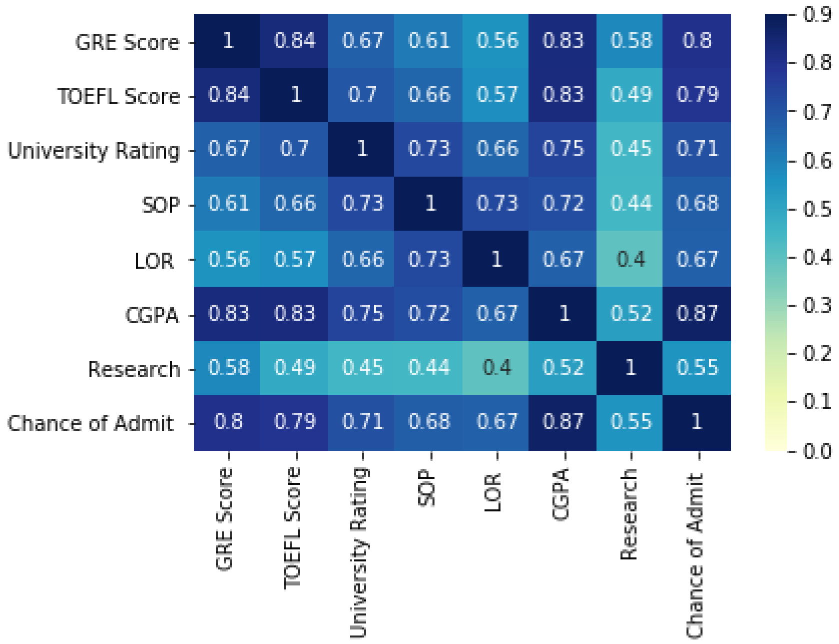
In [12]:

```
corr=d.corr()
sns.heatmap(corr, vmax=0.9,vmin=0,annot=True,cmap="YlGnBu")
corr

```

Out[12]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
GRE Score	1.000000	0.835977	0.668976	0.612831	0.557555	0.833060	0.580391	0.802610
TOEFL Score	0.835977	1.000000	0.695590	0.657981	0.567721	0.828417	0.489858	0.791594
University Rating	0.668976	0.695590	1.000000	0.734523	0.660123	0.746479	0.447783	0.711250
SOP	0.612831	0.657981	0.734523	1.000000	0.729593	0.718144	0.444029	0.675732
LOR	0.557555	0.567721	0.660123	0.729593	1.000000	0.670211	0.396859	0.669889
CGPA	0.833060	0.828417	0.746479	0.718144	0.670211	1.000000	0.521654	0.873289
Research	0.580391	0.489858	0.447783	0.444029	0.396859	0.521654	1.000000	0.553202
Chance of Admit	0.802610	0.791594	0.711250	0.675732	0.669889	0.873289	0.553202	1.000000

