Prediction of Graduate Admissions from an Indian perspective

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression as LR

from sklearn.metrics import mean_absolute_error as mae, r2_score, mean_squar

from math import sqrt
```

In [3]:
```
## Loading the csv file
df=pd.read_csv("Admission_Predict_Ver1.1.csv")
```

In [6]:
```
df.sample(5)
```

Out[6]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
325	326	326	116	3	3.5	4.0	9.14	1	0.81
381	382	319	105	3	3.0	3.5	8.67	1	0.73
121	122	334	119	5	4.5	4.5	9.48	1	0.94
463	464	304	107	3	3.5	3.0	7.86	0	0.57
90	91	318	106	2	4.0	4.0	7.92	1	0.64

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SO
P',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

In [8]:
```
df.shape
```

Out[8]:
```
(500, 9)
```

In [9]:
```
df.describe()
```

Out[9]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	50
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	

Missing values

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
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 [11]:
```
df.duplicated().sum()
```

Out[11]:
```
0
```

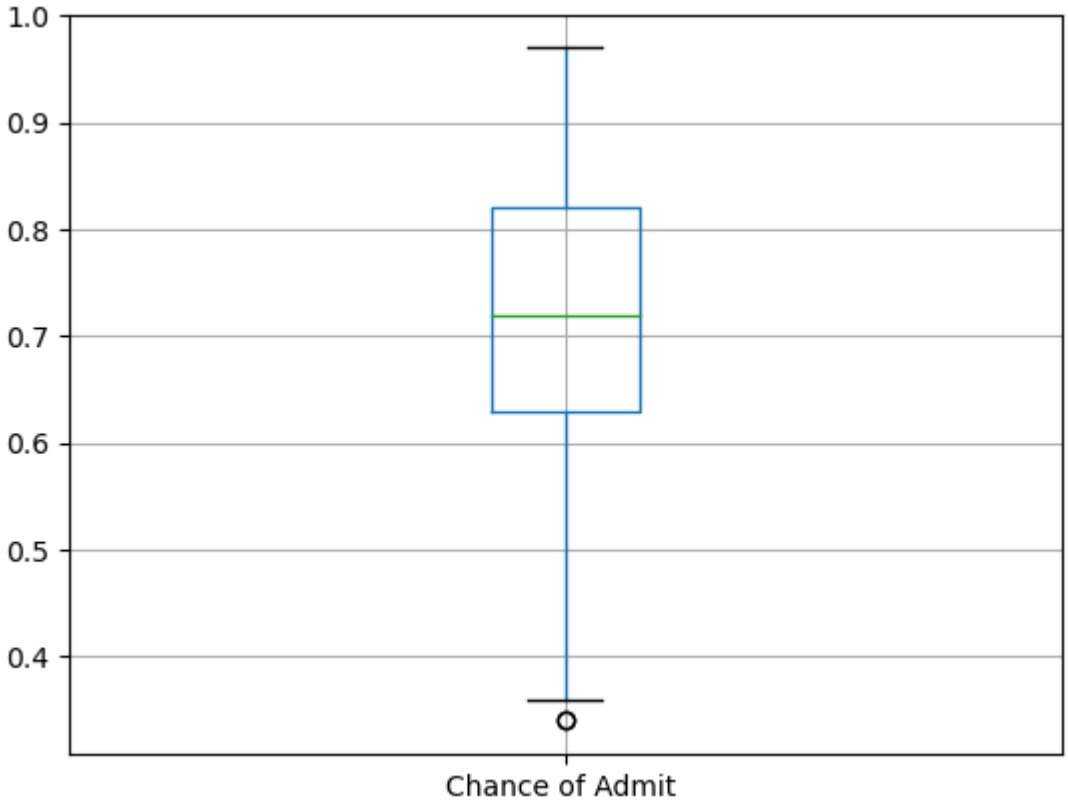
Creating a copy and removing the Sl.No column

In [12]:
```
df1=df.copy()
df1.drop(['Serial No.'],axis=1,inplace=True)
```

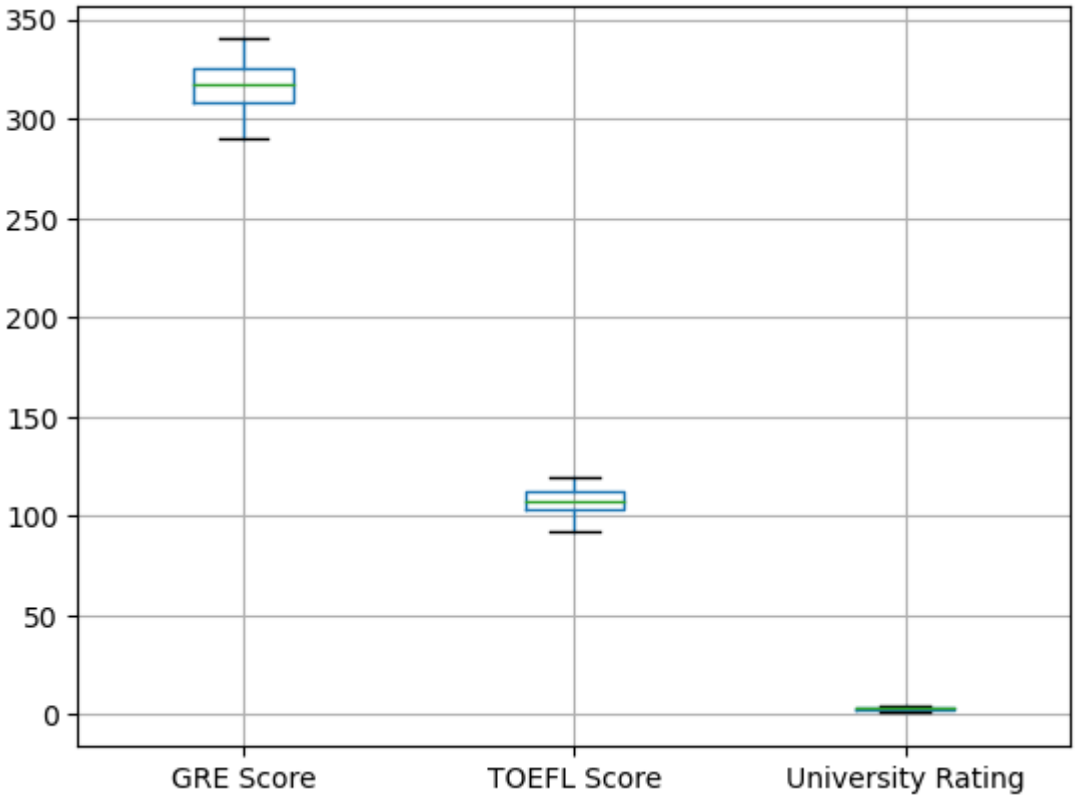
There are no missing and duplicated values in the dataset

Identifying & Removing outliers

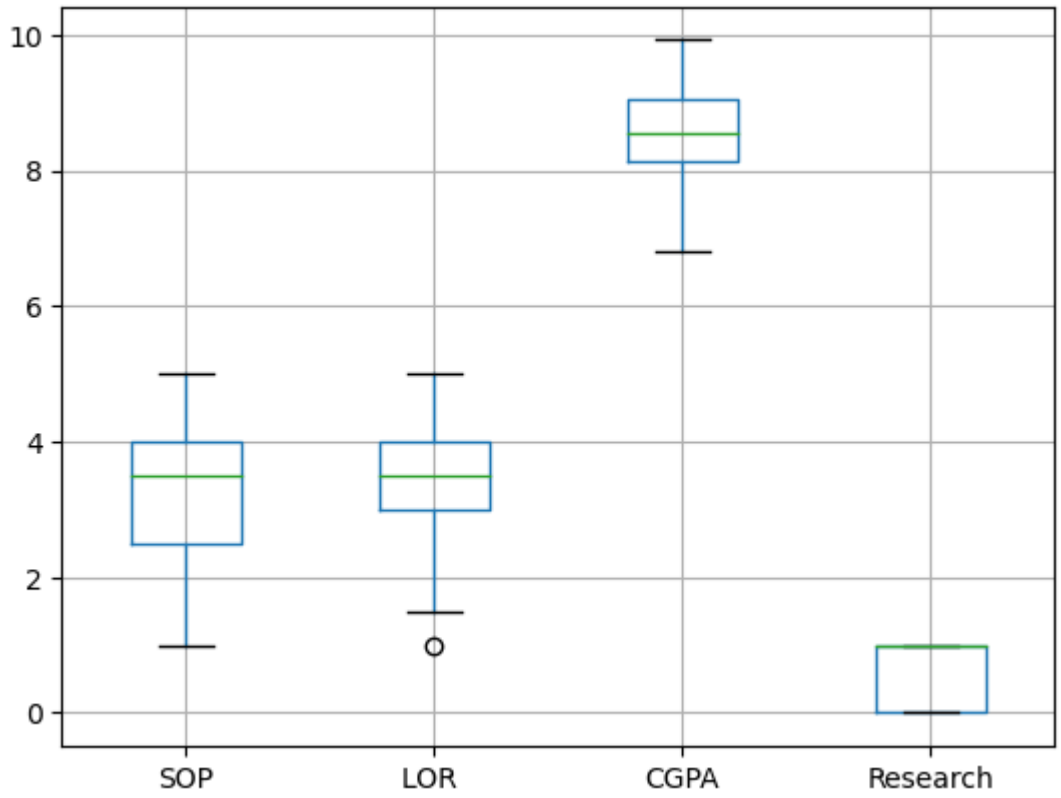
In [14]:
```
df1.boxplot(column=['Chance of Admit '])
plt.show()
```



In [15]:
```
df1.boxplot(column=['GRE Score', 'TOEFL Score', 'University Rating'])
