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

```
df = pd.read_csv("/content/Admission_Predict_Ver1.1.csv")
df.head()
```

	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

```
df.columns.str.replace('t ','t')
```

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

```
print("Data Information and Data Types")
df.info()
```

```
Data Information and Data Types
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         500 non-null    int64  
 1   GRE Score          500 non-null    int64  
 2   TOEFL Score        500 non-null    int64  
 3   University Rating  500 non-null    int64  
 4   SOP                500 non-null    float64
 5   LOR                500 non-null    float64
 6   CGPA               500 non-null    float64
 7   Research           500 non-null    int64  
 8   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 35.3 KB
```

```
print ("Missing Data (If Any)")
df.isnull().sum()
```

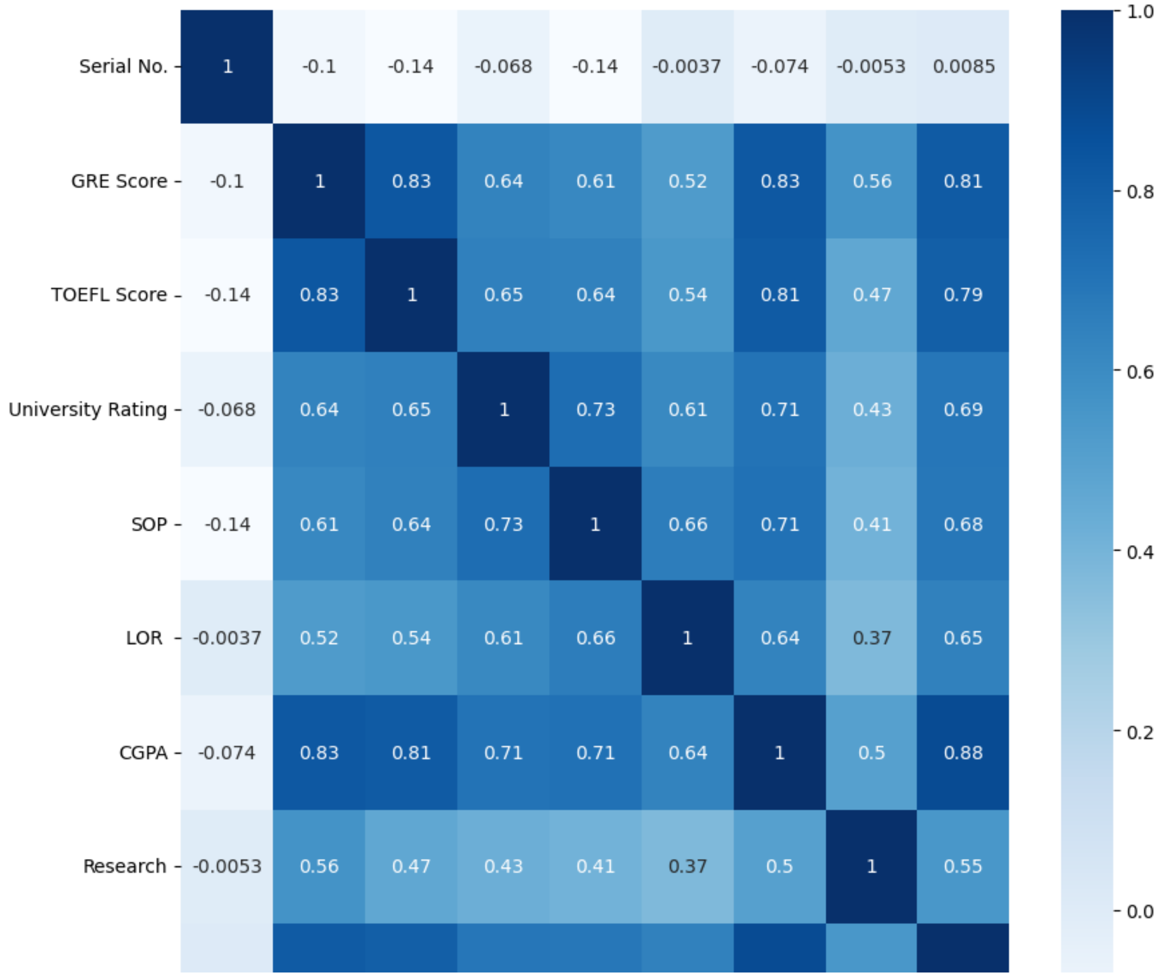
```
Missing Data (If Any)
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

