Evalution-01 ML

Q1

Define a function that will return the Mean Square, Mean absolute error and root mean Squared error.

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [79]:

```
Actual_Value = [2,6,45,7,3,89,3,7,8,23,56,87]
Predicted_Val = [5,8,34,9,4,67,3,8,52,34,40,78]

Actual_Val = np.array(Actual_Value)
Predicted_Val = np.array(Actual_Value)
```

In [82]:

```
X = Actual_Val[:,np.newaxis]
y = Predicted_Val[:,np.newaxis]
```

In [83]:

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()

lr.fit(X,y)
```

Out[83]:

LinearRegression

LinearRegression()

In [84]:

```
y_pred = lr.predict(X)
```

In [87]:

```
from sklearn import metrics
```

In [89]:

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y, y_pred)))
print('Root Squared Error:', metrics.r2_score(y, y_pred))
```

```
Mean Absolute Error: 4.255854927729767e-15
Mean Squared Error: 2.4865553116653975e-29
Root Mean Squared Error: 4.986537186931827e-15
Root Squared Error: 1.0
```

Q2

Find out the value of c and m after 500 epochs

which one is better to user m=1 or m = 0 before iteration

Step size for first iteration

In [95]:

```
explanatory_variable=np.array ([1,2,4,3,34,78,5,4,3,5])
explained_variable=np.array ([1,3,3,2,45,6,7,0,345,2])
```

In [98]:

```
while epoch<500:

    epoch=epoch+1
    counter=0
    for x in explanatory_variable:

        yhat=(m*x)+c
        error=yhat-explained_variable[counter]
        c=c-(LR*error)
        m=m-(LR*error*x)
        counter=counter+1

print("The final value of m", m)
print("The final value of c", c)
```

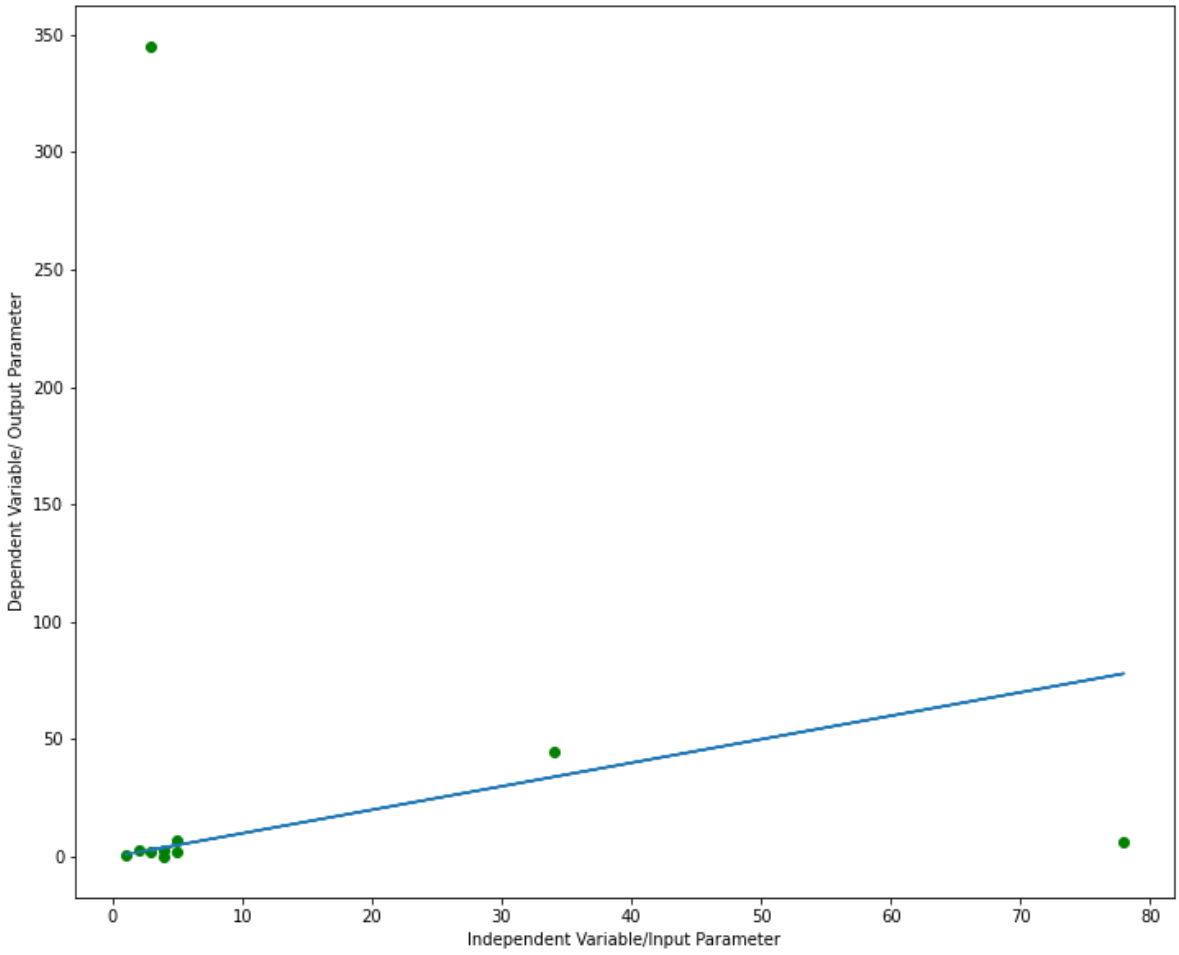
```
The final value of m nan
The final value of c nan
```

