# Prediction of Graduate Admissions from an Indian perspective

## Importing Libraries

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

In [3]:

```
#Load CSV here
df = pd.read_csv("Admission_Dataset.csv")
```

In [7]:

```
df.sample(7)
```

Out[7]:

| | Serial No. | GRE Score | TOEFL Score | University Rating | SOP | LOR | CGPA | Research | Chance of Admit |
|---|---|---|---|---|---|---|---|---|---|
| **417** | 418 | 316 | 103 | 3 | 3.5 | 2.0 | 7.68 | 0 | 0.52 |
| **161** | 162 | 298 | 99 | 1 | 1.5 | 3.0 | 7.46 | 0 | 0.53 |
| **257** | 258 | 324 | 100 | 3 | 4.0 | 5.0 | 8.64 | 1 | 0.78 |
| **329** | 330 | 297 | 96 | 2 | 2.5 | 1.5 | 7.89 | 0 | 0.43 |
| **465** | 466 | 305 | 96 | 4 | 3.0 | 4.5 | 8.26 | 0 | 0.54 |
| **6** | 7 | 321 | 109 | 3 | 3.0 | 4.0 | 8.20 | 1 | 0.75 |
| **337** | 338 | 332 | 118 | 5 | 5.0 | 5.0 | 9.47 | 1 | 0.94 |

In [8]:

```
df.columns
```

Out[8]:

```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

In [9]:

```
df.shape
```

Out[9]:

```
(500, 9)
```

df.describe()

In [12]:

```
#Find missing value
df.isnull().sum()
```

Out[12]:

```
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
```

In [13]:

```
#finf duplicate value
df.duplicated().sum()
```

Out[13]:

```
0
```

In [14]:

```
df.head()
```

```
Out[14]:
```

| | 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 |

## Creating a copy and removing the Sl.No column

```
In [15]: df1 = df.copy()
         df1.drop(['Serial No.'], axis=1, inplace=True)
```

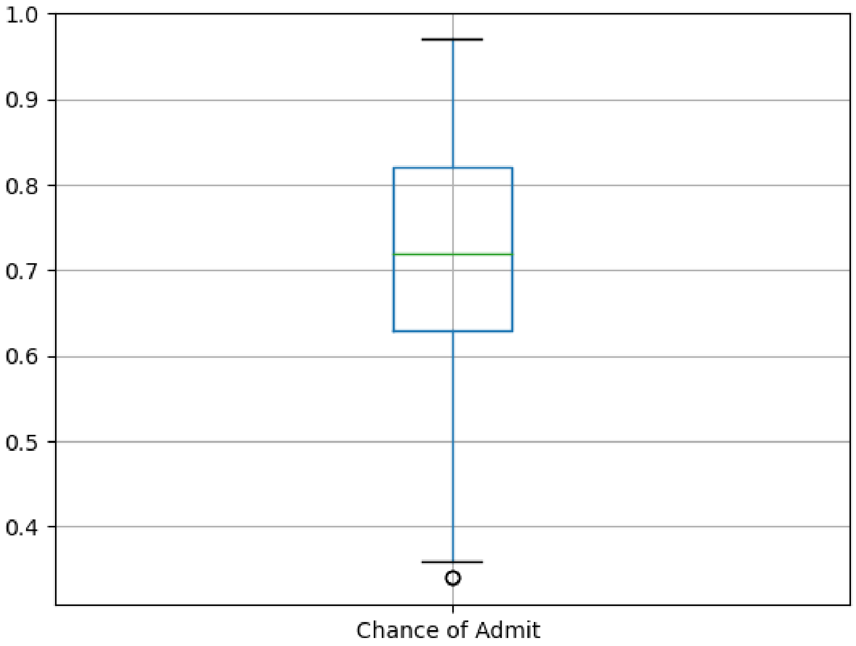
```
In [16]: df1.head()
```