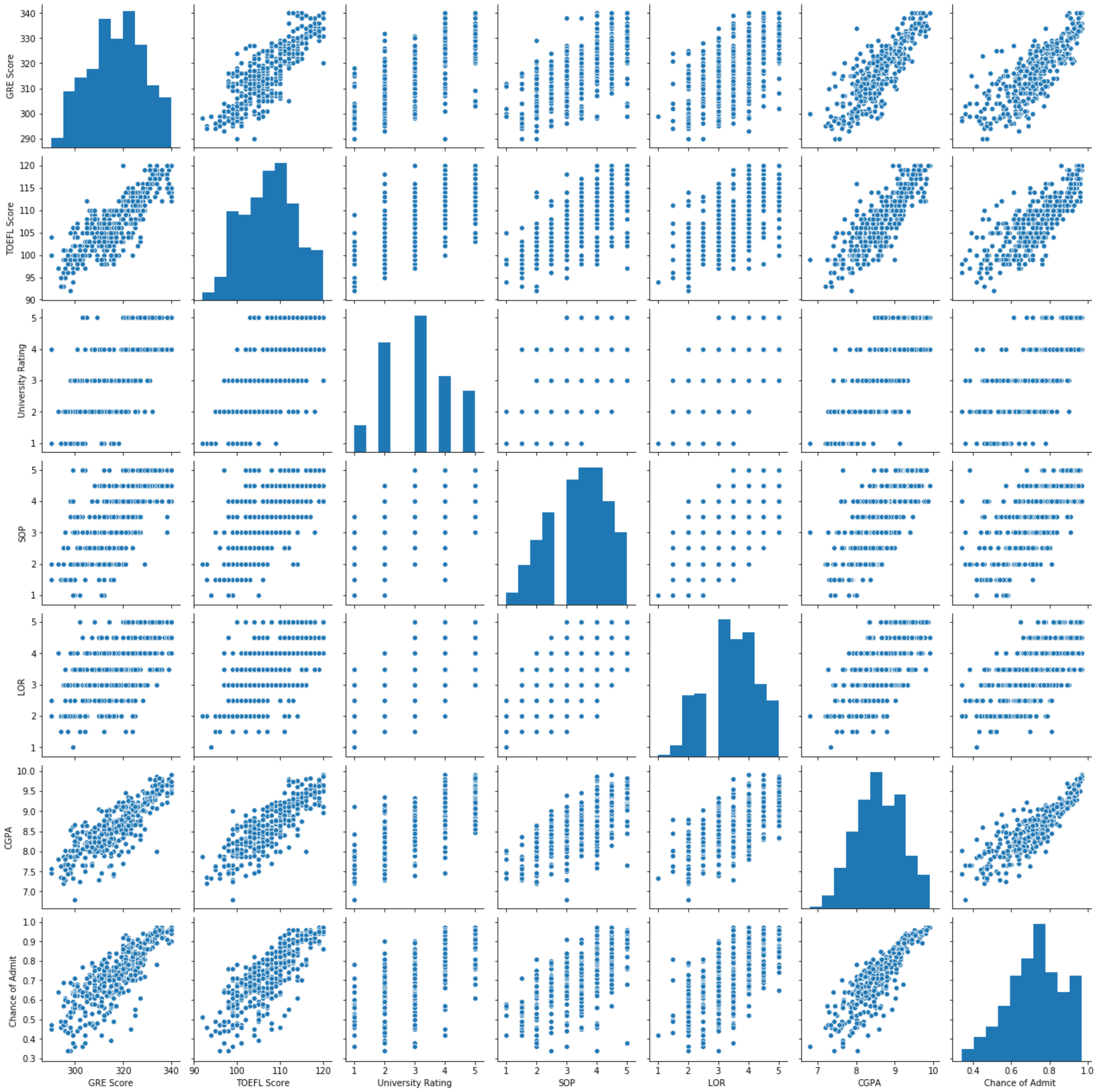


In [13]:

```
sns.pairplot(d.drop(columns='Research'))
```

Out[13]:

```
<seaborn.axisgrid.PairGrid at 0x12e69955688>
```

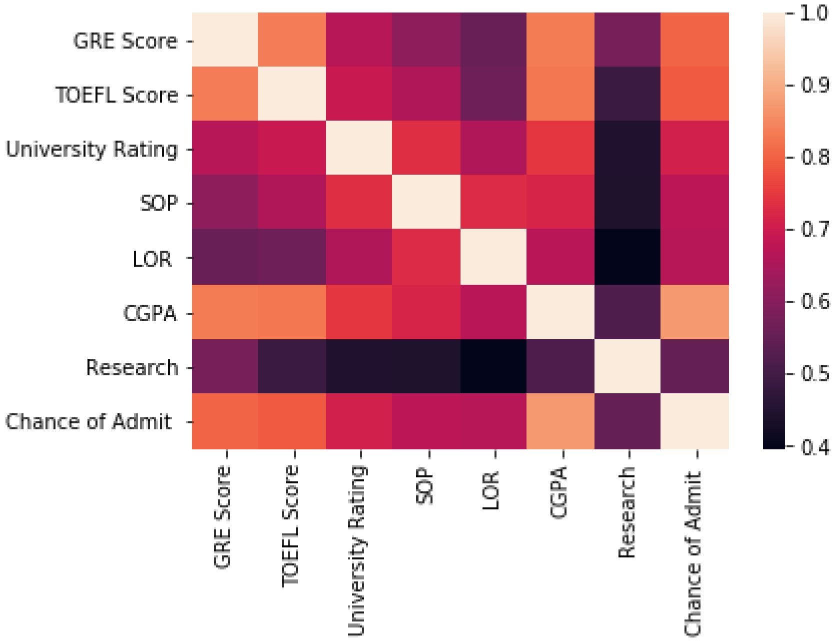


In [14]:

```
sns.heatmap(d.corr())
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12e6a71fbc8>
```



