Importing Libraries

In [340]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,r2_score
import warnings
warnings.filterwarnings("ignore")
```

Reading dataset and creating a dataframe

In [353]:
```
df = pd.read_csv("Admission_Predict.csv")
df
```

Out[353]:

	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
...	...	...	...	...	...	...	...	...	...
395	396	324	110	3	3.5	3.5	9.04	1	0.82
396	397	325	107	3	3.0	3.5	9.11	1	0.84
397	398	330	116	4	5.0	4.5	9.45	1	0.91
398	399	312	103	3	3.5	4.0	8.78	0	0.67
399	400	333	117	4	5.0	4.0	9.66	1	0.95

400 rows × 9 columns

In [354]:
```
df.describe()
```

Out[354]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	200.500000	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.547500	0.724350
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.498362	0.142609
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	0.340000
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.000000	0.640000
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.000000	0.730000
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.000000	0.830000
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.970000

In [357]:
```
df.shape
```

Out[357]:
```
(400, 9)
```

Checking for null values (if any)

In [342]:
```
df.isnull().sum()
```

Out[342]:
```
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

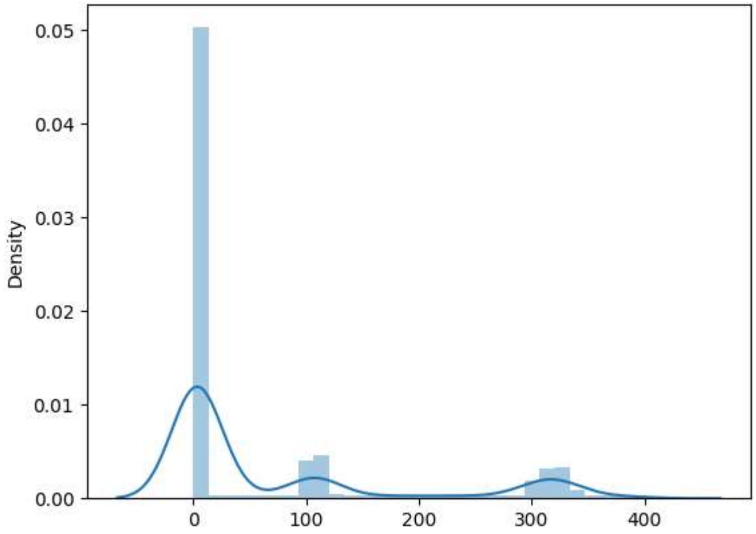
Distribution Plot

In [343]:

```
sns.distplot(df)
```

Out[343]:

```
<AxesSubplot:ylabel='Density'>
```



In [344]:

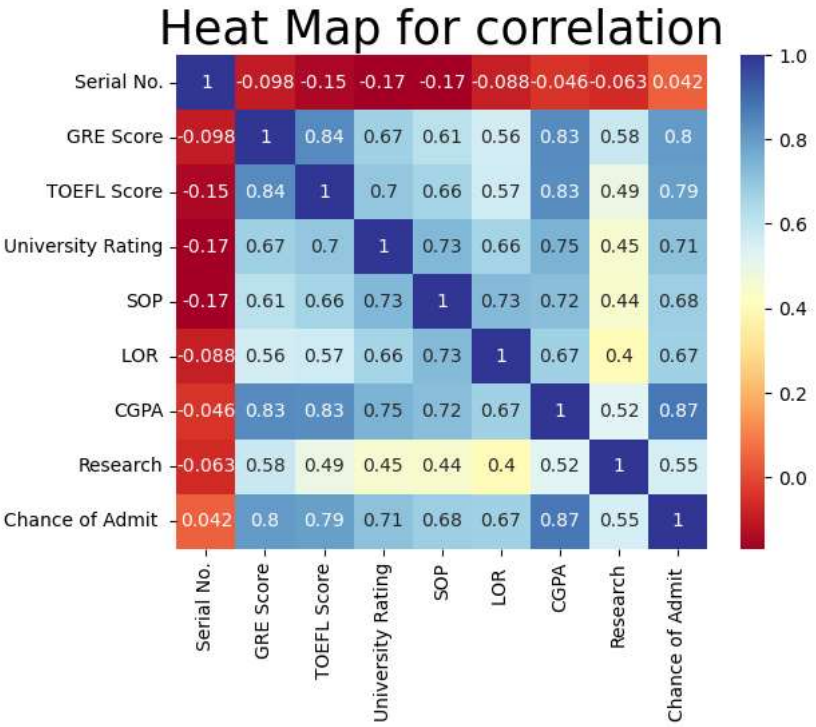
```
cor= pd.DataFrame(df.corr()['Chance of Admit '])
cor.rename({'Chance of Admit ': 'Correlation Coeffecient'}, axis=1, inplace=True)
cor.drop('Chance of Admit ', inplace=True)
cor.sort_values(['Correlation Coeffecient'], ascending=False, inplace=True)
cor_x = cor.index
cor_y =cor['Correlation Coeffecient']
cor
```