In [ ]:

```
# LR=0.002
# m=0
# c=0
# epoch=0
# sample_size=10
```

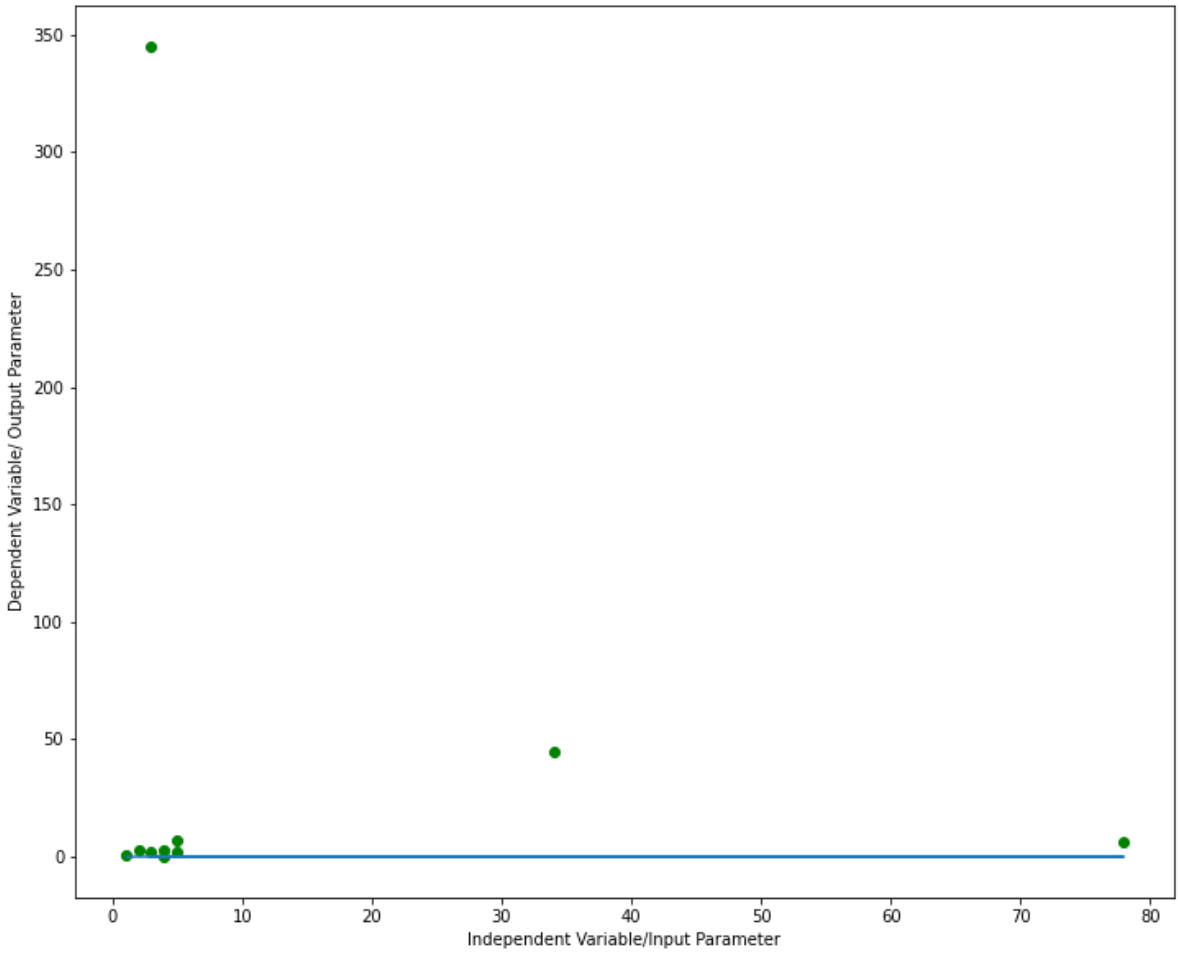
In [99]:

```
m=1
c=0
plt.figure(figsize=(12, 10))
plt.scatter(explanatory_variable, explained_variable, color='green')
plt.plot(explanatory_variable, m*explanatory_variable+c)
plt.xlabel('Independent Variable/Input Parameter')
plt.ylabel('Dependent Variable/ Output Parameter')
plt.show()
```