In [15]:

```
d['GRE Score'].unique()
```

Out[15]:

```
array([337, 324, 316, 322, 314, 330, 321, 308, 302, 323, 325, 327, 328,
       307, 311, 317, 319, 318, 303, 312, 334, 336, 340, 298, 295, 310,
       300, 338, 331, 320, 299, 304, 313, 332, 326, 329, 339, 309, 315,
       301, 296, 294, 306, 305, 290, 335, 333, 297, 293], dtype=int64)
```

In [16]:

```
d.columns.values
```

Out[16]:

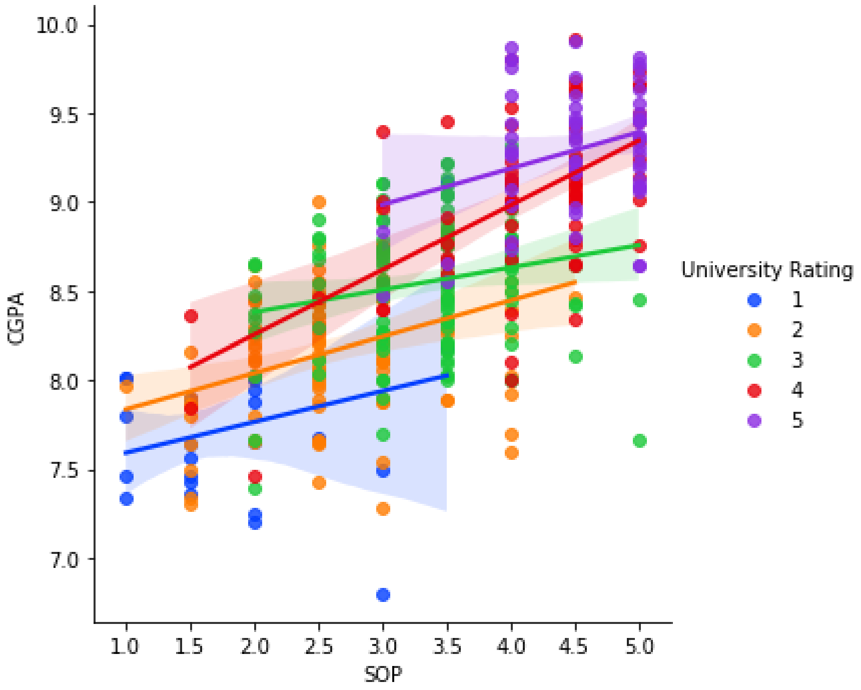
```
array(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ',
       'CGPA', 'Research', 'Chance of Admit '], dtype=object)
```

In [17]:

```
sns.lmplot(x='SOP',y='CGPA',data=d,hue='University Rating',palette='bright')
```

Out[17]:

```
<seaborn.axisgrid.FacetGrid at 0x12e6c54ab88>
```

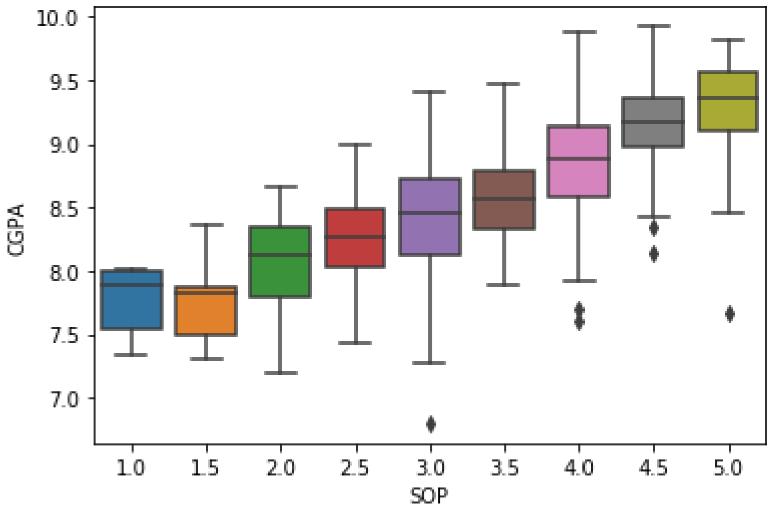


In [18]:

```
sns.boxplot(x="SOP",y="CGPA",data=d)
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12e6c614ac8>
```