```
Out[16]:
```

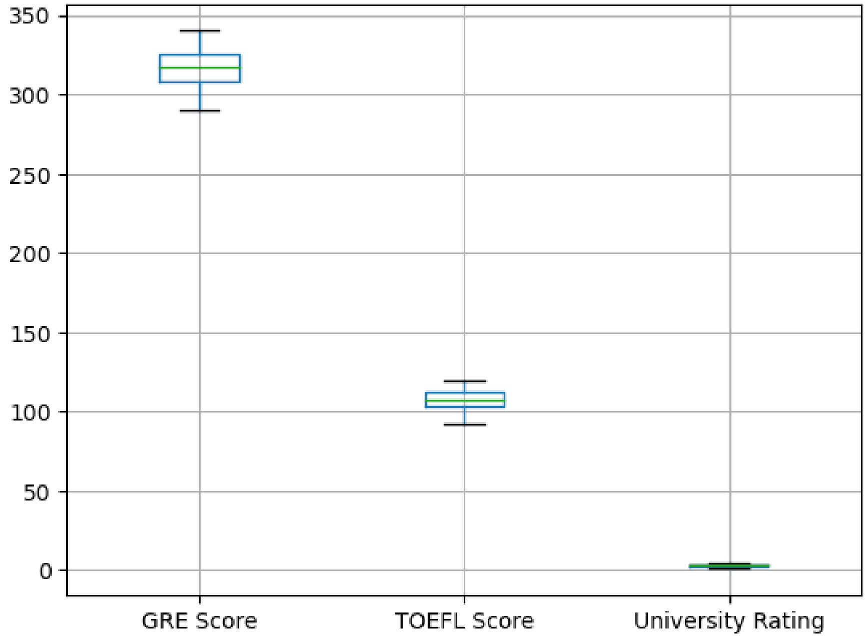
| | GRE Score | TOEFL Score | University Rating | SOP | LOR | CGPA | Research | Chance of Admit |
|---|---|---|---|---|---|---|---|---|
| **0** | 337 | 118 | 4 | 4.5 | 4.5 | 9.65 | 1 | 0.92 |
| **1** | 324 | 107 | 4 | 4.0 | 4.5 | 8.87 | 1 | 0.76 |
| **2** | 316 | 104 | 3 | 3.0 | 3.5 | 8.00 | 1 | 0.72 |
| **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 |

## Identifying & Removing outliers

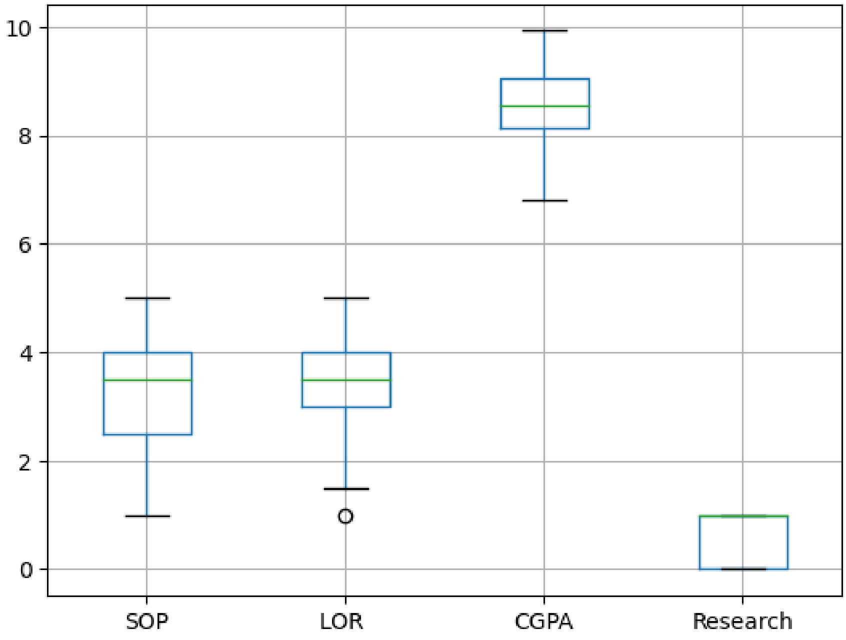
```
In [19]: df1.boxplot(column=['Chance of Admit '])
         plt.show()
```