```
df.corr()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
Serial No.	1.000000	-0.103839	-0.141696	-0.067641	-0.137352	-0.003694	-0.074289	-0.005332	0.008505
GRE Score	-0.103839	1.000000	0.827200	0.635376	0.613498	0.524679	0.825878	0.563398	0.810351
TOEFL Score	-0.141696	0.827200	1.000000	0.649799	0.644410	0.541563	0.810574	0.467012	0.792228
University Rating	-0.067641	0.635376	0.649799	1.000000	0.728024	0.608651	0.705254	0.427047	0.690132
SOP	-0.137352	0.613498	0.644410	0.728024	1.000000	0.663707	0.712154	0.408116	0.684137
LOR	-0.003694	0.524679	0.541563	0.608651	0.663707	1.000000	0.637469	0.372526	0.645365

```
plt.figure(figsize = (10,10))
sns.heatmap(df.corr(),annot=True,cmap='Blues')
```

```
<Axes: >
```

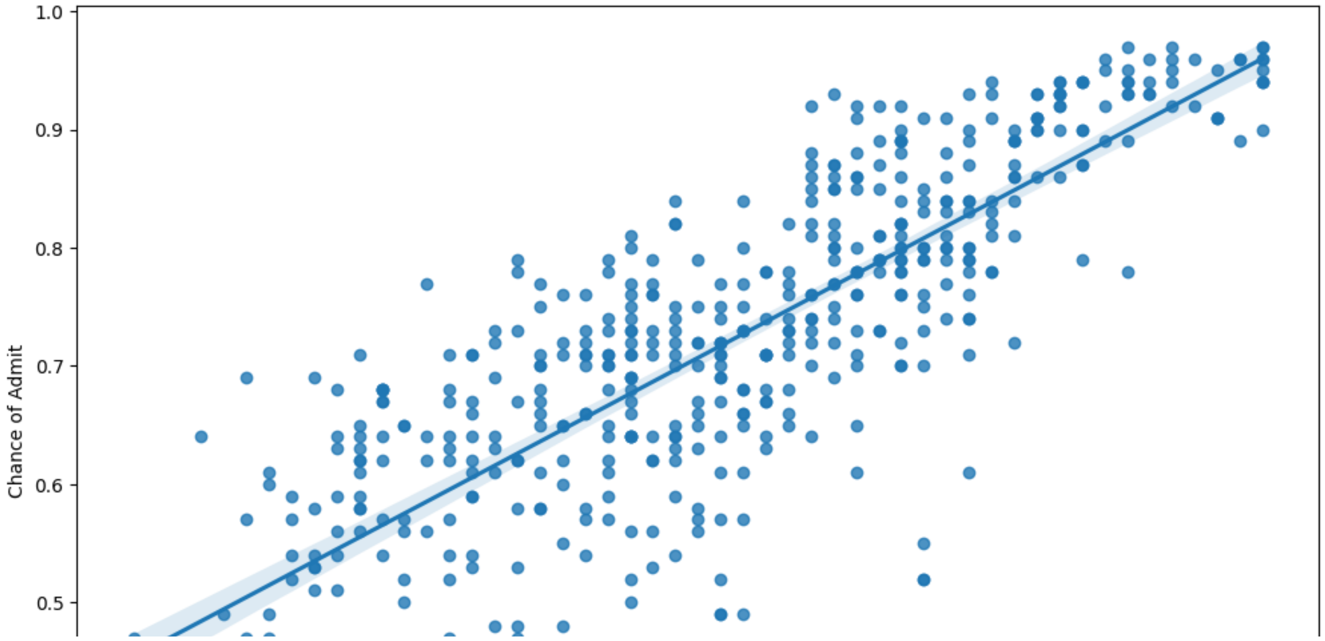


```
plt.subplots(figsize=(12,8))
plt.scatter(df["Chance of Admit "],df["GRE Score"])
plt.xlabel("Chance of Admit ")
plt.ylabel("GRE Score")
```

```
Text(0, 0.5, 'GRE Score')
```