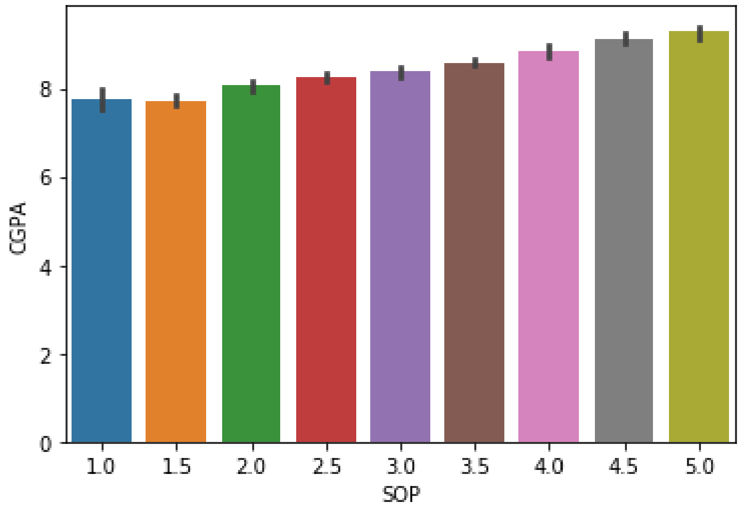


In [19]:

```
sns.barplot(x='SOP',y='CGPA',data=d)
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12e6c73b9c8>
```



In [20]:

```
x = d.drop(['Chance of Admit ','SOP'],axis=1) #SOP dropeed sue to high p-value
```

In [21]:

```
x
```

Out[21]:

	GRE Score	TOEFL Score	University Rating	LOR	CGPA	Research
0	337	118	4	4.5	9.65	1
1	324	107	4	4.5	8.87	1
2	316	104	3	3.5	8.00	1
3	322	110	3	2.5	8.67	1
4	314	103	2	3.0	8.21	0
...	...	...	...	...	...	...
395	324	110	3	3.5	9.04	1
396	325	107	3	3.5	9.11	1
397	330	116	4	4.5	9.45	1
398	312	103	3	4.0	8.78	0
399	333	117	4	4.0	9.66	1

400 rows × 6 columns

In [22]:

```
x.shape
```

Out[22]:

```
(400, 6)
```

g. FEATURE SCALING

In [23]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 5))
cols=x.columns
array=np.asarray(x[cols])
rs = scaler.fit_transform(array)
rs

```

Out[23]:

```
array([[4.7       , 4.64285714, 3.75      , 4.375     , 4.56730769,
        5.        ],
       [3.4       , 2.67857143, 3.75      , 4.375     , 3.31730769,
        5.        ],
       [2.6       , 2.14285714, 2.5       , 3.125     , 1.92307692,
        5.        ],
       ...,
       [4.        , 4.28571429, 3.75      , 4.375     , 4.24679487,
        5.        ],
       [2.2       , 1.96428571, 2.5       , 3.75      , 3.17307692,
        0.        ],
       [4.3       , 4.46428571, 3.75      , 3.75      , 4.58333333,
        5.        ]])
```

In [24]:

```
x= pd.DataFrame(data=rs,columns=cols)
x.head()
```

Out[24]:

	GRE Score	TOEFL Score	University Rating	LOR	CGPA	Research
0	4.7	4.642857	3.75	4.375	4.567308	5.0
1	3.4	2.678571	3.75	4.375	3.317308	5.0
2	2.6	2.142857	2.50	3.125	1.923077	5.0
3	3.2	3.214286	2.50	1.875	2.996795	5.0
4	2.4	1.964286	1.25	2.500	2.259615	0.0

In [30]:

```

```

h.SPLITTING DATA INTO TRAIN AND TEST

In [36]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.3,random_state=50)
```