```
In [20]: df1.boxplot(column=['GRE Score', 'TOEFL Score', 'University Rating'])
         plt.show()
```



In [21]:
```
df1.boxplot(column=['SOP','LOR ', 'CGPA', 'Research'])
plt.show()
```



we can see there are outliers in chance of admit & LOR columns.

In [23]:
```
Q1 = df1.quantile(0.25)
Q3 = df1.quantile(0.75)
IQR=Q3-Q1
IQR
```

Out[23]:
```
GRE Score            17.0000
TOEFL Score           9.0000
University Rating     2.0000
SOP                   1.5000
LOR                   1.0000
CGPA                  0.9125
Research              1.0000
Chance of Admit       0.1900
dtype: float64
```

In [28]:
```
#upper limit
UL=Q3+IQR*1.5
print("Upper Limit:")
print(UL,"\n\n\n")

#lower limit
print("Lower Limit:")
```

```
LL=Q1-IQR*1.5
print(LL)
```

```
Upper Limit:
GRE Score              350.50000
TOEFL Score            125.50000
University Rating        7.00000
SOP                      6.25000
LOR                      5.50000
CGPA                    10.40875
Research                 2.50000
Chance of Admit          1.10500
dtype: float64



Lower Limit:
GRE Score              282.50000
TOEFL Score             89.50000
University Rating       -1.00000
SOP                      0.25000
LOR                      1.50000
CGPA                     6.75875
Research                -1.50000
Chance of Admit          0.34500
dtype: float64
```

In [29]:

```
#remove outliers based on the lower and upper limits
df_outliers_removed = df1[(df1>LL) & (df1<UL)]
df_outliers_removed
```

Out[29]:

| | 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 [30]:

```
#checking null values
df_outliers_removed.isnull().sum()
```

Out[30]:

```
GRE Score             0
TOEFL Score           0
University Rating     0
SOP                   0
LOR                  12
CGPA                  0
Research              0
Chance of Admit       2
dtype: int64
```

In [31]:

```
#Drop the null values
df_outliers_removed.dropna(inplace=True)
```

In [32]:

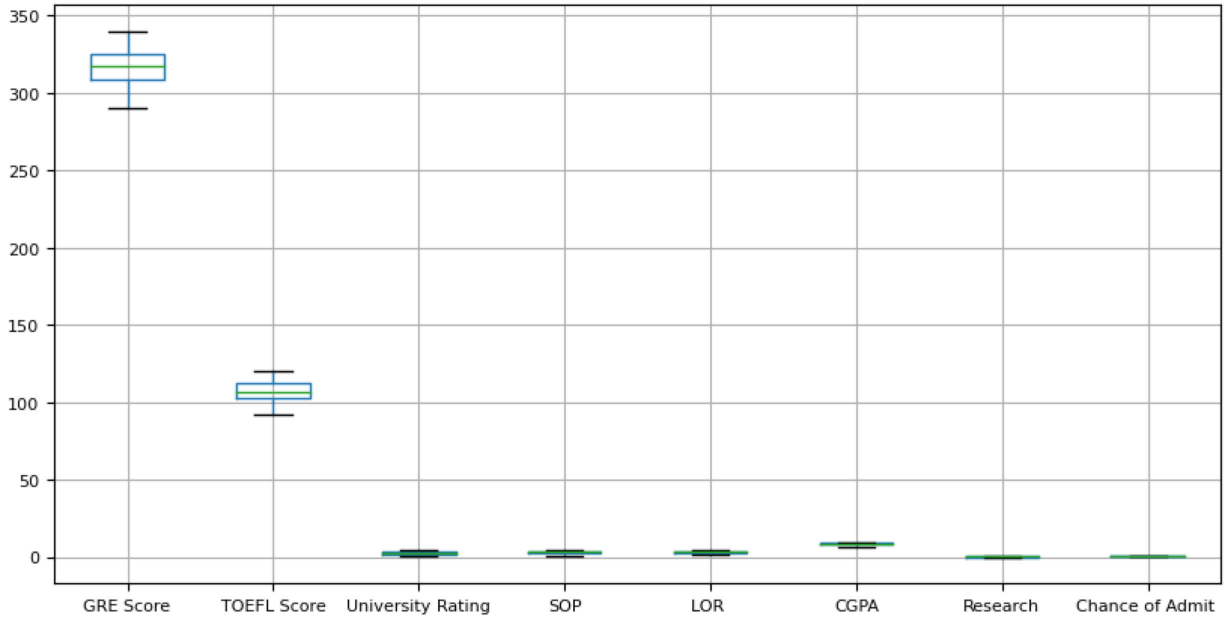
```
df_outliers_removed.shape
```

Out[32]:

```
(486, 8)
```

In [35]:

```
df_outliers_removed.boxplot(figsize=(10,5), fontsize=8)
plt.show()
```

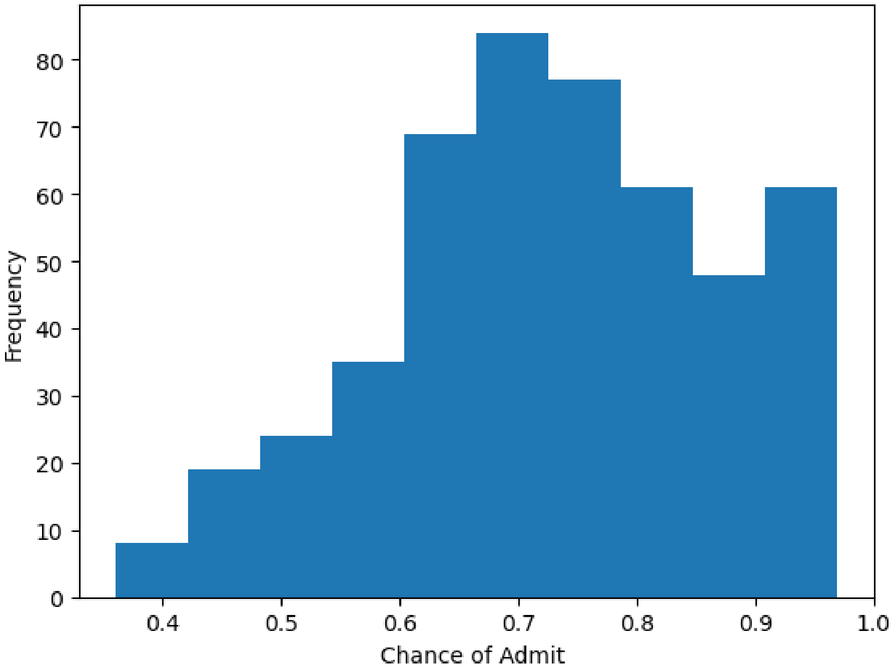


### we can see there are no outliers anymore.

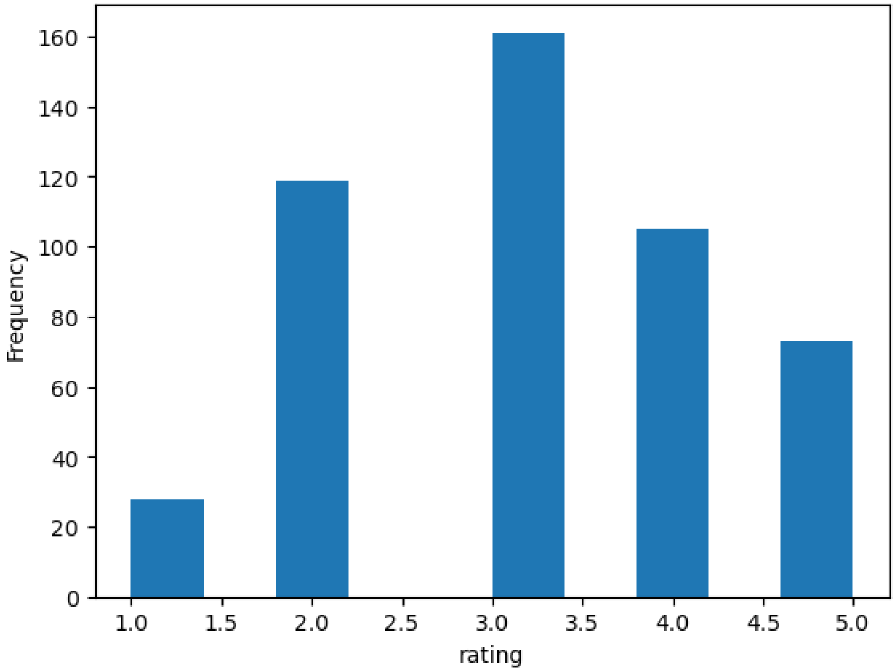
```
In [36]: df2 = df_outliers_removed.copy()
```

## Univariate analysis