In [100]:

```
m=0
c=0
plt.figure(figsize=(12, 10))
plt.scatter(explanatory_variable, explained_variable, color='green')
plt.plot(explanatory_variable, m*explanatory_variable+c)
plt.xlabel('Independent Variable/Input Parameter')
plt.ylabel('Dependent Variable/ Output Parameter')
plt.show()
```



In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

Q3.Please refer the dataset for this

1) Plot Correlations heatmap

In [40]:

```
df = pd.read_excel("ML_EVAL.xlsx")
```

In [26]:

```
# df.drop(['Serial No.'],axis=1,inplace=True)
```

In [41]:

```
df.head()
```

Out[41]:

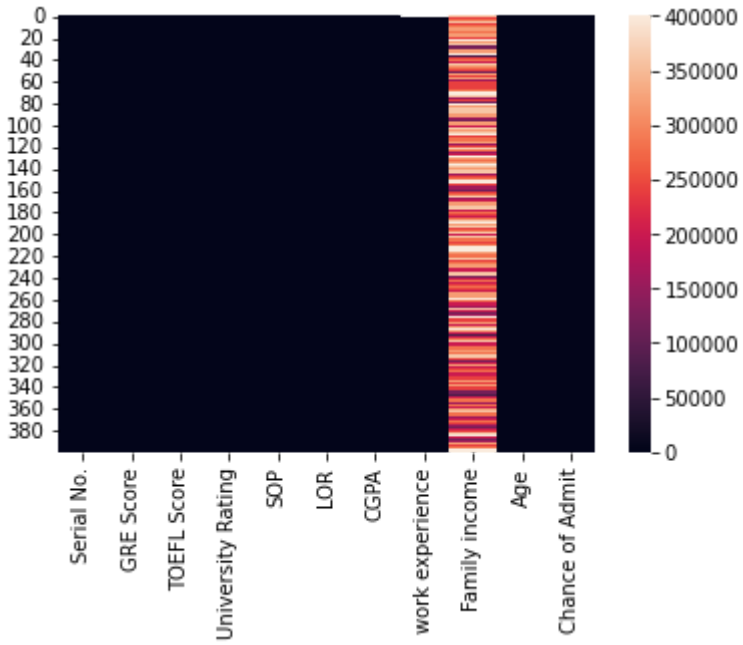
	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	work experience	Family income	Age	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	NaN	360000	18	0.92
1	2	324	107	4	4.0	4.5	8.87	1.0	320000	19	0.76
2	3	316	104	3	3.0	3.5	8.00	1.0	240000	20	0.72
3	4	322	110	3	3.5	2.5	8.67	1.0	280000	21	0.80
4	5	314	103	2	2.0	3.0	8.21	0.0	160000	22	0.65

In [42]:

```
sns.heatmap(df)
```

Out[42]:

<AxesSubplot:>



In [44]:

```
df2 = df.drop(['Serial No.','University Rating','work experience','Family income','Age'],ax
```

In [45]:

```
df2.head()
```

Out[45]:

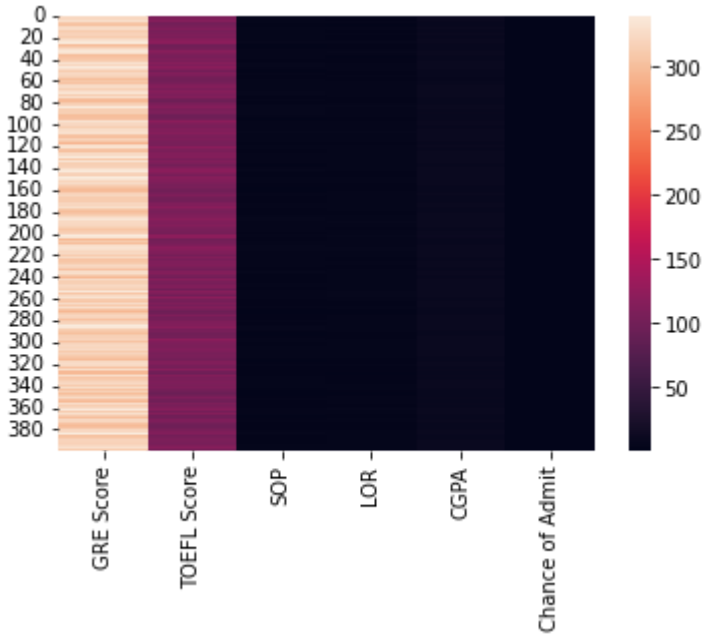
	GRE Score	TOEFL Score	SOP	LOR	CGPA	Chance of Admit
0	337	118	4.5	4.5	9.65	0.92
1	324	107	4.0	4.5	8.87	0.76
2	316	104	3.0	3.5	8.00	0.72
3	322	110	3.5	2.5	8.67	0.80
4	314	103	2.0	3.0	8.21	0.65

In [46]:

```
sns.heatmap(df2)
```

Out[46]:

<AxesSubplot:>



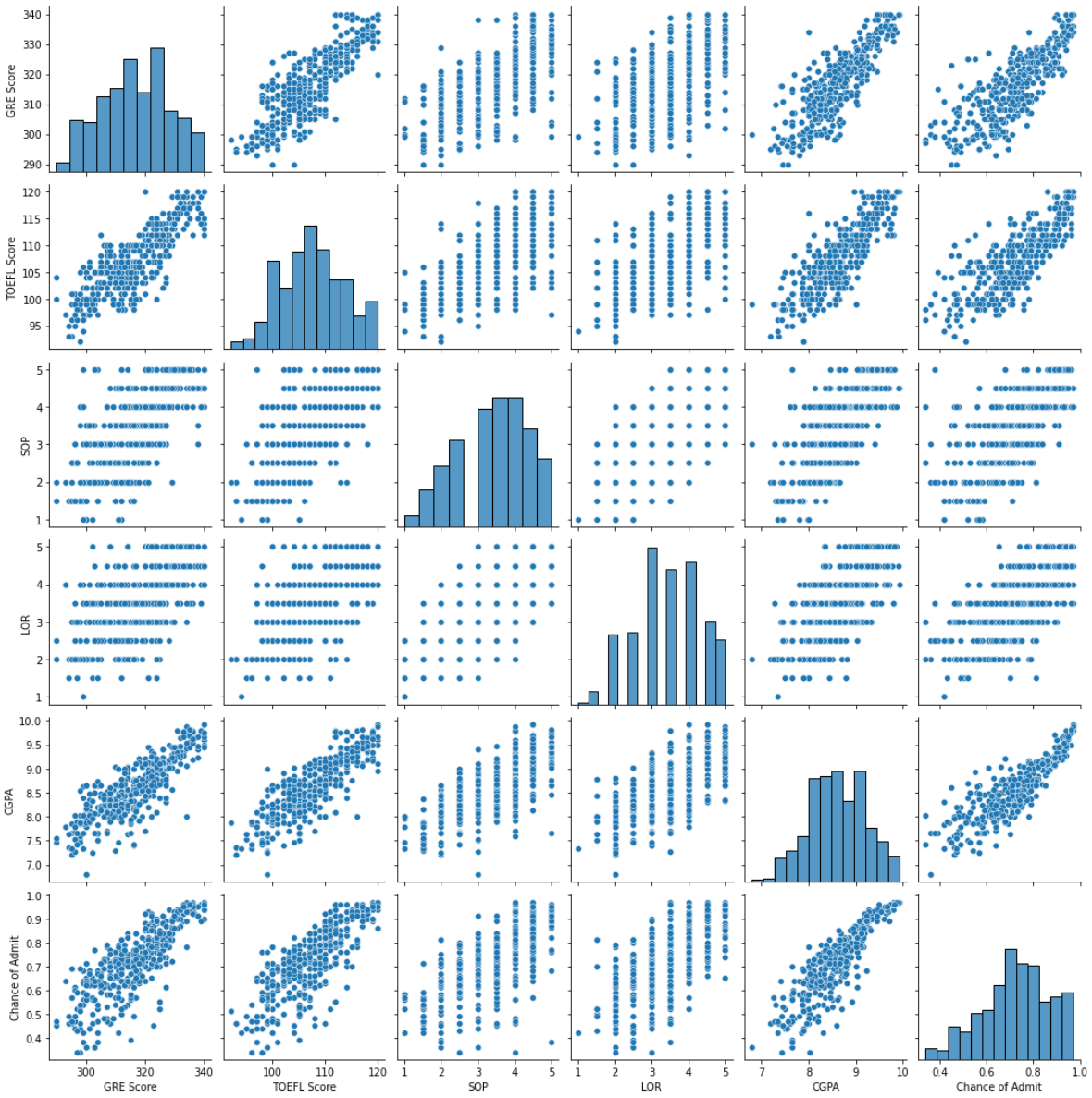
2) To predict the chance of admission, which type of models can be used? Give 3 examples.