In [37]:

```
x_train.shape
```

Out[37]:

```
(280, 6)
```

In [38]:

```
x_test.shape
```

Out[38]:

```
(120, 6)
```

In [39]:

```
y_train.shape
```

Out[39]:

```
(280,)
```

2.MODEL BUILDING

a. TRAINING AND TESTING

b.EVALUATION

linear regression

In [43]:

```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
predictions=reg.predict(x_test)
```

In [44]:

```
from sklearn.metrics import r2_score
R2=r2_score(y_test,predictions)
```

In [45]:

```
from sklearn.metrics import mean_squared_error
MSE=mean_squared_error(y_test,predictions)
```

In [46]:

```
R2
```

Out[46]:

```
0.6364126221433698
```

In [47]:

```
MSE
```

Out[47]:

```
0.005668084829199627
```

In [48]:

```
print(f'R-square= {round(R2*100,2)}% \nMean Squared Error= {"%.10f" %MSE}')
```

```
R-square= 63.64%
Mean Squared Error= 0.0056680848
```

Support vector Regressor (svm)

In [49]:

```
from sklearn.svm import SVR
clf = SVR()
clf.fit(x_train, y_train)
predictions=clf.predict(x_test)
predictions=clf.predict(x_test)
from sklearn.metrics import r2_score
R2=r2_score(y_test,predictions)
from sklearn.metrics import mean_squared_error
MSE=mean_squared_error(y_test,predictions)
print(f'R-square= {round(R2*100,2)}% \nMean Squared Error= {"%.10f" %MSE}')
```

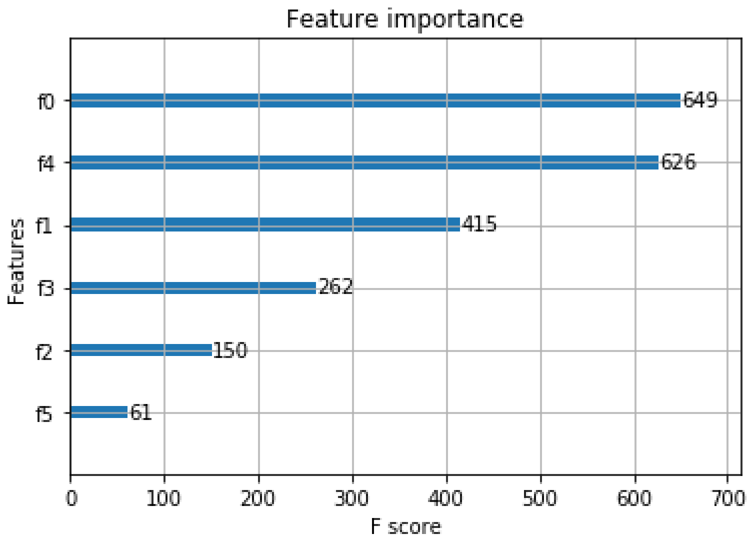
```
R-square= 43.54%
Mean Squared Error= 0.0088015847
```

XGB Regressor

In [50]:

```
from xgboost import XGBRegressor
XGBreg=XGBRegressor()
XGBreg.fit(x_train,y_train)
xgboost.plot_importance(XGBreg)
plt.show()

```



In [51]:

```
predictions=XGBreg.predict(x_test)
from sklearn.metrics import r2_score
R2=r2_score(y_test,predictions)
from sklearn.metrics import mean_squared_error
MSE=mean_squared_error(y_test,predictions)
print(f'R-square= {round(R2*100,2)}% \nMean Squared Error= {"%.10f" %MSE}')
```

```
R-square= 55.64%
Mean Squared Error= 0.0069148135
```

Random Forest Regressor

In [52]:

```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor(n_estimators = 100, random_state = 42)
rfr.fit(x_train,y_train)
predictions = rfr.predict(x_test)
from sklearn.metrics import r2_score
R2=r2_score(y_test,predictions)
from sklearn.metrics import mean_squared_error
MSE=mean_squared_error(y_test,predictions)
print(f'R-square= {round(R2*100,2)}% \nMean Squared Error= {"%.10f" %MSE}')
```

```
R-square= 56.58%
Mean Squared Error= 0.0067691823
```