plt.show()
```



In [17]:
```
df1.boxplot(column=['SOP','LOR ', 'CGPA', 'Research'])
plt.show()
```



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

In [19]:
```
Q1 = df1.quantile(0.25)
Q3 = df1.quantile(0.75)
IQR=Q3-Q1
IQR
```

Out[19]:
```
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 [20]:
```
#upper limit
UL=Q3+IQR*1.5
print(UL)

#lower limit
LL=Q1-IQR*1.5
print(LL)
```

```
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
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 [21]:
```
df_outliers_removed = df1[(df1>LL) & (df1<UL)]
df_outliers_removed
```

Out[21]:

	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 [22]:
```
df_outliers_removed.isnull().sum()
```

Out[22]:
```
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                 12
CGPA                 0
Research             0
Chance of Admit      2
dtype: int64
```

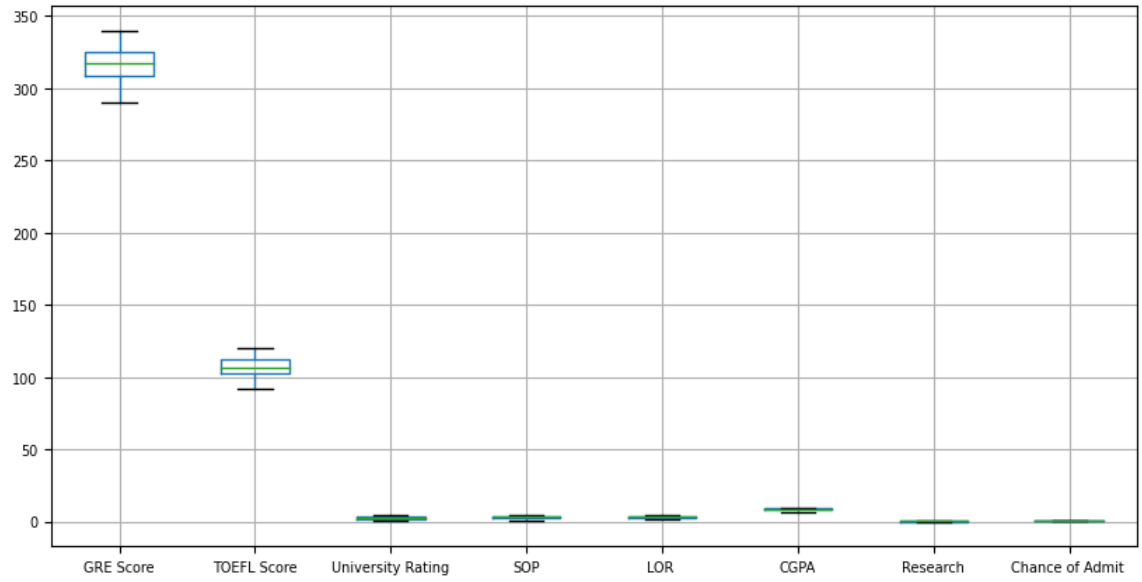
Droppimg the null values

In [23]:
```
df_outliers_removed.dropna(inplace=True)
```

In [24]:
```
df_outliers_removed.shape
```

Out[24]:
```
(486, 8)
```

In [26]:
```
df_outliers_removed.boxplot(figsize=(10,5), fontsize=7)
plt.show()
```

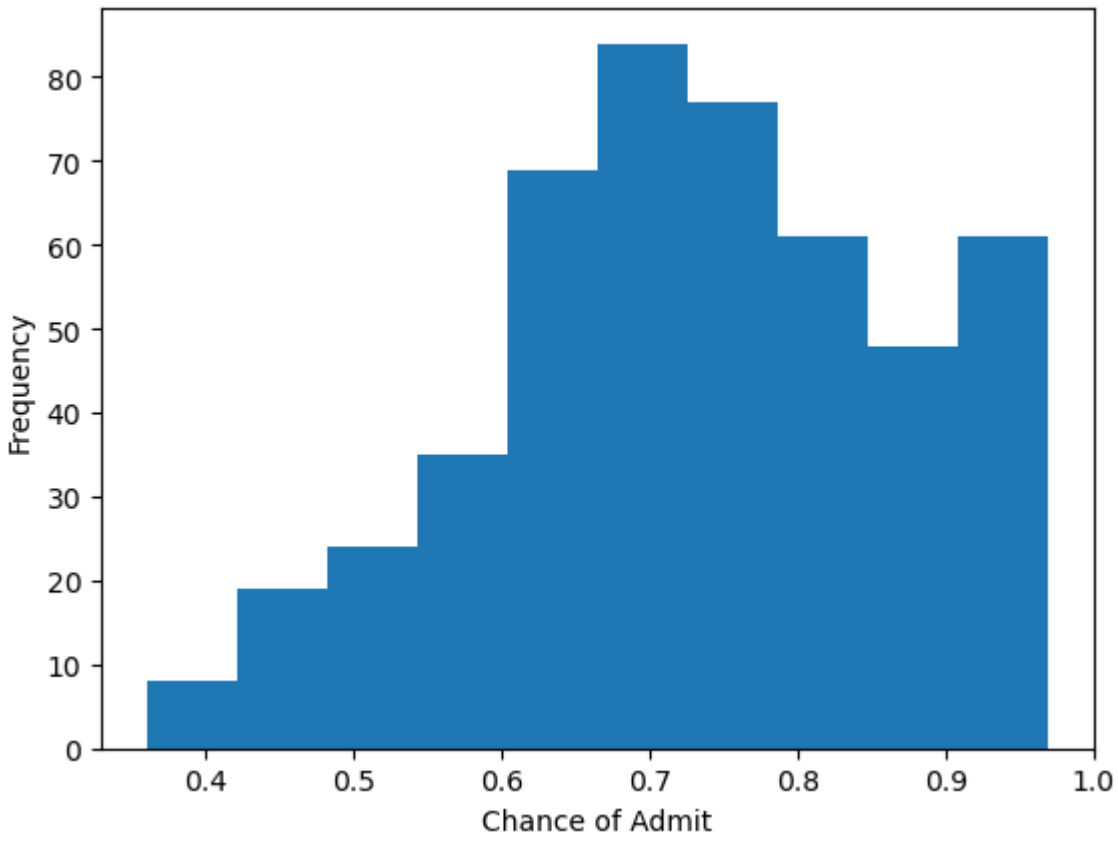


As we can see there are no outliers anymore.

In [27]:
```
df2=df_outliers_removed.copy()
```

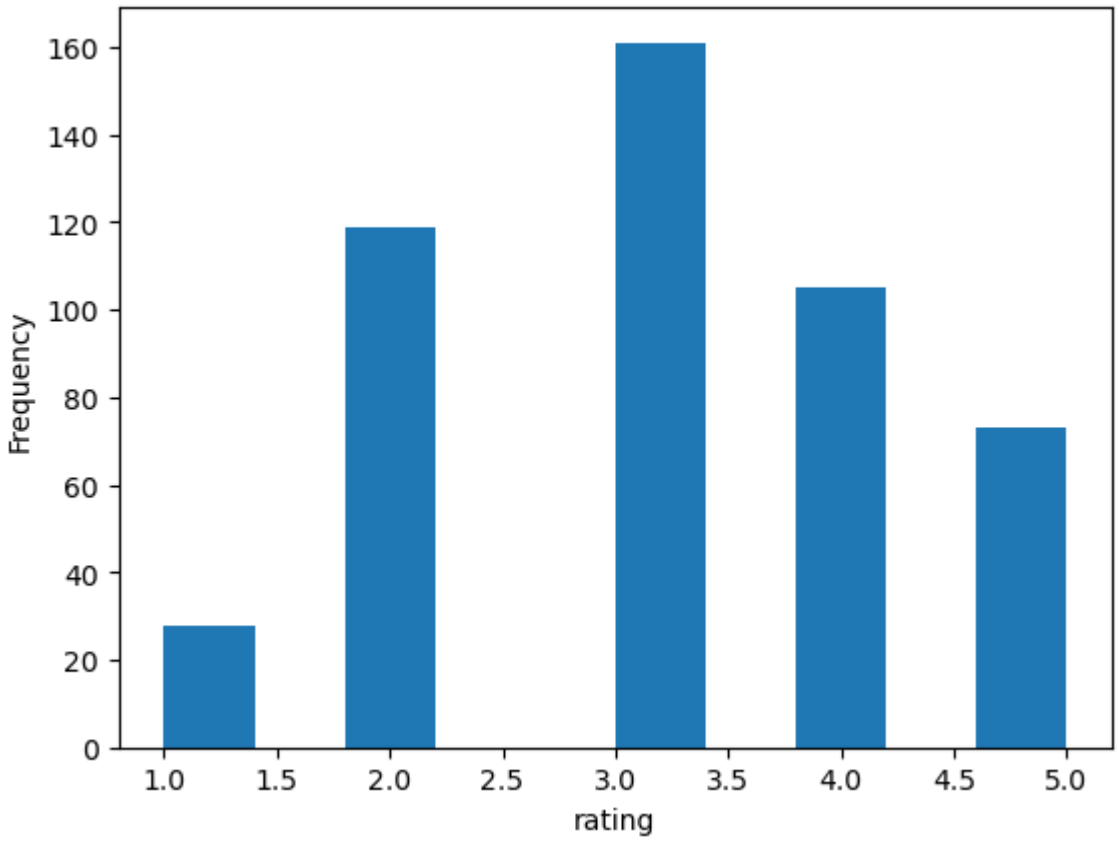
Univariate analysis

In [28]:
```
df2['Chance of Admit '].plot.hist()
plt.xlabel('Chance of Admit')
plt.show()
```



There is some variation in data,so it is useful for the prediction.