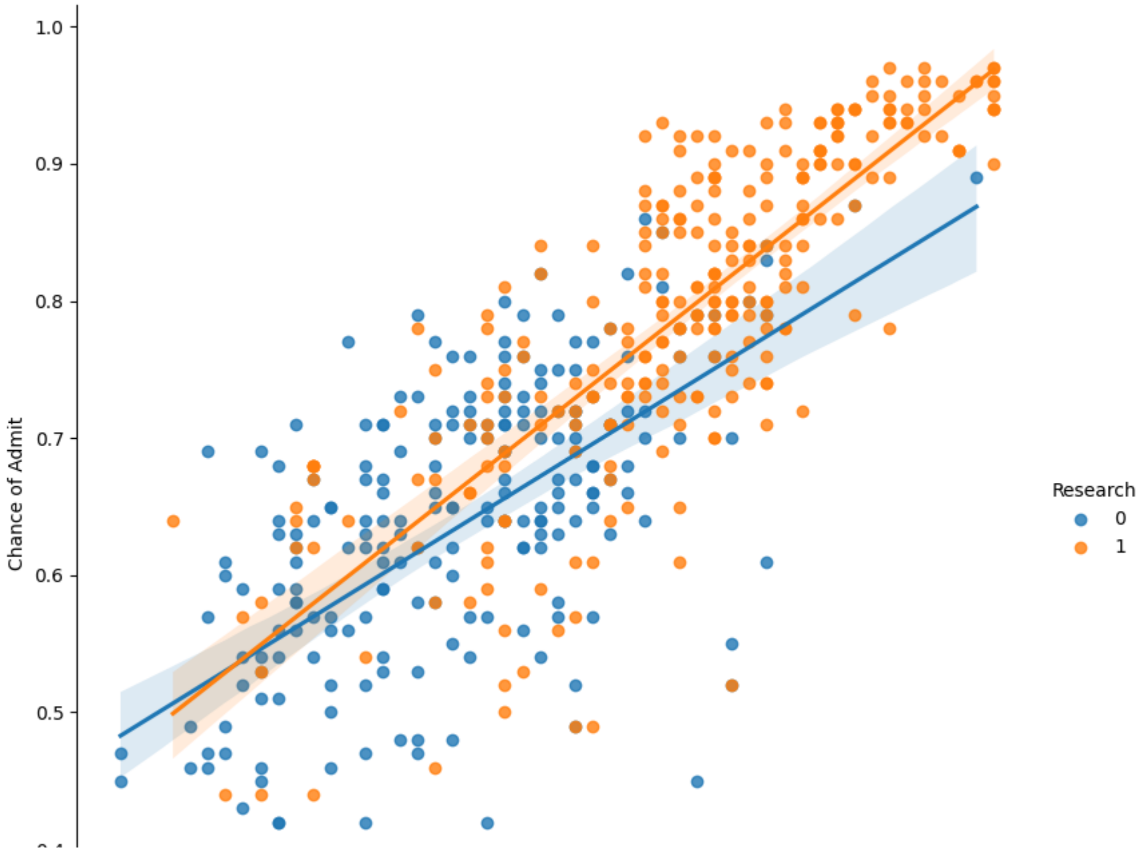
```
plt.subplots(figsize=(12,8))
sns.regplot(x="GRE Score",y="Chance of Admit ",data=df)
```