```
In [37]: df2['Chance of Admit '].plot.hist()
         plt.xlabel('Chance of Admit ')
         plt.show()
```



```
In [39]: df2['University Rating'].plot.hist()
         plt.xlabel('rating')
         plt.show()
```



see the maximun no.of students are getting rating from 3 to 3.5

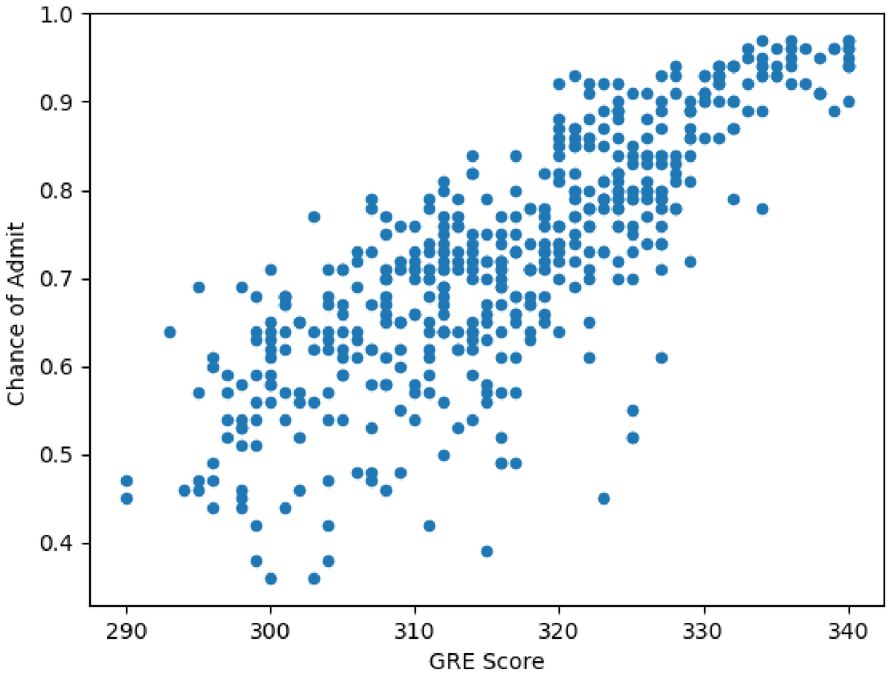
```
In [40]: df2['Research'].value_counts()
```

```
Out[40]: 1    277
         0    209
         Name: Research, dtype: int64
```

277 students have research experience and 209 students have no experience

## Bi-variate analysis