Out[344]:

	Correlation Coeffecient
CGPA	0.873289
GRE Score	0.802610
TOEFL Score	0.791594
University Rating	0.711250
SOP	0.675732
LOR	0.669889
Research	0.553202
Serial No.	0.042336

In [345]:
```
sns.heatmap(df.corr(),annot=True,cmap="RdYlBu").set_title('Heat Map for correlation',color='black',size='25')
```

Out[345]:
```
Text(0.5, 1.0, 'Heat Map for correlation')
```



In [346]:
```
x = df.drop(['Chance of Admit '], axis=1)
y = df['Chance of Admit ']
```

Standardisation of Data

In [347]:
```
sc = StandardScaler()
x = sc.fit_transform(x)
```

Splitting into Train and Test data

In [348]:
```
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,shuffle=True)
```

Fitting the data into Lasso Regression Model

In [349]:
```
L=Lasso(alpha=0.001).fit(x_train,y_train)
print("Lasso Score :",L.score(x_test,y_test))
alpha=[0.1,0.01,0.001,0.0001,0.005,0.015]
cv=GridSearchCV(estimator=L,param_grid=dict(alpha=alpha),scoring='r2').fit(x_train,y_train)
print("Best Score :",cv.best_score_)
print("Best Alpha :",cv.best_params_) #finding the best value for the hyper parameter Alpha
```

```
Lasso Score : 0.8497613646617911
Best Score : 0.7753449354845807
Best Alpha : {'alpha': 0.001}
```

In [350]:
```
L=Lasso(alpha=0.001).fit(x_train,y_train)
y_pred=L.predict(x_test)
mse=mean_squared_error(y_pred,y_test)
rmse=np.sqrt(mse)
r2=r2_score(y_pred,y_test)
print("MSE :",mse)
print("RMSE :",rmse)
print("R2 :",r2)
```

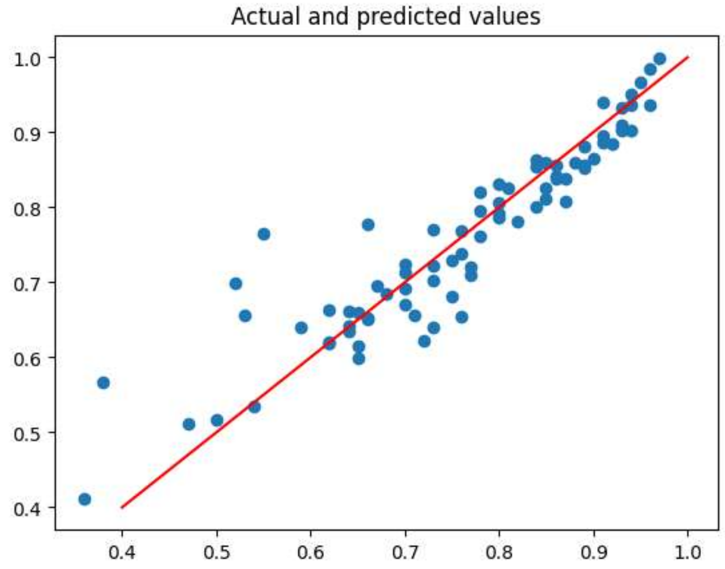
```
MSE : 0.0029201672277606727
RMSE : 0.05403857166654826
R2 : 0.807048337153556
```

```
In [351]: h=pd.DataFrame({"Actual": y_test, "Predicted": y_pred}).head()
          print("Actual and predicted values\n",h)
```

```
Actual and predicted values
      Actual  Predicted
297    0.86   0.855876
12     0.78   0.820860
70     0.94   0.935809
257    0.78   0.760853
4      0.65   0.599449
```

Scatter Plot between Actual and Predicted data to understand the correlation

```
In [352]: Actual=y_test
          Predicted=y_pred
          plt.scatter(Actual,Predicted)
          plt.plot([0.4,0.8,1],[0.4,0.8,1],color="red")
          plt.title("Actual and predicted values")
          plt.show()
```


Actual and predicted values