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



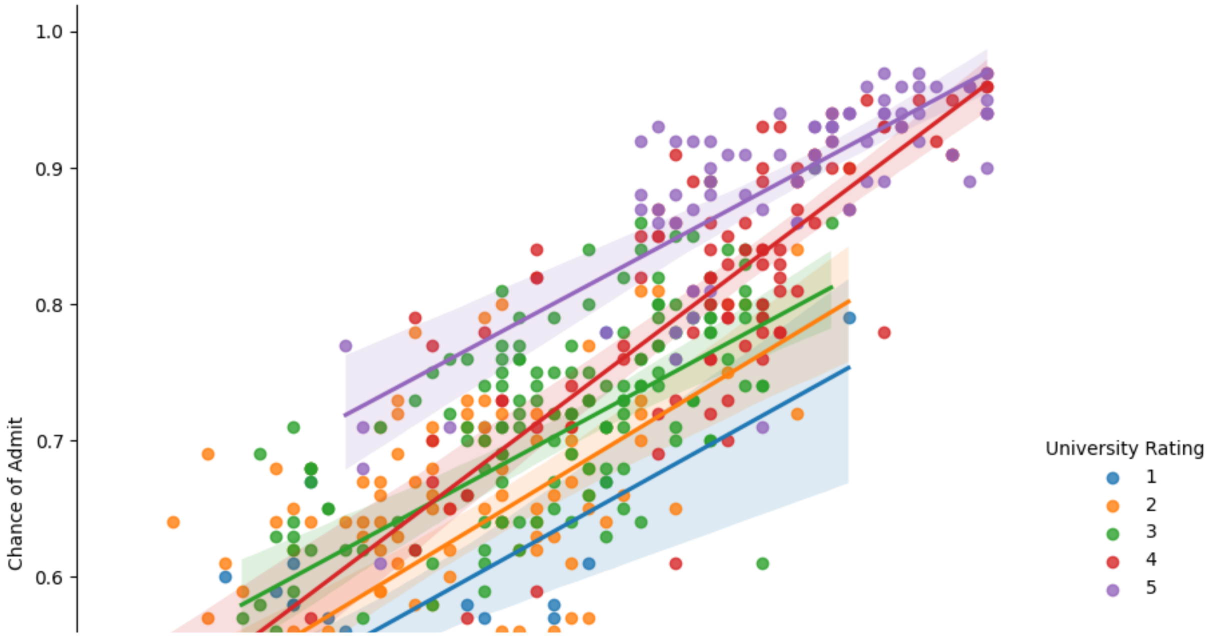
```
sns.lmplot(x="GRE Score",y="Chance of Admit ",data=df,hue="Research",height=8)
```

```
<seaborn.axisgrid.FacetGrid at 0x78004a41ad40>
```



```
sns.lmplot(x="GRE Score",y="Chance of Admit ",data=df,hue="University Rating",height=8)
```

```
<seaborn.axisgrid.FacetGrid at 0x780049f0e740>
```



```
admit_high_chance=df[df["Chance of Admit "]>=0.8]
admit_high_chance.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 155 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         155 non-null    int64  
 1   GRE Score          155 non-null    int64  
 2   TOEFL Score        155 non-null    int64  
 3   University Rating  155 non-null    int64  
 4   SOP                155 non-null    float64
 5   LOR                155 non-null    float64
 6   CGPA               155 non-null    float64
 7   Research           155 non-null    int64  
 8   Chance of Admit    155 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 12.1 KB
```

```
admit_high_chance.corr()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
Serial No.	1.000000	-0.172329	-0.173180	-0.106812	-0.080199	-0.008664	-0.131549	-0.030609	-0.145608
GRE Score	-0.172329	1.000000	0.697788	0.376891	0.382035	0.199562	0.685982	0.193577	0.708793
TOEFL Score	-0.173180	0.697788	1.000000	0.303186	0.377039	0.280364	0.629591	0.099272	0.685061
University Rating	-0.106812	0.376891	0.303186	1.000000	0.586152	0.505351	0.466308	0.234373	0.595083
SOP	-0.080199	0.382035	0.377039	0.586152	1.000000	0.529265	0.512077	0.178608	0.586424
LOR	-0.008664	0.199562	0.280364	0.505351	0.529265	1.000000	0.429269	0.049904	0.466544
CGPA	-0.131549	0.685982	0.629591	0.466308	0.512077	0.429269	1.000000	0.169378	0.856958
Research	-0.030609	0.193577	0.099272	0.234373	0.178608	0.049904	0.169378	1.000000	0.252328
Chance of Admit	-0.145608	0.708793	0.685061	0.595083	0.586424	0.466544	0.856958	0.252328	1.000000

