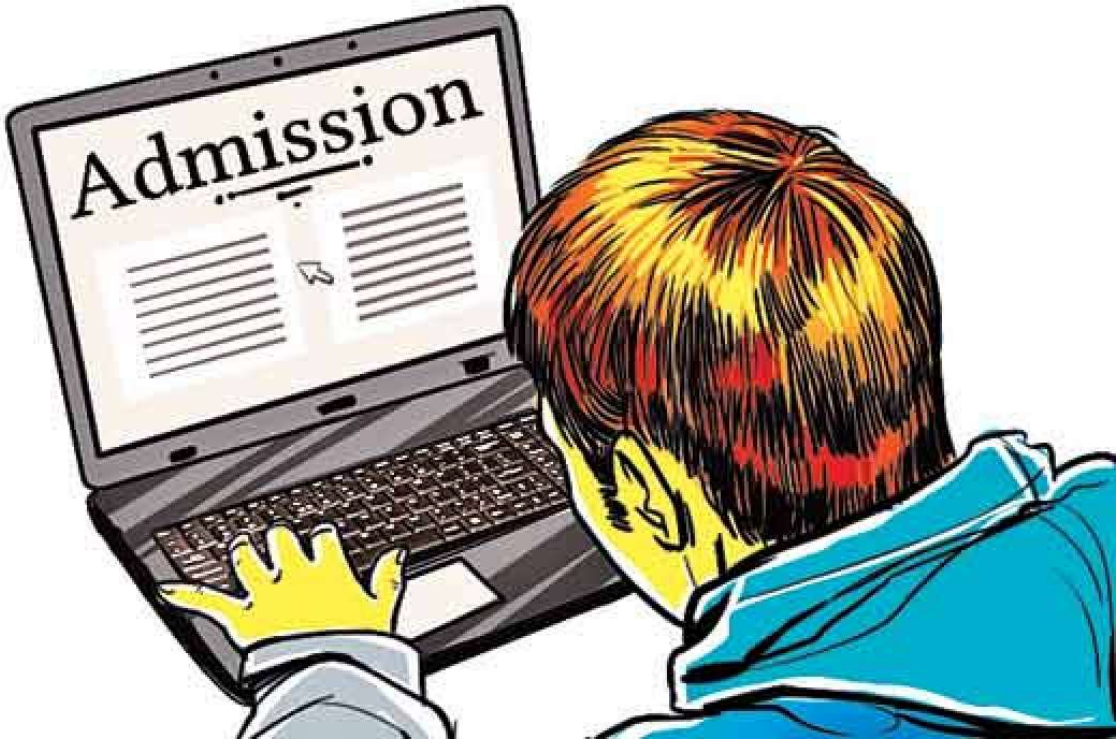
★ Regression Model to Predict Chance of Admission



Features in the dataset:

1. GRE Scores (290 to 340)
2. TOEFL Scores (92 to 120)
3. University Rating (1 to 5)
4. Statement of Purpose (1 to 5)
5. Letter of Recommendation Strength (1 to 5)
6. Undergraduate CGPA (6.8 to 9.92)
7. Research Experience (0 or 1)
8. Chance of Admit (0.34 to 0.97)

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

```
# import data
admission = pd.read_csv('https://github.com/ybifoundation/Dataset/raw/main/Admission%20Chance.csv
```

```
# view data
admission.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

```
# info of data
admission.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No          400 non-null    int64  
 1   GRE Score          400 non-null    int64  
 2   TOEFL Score        400 non-null    int64  
 3   University Rating  400 non-null    int64  
 4    SOP               400 non-null    float64
 5   LOR                400 non-null    float64
 6   CGPA               400 non-null    float64
 7   Research           400 non-null    int64  
 8   Chance of Admit    400 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 28.2 KB
```

```
# summary statistics
admission.describe()
```

	Serial No	GRE Score	TOEFL Score	University Rating	SOP	LOR	
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.00
mean	200.500000	316.807500	107.410000	3.087500	3.400000	3.452500	8.59
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.59
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.80
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.17
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.61
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.06