In [29]:
```
df2['University Rating'].plot.hist()
plt.xlabel('rating')
plt.show()
```



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

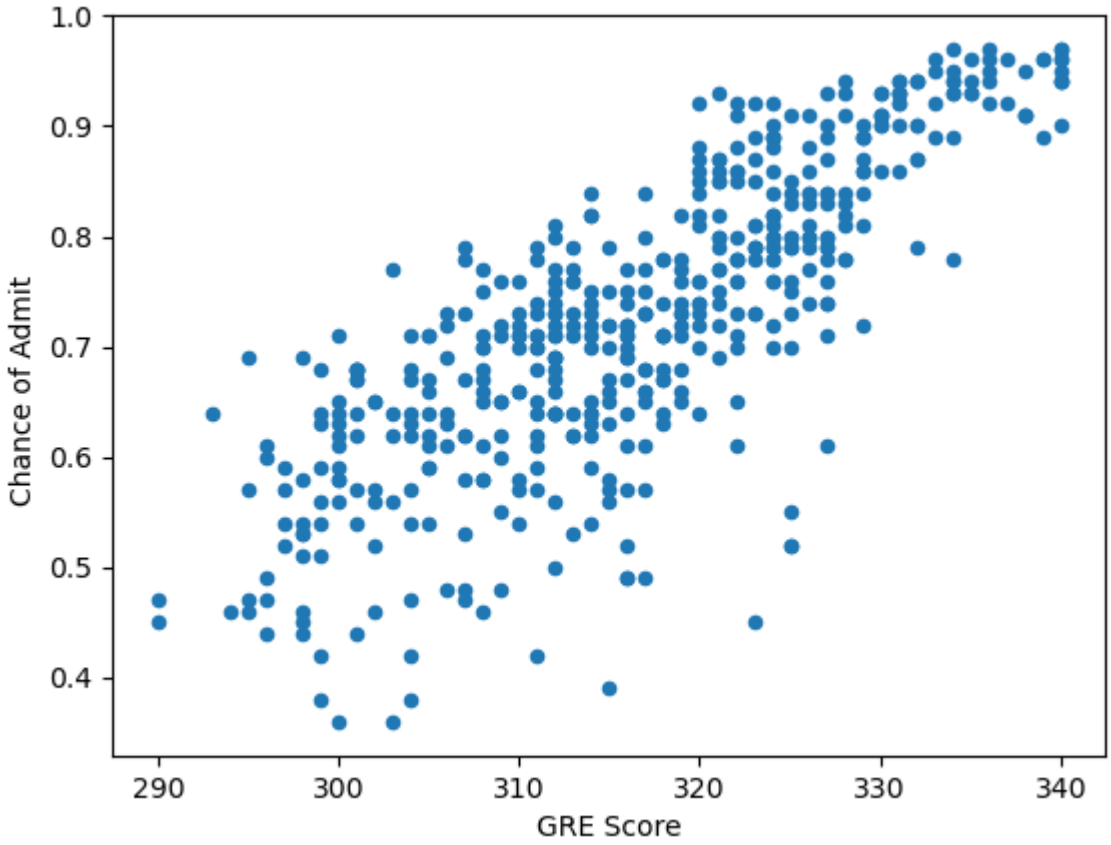
In [30]:
```
df2['Research'].value_counts()
```

Out[30]:
```
1    277
0    209
Name: Research, dtype: int64
```

We can say that 277 students have research experience and 209 students have no experience

Bi-variate analysis

In [31]:
```
df2.plot.scatter('GRE Score','Chance of Admit ')
plt.show()
```

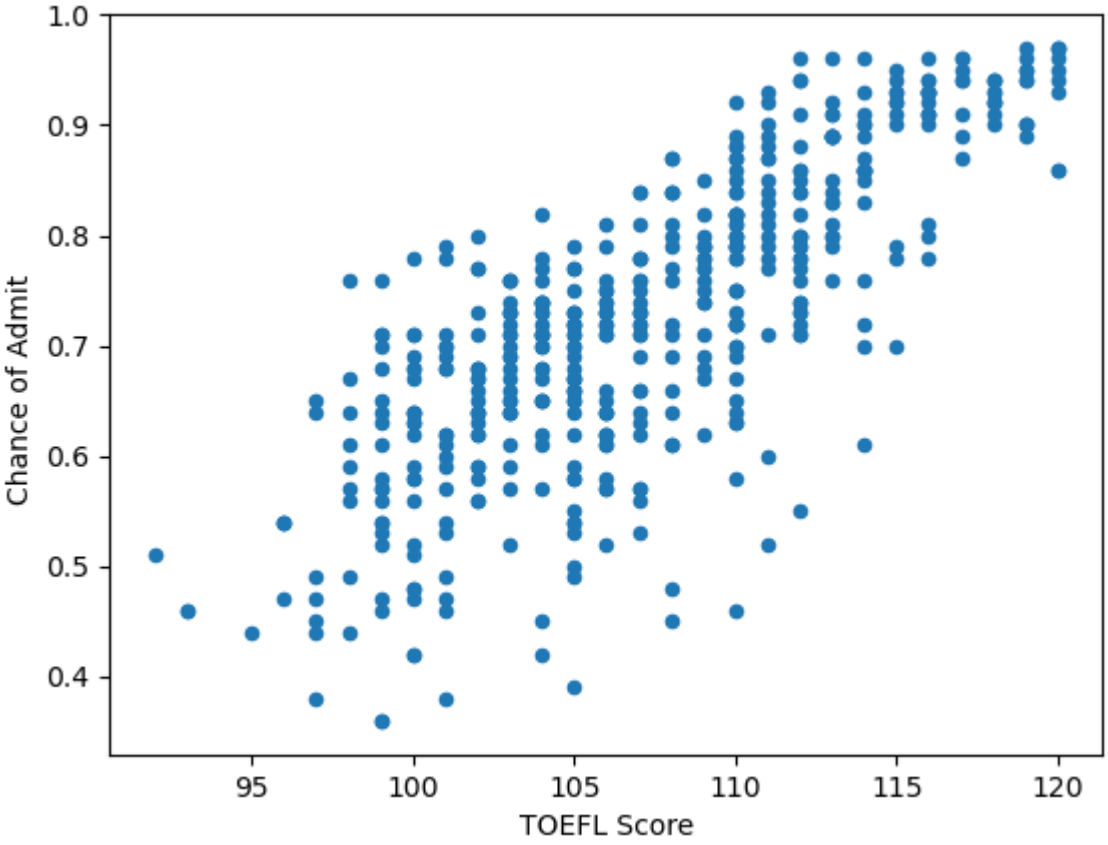


In [32]:
```
df2['Chance of Admit '].corr(df2['GRE Score'])
```

Out[32]:
```
0.803189604437301
```

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

In [33]:
```
df2.plot.scatter('TOEFL Score','Chance of Admit ')
plt.show()
```

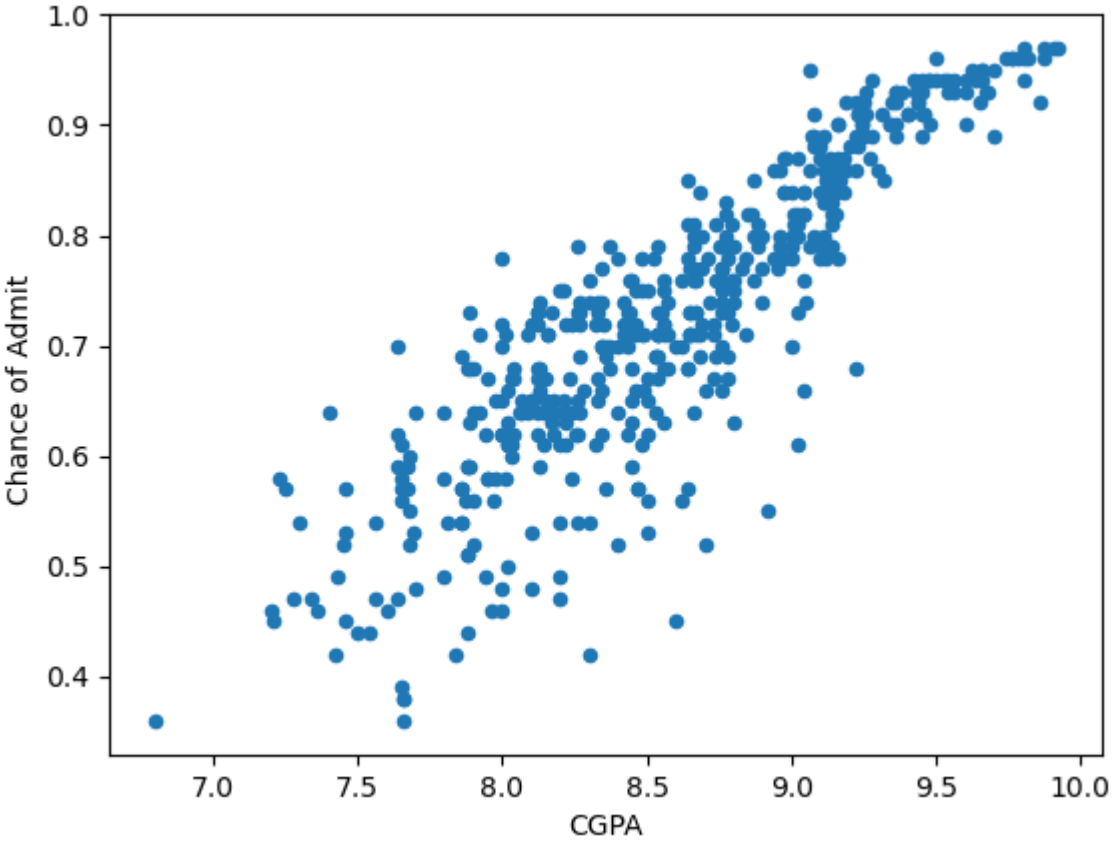