```
In [41]: df2.plot.scatter('GRE Score','Chance of Admit ')
         plt.show()
```

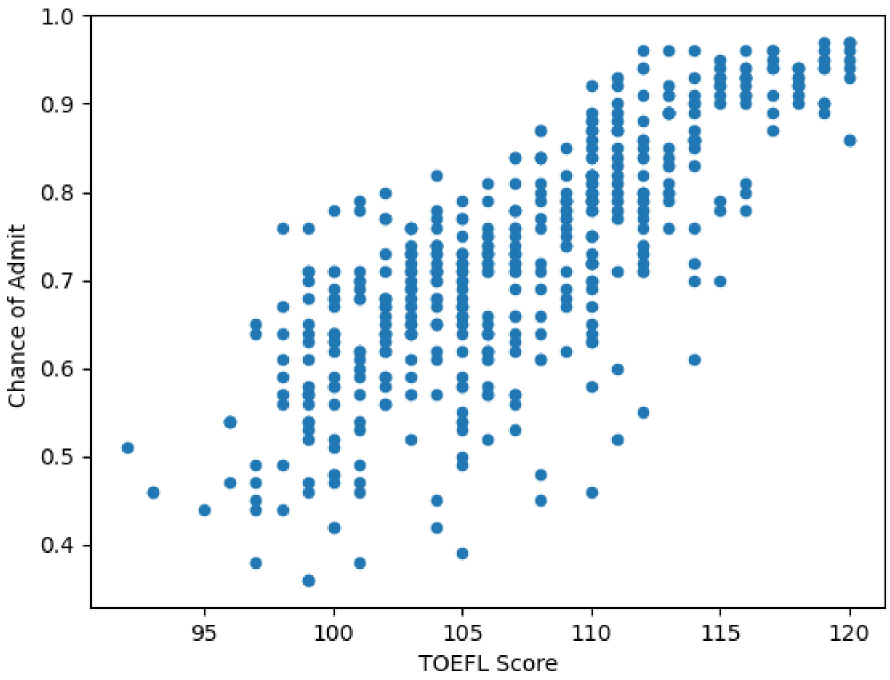