```
# check for missing value
admission.isna().sum()
```

```
Serial No           0
GRE Score           0
TOEFL Score         0
University Rating   0
 SOP                0
LOR                 0
CGPA                0
Research            0
```

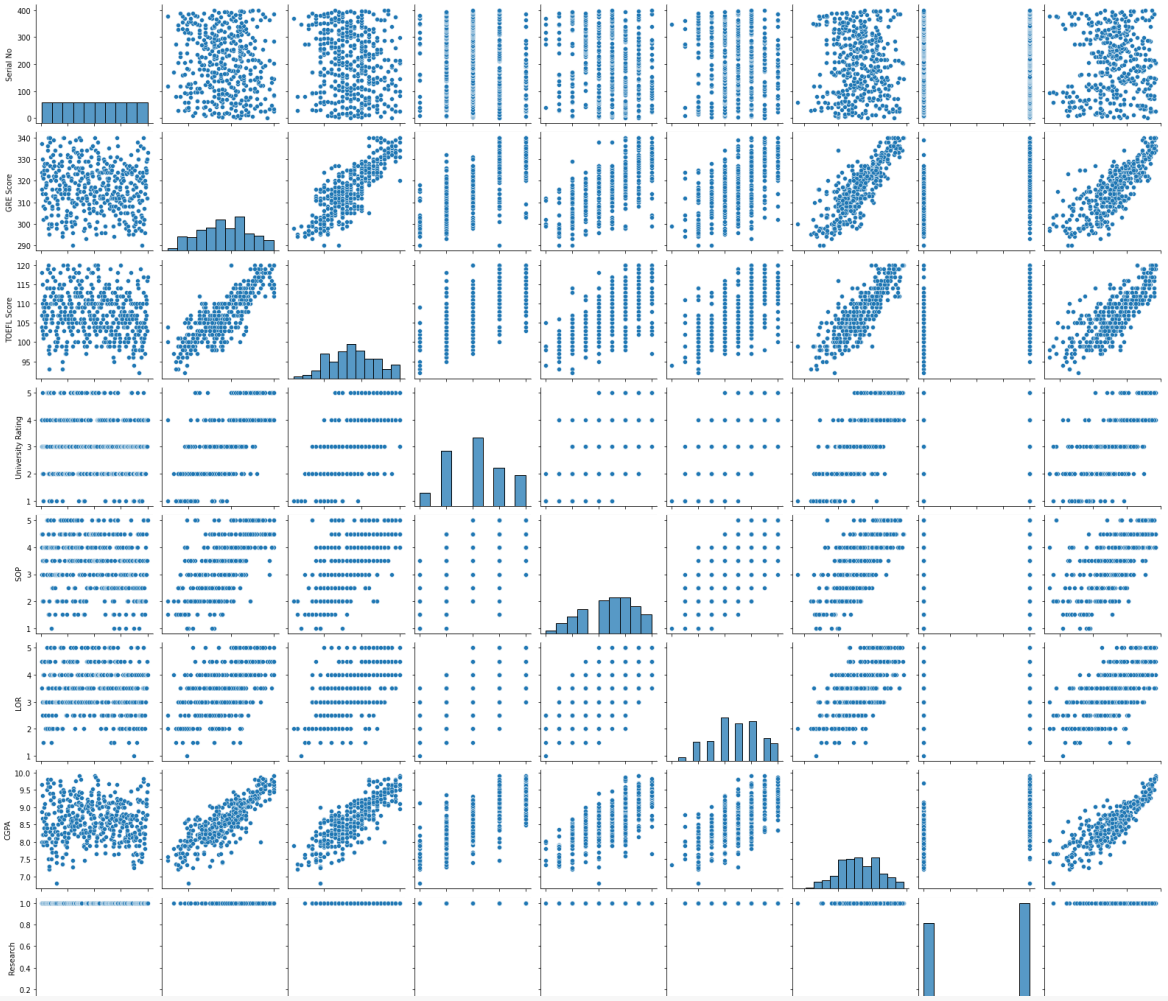
```
Chance of Admit      0
dtype: int64
```

```
# check for categories
admission.nunique()
```

```
Serial No           400
GRE Score            49
TOEFL Score          29
University Rating     5
 SOP                  9
LOR                   9
CGPA                168
Research              2
Chance of Admit      60
dtype: int64
```

```
# visualize pairplot
sns.pairplot(admission)
```

```
<seaborn.axisgrid.PairGrid at 0x7fcd8d25f350>
```




```
# columns name
admission.columns
```

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

```
# define y
y = admission['Chance of Admit ']
```

```
# define X
X = admission[['Serial No', 'GRE Score', 'TOEFL Score', 'University Rating', ' SOP',
       'LOR ', 'CGPA', 'Research']]
```

```
# split data
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=2529)
```

```
# verify shape
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((280, 8), (120, 8), (280,), (120,))
```

```
# select model
from sklearn.linear_model import LinearRegression
```

```
# train model
model = LinearRegression()
model.fit(X_train,y_train)
```

```
LinearRegression()
```

```
# predict with model
y_pred = model.predict(X_test)
```

```
# model evaluation
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, mean_squared_err
```

```
# model MAE
mean_absolute_error(y_test, y_pred)
```

```
0.04308714723355852
```

```
# model MAPE
mean_absolute_percentage_error(y_test, y_pred)
```

```
0.074004372761339
```

```
# model MSE
mean_squared_error(y_test, y_pred)
```

```
0.003922607781791493
```

```
# future prediction
sample = admission.sample()
sample
```

	Serial No	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
236	237	325	112	4	4.0	4.5	9.17	1	0.85

```
# define X_new
X_new = sample.loc[:,X.columns]
X_new
```

	Serial No	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
236	237	325	112	4	4.0	4.5	9.17	1

```
# predict for X_new
model.predict(X_new)
```

```
array([0.86245255])
```

✓ 0s completed at 12:32 PM