In [47]:

```
sns.pairplot(df2)
```

Out[47]:

```
<seaborn.axisgrid.PairGrid at 0x17068d6aa40>
```



we can use Supervised ML and linear Regression model

reason 1 -> we have numeric data and input and output both are avalible

reason 2 -> i prefered linear regression because, Showing the graph of TOFEL score vs CGPA, it's continuous increasing

reason 3 -> Showing the graph of GRE score vs CGPA, it is also continuous increasing

3) Find null values.

In [35]:

```
df.isna().sum()
```

Out[35]:

```
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
work experience      1
Family income        0
Age                  0
Chance of Admit      0
dtype: int64
```

Summery:- There is a one null value present in "work experience" features

4) Find duplicate values (At Least 7 columns should have duplicate value) don't include predicted value

In [49]:

```
newdf = df.drop(['Chance of Admit'],axis=1)
newdf.head()
```

Out[49]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	work experience	Family income	Age
0	1	337	118	4	4.5	4.5	9.65	NaN	360000	18
1	2	324	107	4	4.0	4.5	8.87	1.0	320000	19
2	3	316	104	3	3.0	3.5	8.00	1.0	240000	20
3	4	322	110	3	3.5	2.5	8.67	1.0	280000	21
4	5	314	103	2	2.0	3.0	8.21	0.0	160000	22

In [54]:

```
newdf[newdf.duplicated()]
```

Out[54]:

Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	work experience	Family income	Age

Summery: There is No Duplicate records availble

5) Import the dataset and save into csv json and excel file in your local system (Please write the code only, don't need to upload the file).

In [56]:

```
df.to_csv()
df.to_json()
df.to_excel()
```

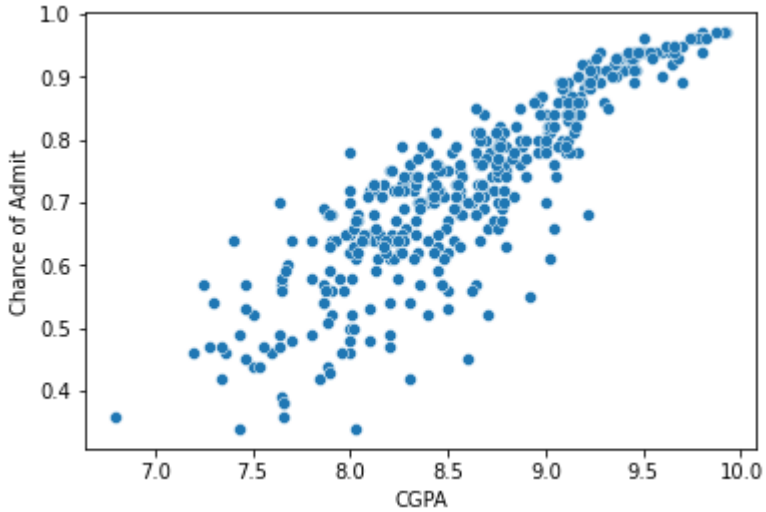
6) Which columns are highly co-relate?