```
In [42]: df2['Chance of Admit '].corr(df2['GRE Score'])
```

```
Out[42]: 0.803189604437301
```

chance of admit and GRE score are positively correlated. if GRE score increases there is more chance of getting admission.

In [43]:
```
df2.plot.scatter('TOEFL Score','Chance of Admit ')
plt.show()
```

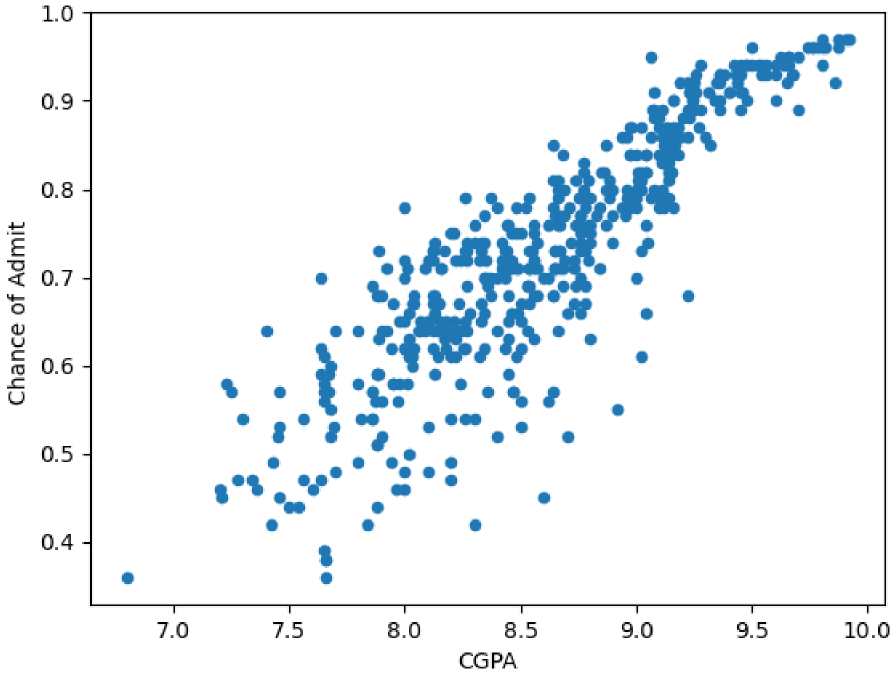


In [44]:
```
df2['TOEFL Score'].corr(df2['Chance of Admit '])
```

Out[44]:
```
0.7857296232445918
```

chance of admit and TOEFL score are positively correlated. if TOEFL score increases there is more chance of getting admission.

In [46]:
```
df2.plot.scatter('CGPA','Chance of Admit ')
plt.show()
```

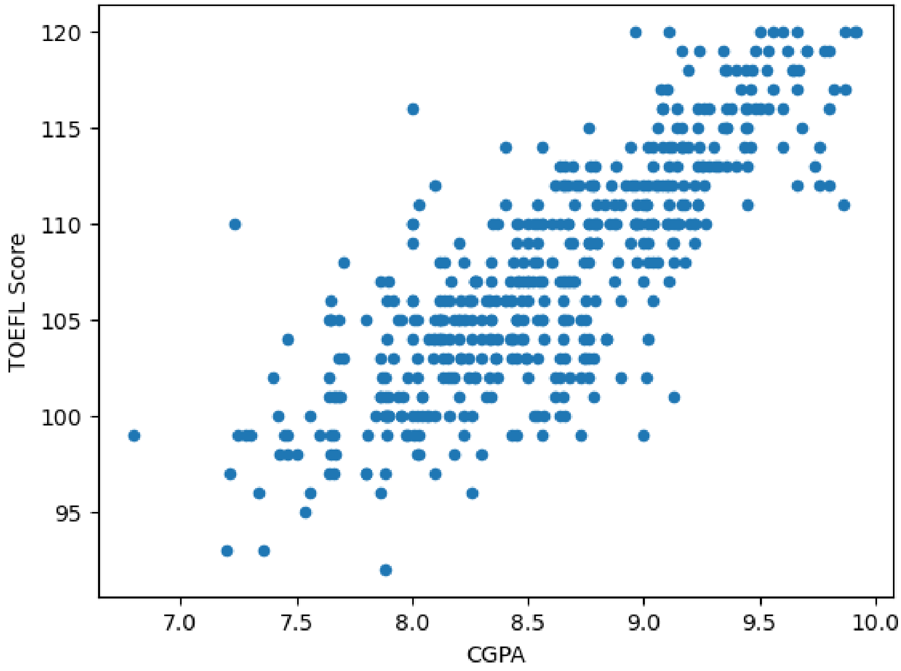


In [47]:
```
df2['CGPA'].corr(df2['Chance of Admit '])
```

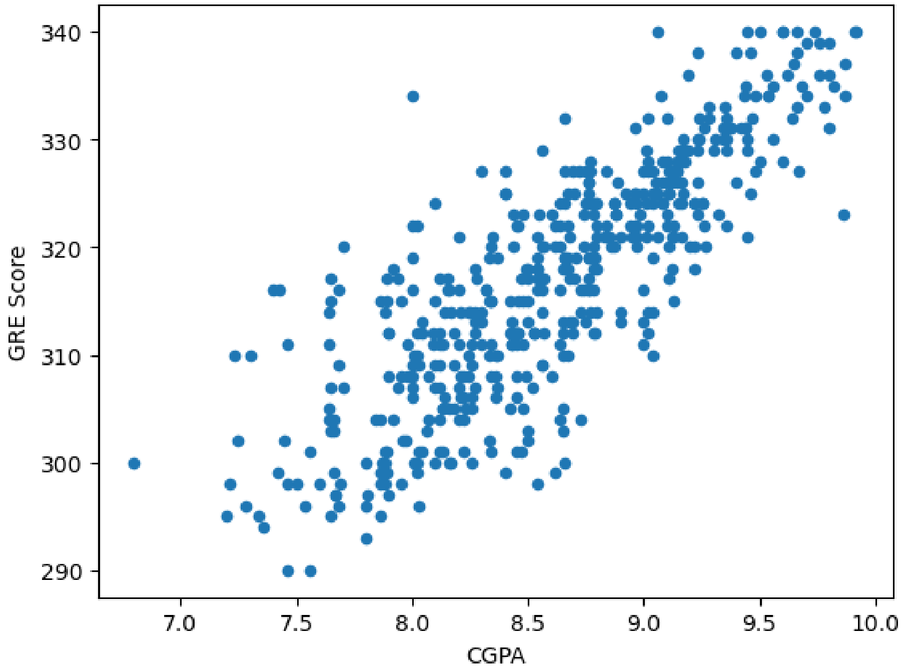
Out[47]:
```
0.8821495912854789
```

chance of admit and CGPA are positively correlated. if CGPA increases there is more chance of getting admission.

In [48]:
```
df2.plot.scatter('CGPA','TOEFL Score')
plt.show()
```



In [49]:
```
df2.plot.scatter('CGPA','GRE Score')
plt.show()
```



In [50]:
```
df2['CGPA'].corr(df2['GRE Score'])
```

Out[50]:
```
0.8208424849253341
```

In [51]:
```
df2['CGPA'].corr(df2['TOEFL Score'])
```

Out[51]:
```
0.8081094221483263
```

Students who have good CGPA , will definitely get a good score in TOEFL and GRE exams.

## Now, we'll Separating x and y

In [52]:
```
x=df2.drop(['Chance of Admit '],axis=1)
y=df2['Chance of Admit ']
x.shape,y.shape
```

Out[52]:
```
((486, 7), (486,))
```

In [53]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression as LR

x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=56)
```