In [34]:
```
df2['TOEFL Score'].corr(df2['Chance of Admit '])
```

Out[34]:
```
0.7857296232445918
```

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

In [35]:
```
df2.plot.scatter('CGPA','Chance of Admit ')
plt.show()
```

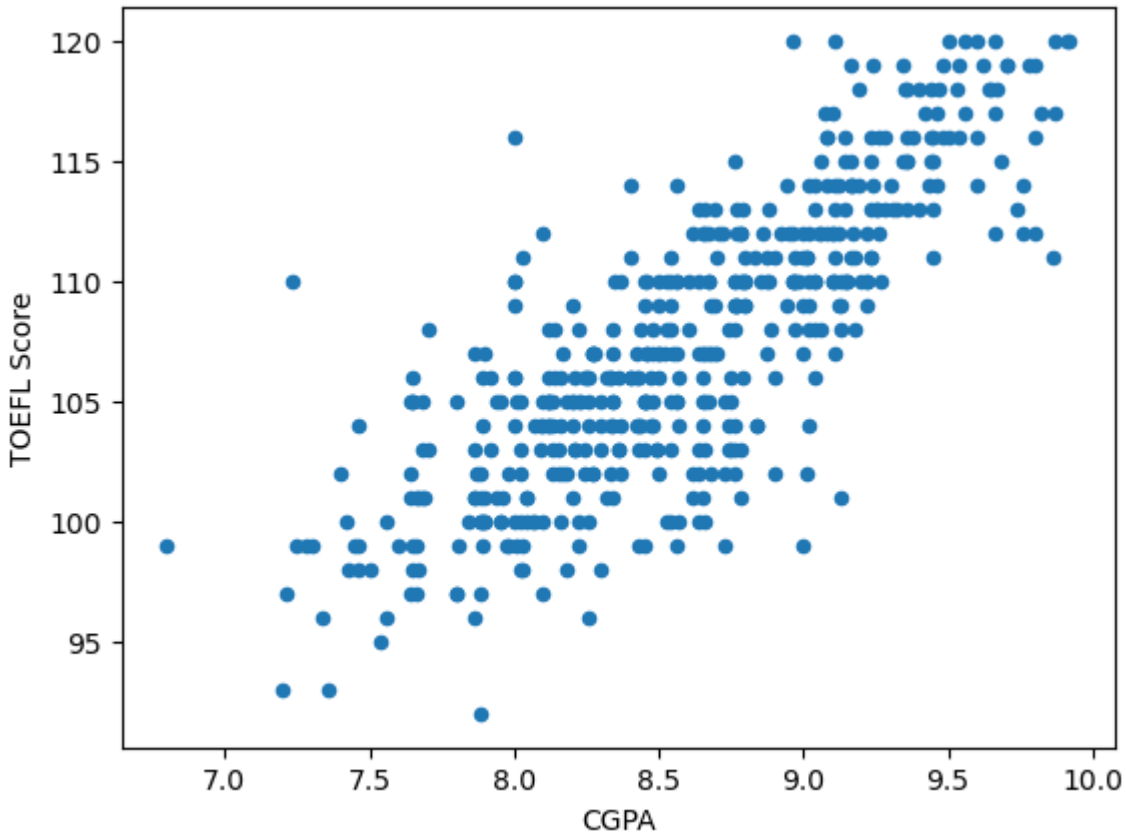


In [36]:
```
df2['CGPA'].corr(df2['Chance of Admit '])
```

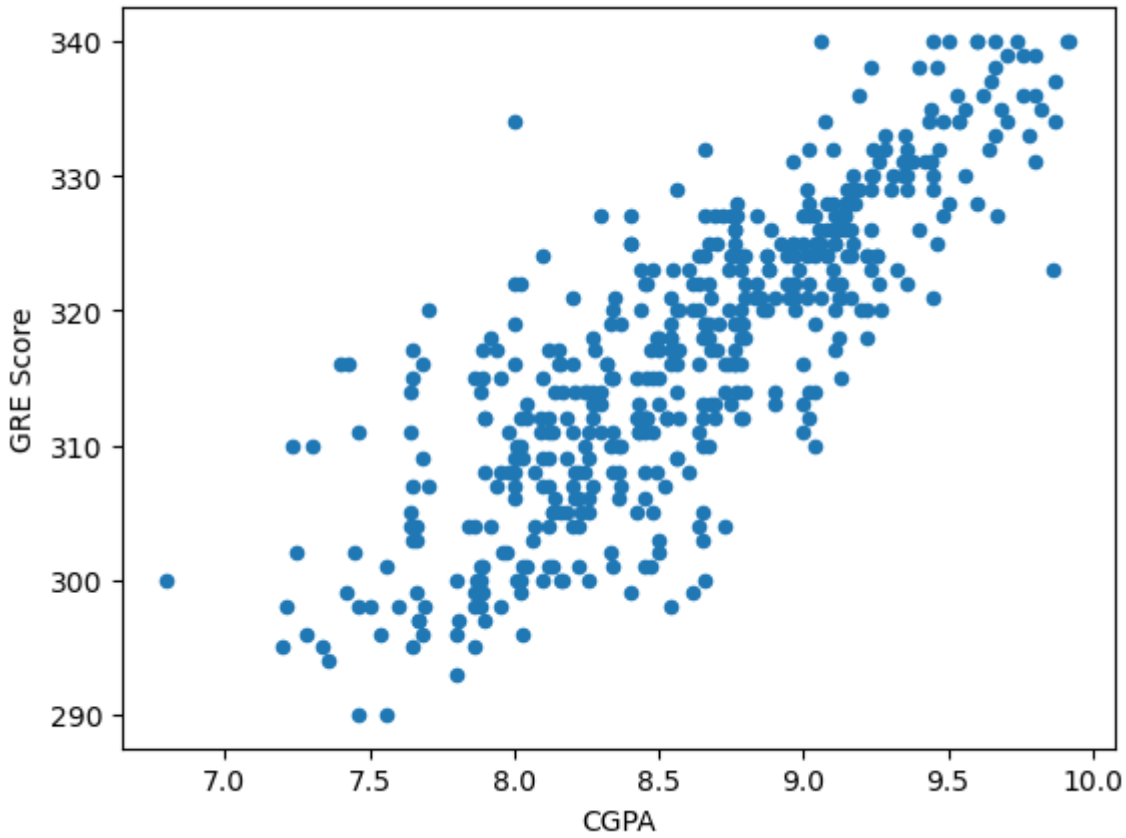
Out[36]:
```
0.8821495912854789
```

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

In [37]:
```
df2.plot.scatter('CGPA','TOEFL Score')
plt.show()
```



In [38]:
```
df2.plot.scatter('CGPA','GRE Score')
plt.show()
```



In [39]:
```
df2['CGPA'].corr(df2['GRE Score'])
```

Out[39]:
```
0.8208424849253341
```

In [40]:
```
df2['CGPA'].corr(df2['TOEFL Score'])
```

Out[40]:
```