In [36]:

```
sns.scatterplot(x=df['CGPA'],y=df['Chance of Admit'],data=df)
```

Out[36]:

```
<AxesSubplot:xlabel='CGPA', ylabel='Chance of Admit'>
```

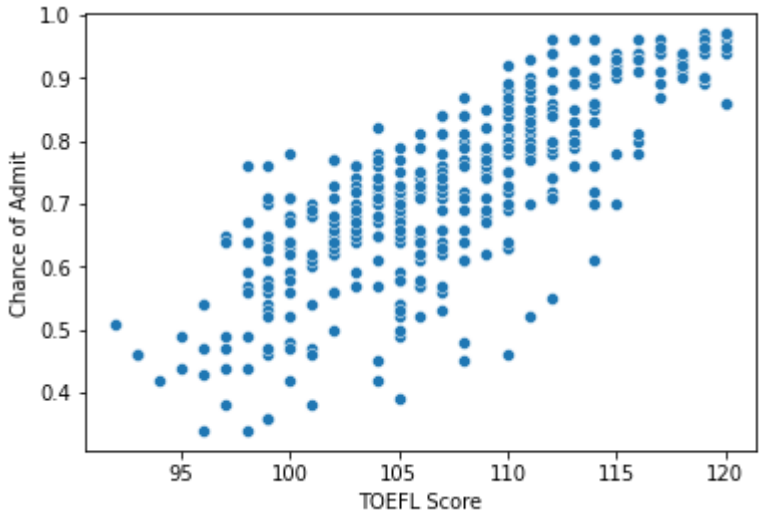


In [37]:

```
sns.scatterplot(x=df['TOEFL Score'],y=df['Chance of Admit'],data=df)
```

Out[37]:

```
<AxesSubplot:xlabel='TOEFL Score', ylabel='Chance of Admit'>
```

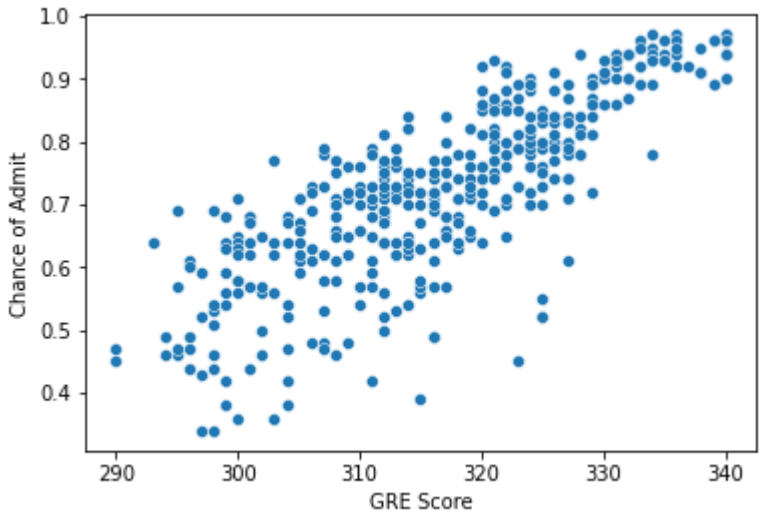


In [38]:

```
sns.scatterplot(x=df['GRE Score'],y=df['Chance of Admit'],data=df)
```

Out[38]:

```
<AxesSubplot:xlabel='GRE Score', ylabel='Chance of Admit'>
```



Summary:= GRE Score,TOEFEL score,CGPA are highly co-related

7) Find out the value of the intercept and coefficient model that you train.

In [60]:

```
X = df2.iloc[:,:5]
y = df2.iloc[:,-1]
```

In [64]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)
```

In [65]:

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()

model.fit(X_train,y_train)
```

Out[65]:

▾ LinearRegression

LinearRegression()

In [71]:

```
print("Coeficient for all features =" , model.coef_)
```

```
Coeficient for all features = [0.00227045 0.00346609 0.00292198 0.01553687
0.11909899]
```

In [69]:

```
print("Intercept =" , model.intercept_)
```

```
Intercept = -1.4510206434491648
```

In [ ]:

```

```