## Fit data into linear model

In [55]:

```
lr = LR()
```

In [56]:

```
lr.fit(x_train, y_train)
```

Out[56]:

```
▾ LinearRegression
LinearRegression()
```

## Predicting over train and test set

In [59]:

```
from sklearn.metrics import mean_absolute_error as mae, r2_score, mean_squared_error, mean_absolute_error
from math import sqrt

pre_train = lr.predict(x_train)
mae_train = mae(pre_train, y_train)
mae_train
```

Out[59]:

```
0.04052008959676385
```

In [60]:

```
pre_test=lr.predict(x_test)
mae_test=mae(pre_test,y_test)
mae_test
```

Out[60]:

```
0.04345173324962816
```

# Model Evaluation

In [61]:

```
n = len(x_train)
m=len(x_test)
```

## Train data

In [63]:

```
RMSE = np.sqrt(mean_squared_error(y_train,pre_train))
MSE = mean_squared_error(y_train, pre_train)
MAE = mean_absolute_error(y_train, pre_train)
r2_train = r2_score(y_train, pre_train)
adj_r2 = 1-(1-r2_train)*(n-1)/(n-mae_train-1)
print(RMSE)
print(MSE)
print(MAE)
print(r2_train)
print(adj_r2)
```

```
0.0572018808365434
0.0032720551712381108
0.04052008959676385
0.8186071138689355
0.8185868635203288
```

## Test data

In [64]:

```
RMSE_test = np.sqrt(mean_squared_error(y_test,pre_test))
MSE_test = mean_squared_error(y_test, pre_test)
MAE_test = mean_absolute_error(y_test, pre_test)
r2_test = r2_score(y_test, pre_test)
adj_r2_test = 1-(1-r2_test)*(m-1)/(m-mae_test-1)
print(RMSE_test)
print(MSE_test)
print(MAE_test)
print(r2_test)
print(adj_r2_test)
```

```
0.06207177414999459
0.003852905146127937
0.04345173324962816
0.8081700586095103
0.8081011467270034
```

## Accuracy of the model

In [65]:

```
print('Accuracy of train set :',r2_train)
print('Accuracy of test set :',r2_test)
```

```
Accuracy of train set : 0.8186071138689355
Accuracy of test set : 0.8081700586095103
```

You can find this project on **GitHub.**