0.8081094221483263
```

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

Separating x and y

In [41]:
```
x=df2.drop(['Chance of Admit '],axis=1)
y=df2['Chance of Admit ']
x.shape,y.shape
```

Out[41]:
```
((486, 7), (486,))
```

In [42]:
```
train_x,test_x,train_y,test_y=train_test_split(x,y,random_state=56)
```

Fitting the data into a linear regression model

In [43]:
```
lr=LR()
```

In [44]:
```
lr.fit(train_x,train_y)
```

Out[44]:
```
LinearRegression
LinearRegression()
```

Predicting over train and test set

In [45]:
```
train_pre=lr.predict(train_x)
mae_train=mae(train_pre,train_y)
```

In [46]:
```
mae_train
```

Out[46]:
```
0.04052008959676385
```

In [47]:
```
test_pre=lr.predict(test_x)
mae_test=mae(test_pre,test_y)
```

In [48]:
```
mae_test
```

Out[48]:
```
0.04345173324962816
```

Model Evaluation

In [49]:
```
n = len(train_x)
m=len(test_x)
```

Train data

In [50]:
```
RMSE = np.sqrt(mean_squared_error(train_y,train_pre))
MSE = mean_squared_error(train_y, train_pre)
MAE = mean_absolute_error(train_y, train_pre)
r2_train = r2_score(train_y, train_pre)
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 [51]:
```
RMSE_test = np.sqrt(mean_squared_error(test_y,test_pre))
MSE_test = mean_squared_error(test_y, test_pre)
MAE_test = mean_absolute_error(test_y, test_pre)
r2_test = r2_score(test_y, test_pre)
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 [52]:
```
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
```

Thank You!

GitHub: https://github.com/anujtiwari21?tab=repositories (https://github.com/anujtiwari21?tab=repositories)