```
plt.subplots(figsize=(12,8))
sns.set_theme(style="darkgrid")
sns.distplot(admit_high_chance["GRE Score"])
```

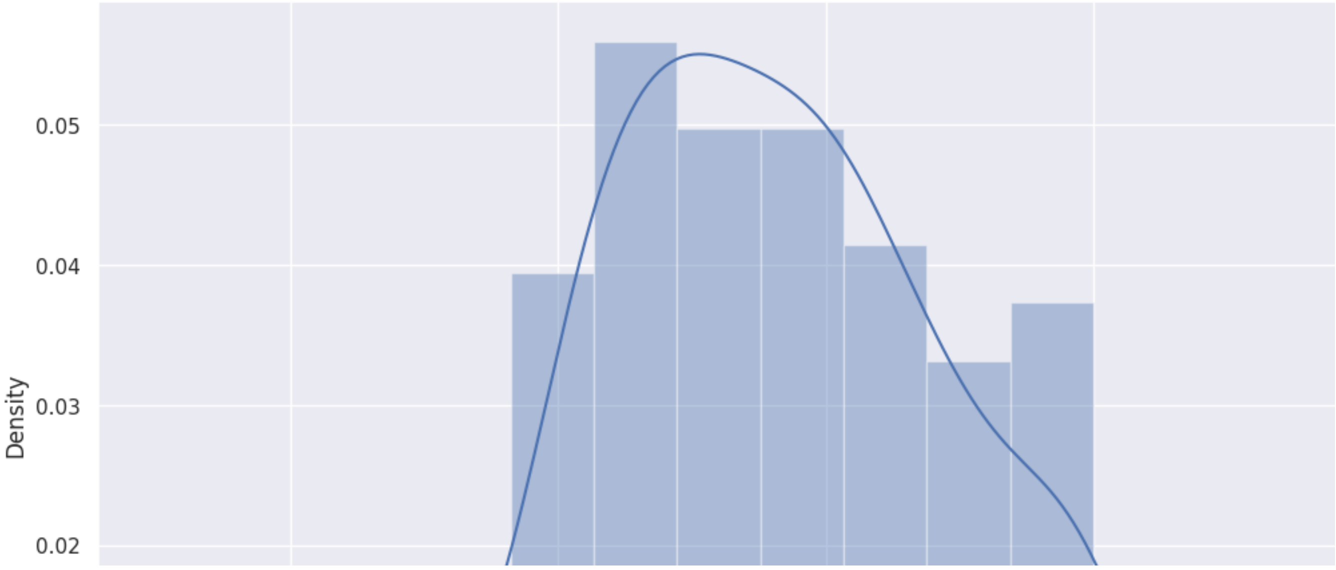
```
<ipython-input-64-bb92418fdf3e>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(admit_high_chance["GRE Score"])
<Axes: xlabel='GRE Score', ylabel='Density'>
```



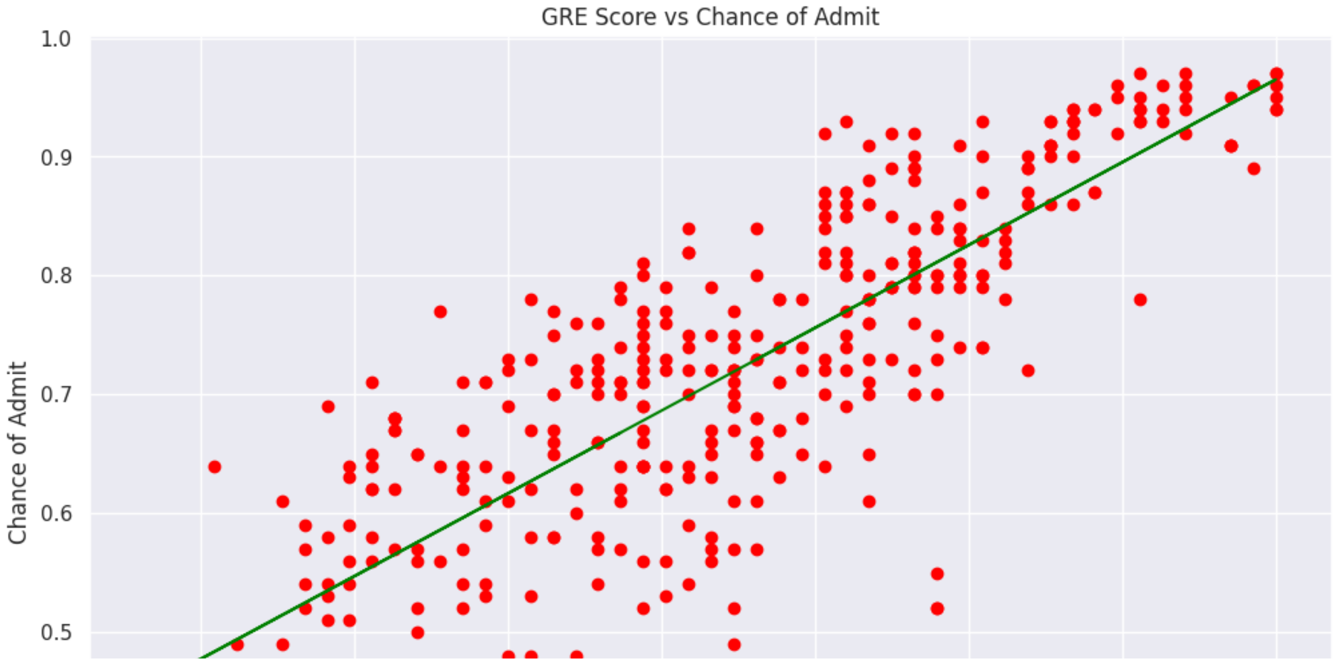
```
plt.subplots(figsize=(12,8))
sns.set_theme(style="darkgrid")
sns.distplot(admit_high_chance["Chance of Admit "])
```

```
X=df["GRE Score"].values
X=X/340
Y=df["Chance of Admit "].values
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.25)
```

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train.reshape(-1,1),Y_train)
Y_pred=lr.predict(X_test.reshape(-1,1))
lr.score(X_test.reshape(-1,1),Y_test.reshape(-1,1))
```

```
0.6250522303168087
```

```
plt.subplots(figsize=(12,8))
plt.scatter(X_train,Y_train,color="red")
plt.plot(X_train,lr.predict(X_train.reshape(-1,1)),color="green")
plt.title("GRE Score vs Chance of Admit")
plt.xlabel("GRE Score")
plt.ylabel("Chance of Admit ")
plt.show()
```



```
test=320
val=test/340
val_out=lr.predict(np.array([[val]]))
print("Chance of Admission: ",val_out[0])
```

```
Chance of Admission:  0.7600330771654371
```

```
x=df.drop(["Chance of Admit ","Serial No."],axis=1)
y=df["Chance of Admit "]
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.25,random_state=7)
from sklearn.ensemble import RandomForestRegressor
regr=RandomForestRegressor(max_depth=2,random_state=0,n_estimators=5)
regr.fit(X_train,Y_train)
regr.score(X_test,Y_test)
```

```
0.688275519532073
```

```
val=regr.predict([[325,100,3,4.1,3.7,7.67,1]])
print("Your Chances are (in %): ",val[0]*100)
```

```
Your Chances are (in %):  50.74660798031341
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegr
  warnings.warn(
```

● ×