```
# This Python 3 environment comes with many helpful analytics
libraries installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing Shift+Enter)
will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/)
that gets preserved as output when you create a version using "Save &
Run All"
# You can also write temporary files to /kaggle/temp/, but they won't
be saved outside of the current session
```

```
/kaggle/input/datasets/rutujapandav/graduate-admission-dataset/
Admission_Predict.csv
```

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

import os

for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/datasets/rutujapandav/graduate-admission-dataset/
Admission_Predict.csv
```

```
admission = pd.read_csv("/kaggle/input/datasets/rutujapandav/graduate-
admission-dataset/Admission_Predict.csv")
admission.head()
```

```
   Serial No  GRE Score  TOEFL Score  University Rating   SOP  LOR
CGPA  \
0          1        337          118                  4   4.5   4.5
9.65
1          2        324          107                  4   4.0   4.5
```

```
8.87
2          3        316          104                   3   3.0   3.5
8.00
3          4        322          110                   3   3.5   2.5
8.67
4          5        314          103                   2   2.0   3.0
8.21

   Research  Chance of Admit
0         1              0.92
1         1              0.76
2         1              0.72
3         1              0.80
4         0              0.65
```

```
admission.shape
admission.info()
admission.describe()
```

```
<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: 28.3 KB
```

```
        Serial No   GRE Score  TOEFL Score  University Rating
SOP  \
count  400.000000  400.000000   400.000000         400.000000
400.000000
mean   200.500000  316.807500   107.410000           3.087500
3.400000
std    115.614301   11.473646     6.069514           1.143728
1.006869
min      1.000000  290.000000    92.000000           1.000000
1.000000
25%    100.750000  308.000000   103.000000           2.000000
2.500000
50%    200.500000  317.000000   107.000000           3.000000
3.500000
```

```
75%    300.250000  325.000000   112.000000           4.000000
4.000000
max    400.000000  340.000000   120.000000           5.000000
5.000000

             LOR         CGPA    Research  Chance of Admit
count  400.000000  400.000000  400.000000        400.000000
mean     3.452500    8.598925    0.547500          0.724350
std      0.898478    0.596317    0.498362          0.142609
min      1.000000    6.800000    0.000000          0.340000
25%      3.000000    8.170000    0.000000          0.640000
50%      3.500000    8.610000    1.000000          0.730000
75%      4.000000    9.062500    1.000000          0.830000
max      5.000000    9.920000    1.000000          0.970000
```

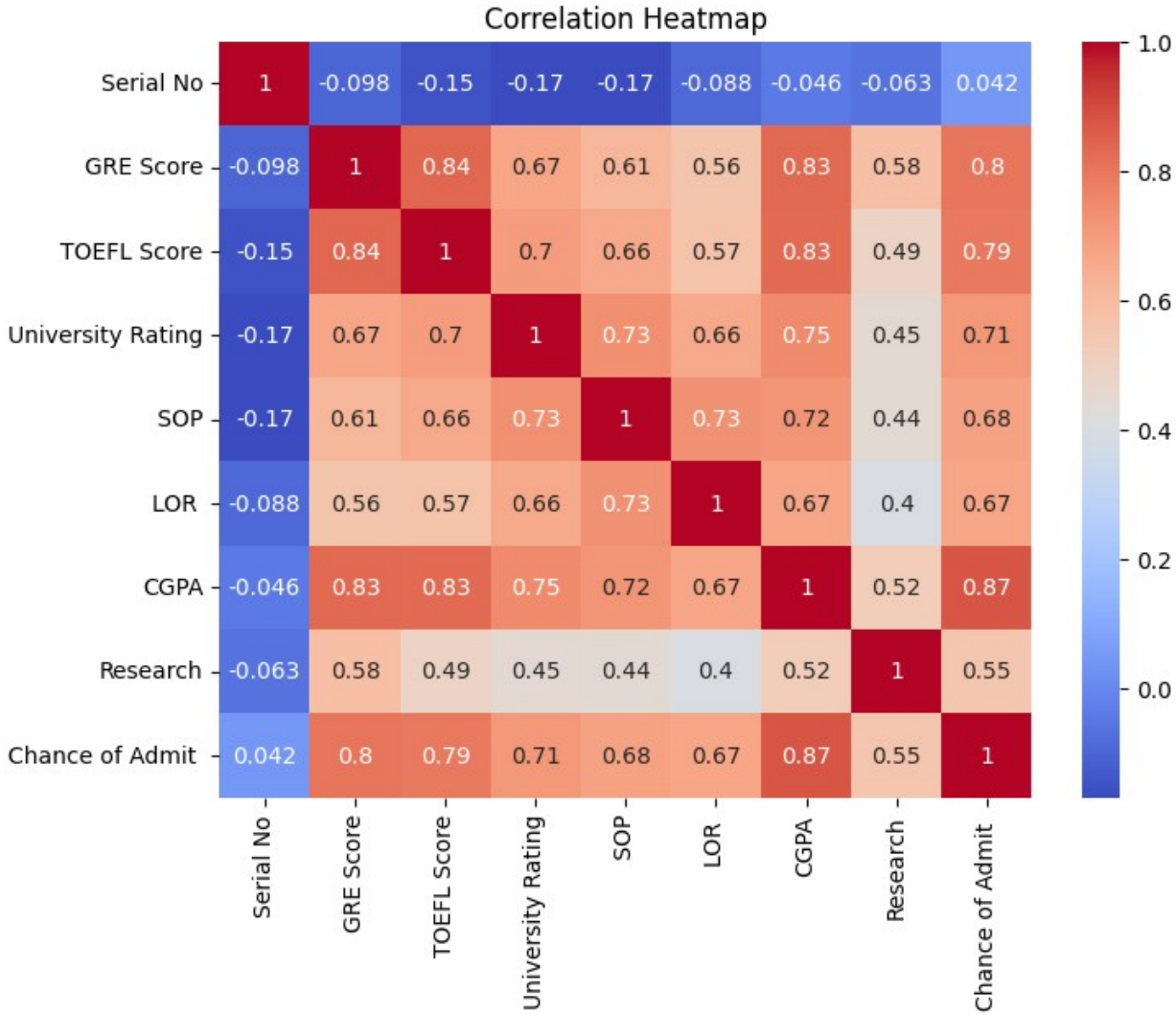
```
admission.columns
```

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

```
X = admission.drop("Chance of Admit ", axis=1)
y = admission["Chance of Admit "]

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=2529
)

from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(X_train, y_train)

y_pred = lr.predict(X_test)

from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_absolute_error

rf = RandomForestRegressor(random_state=2529)
rf.fit(X_train, y_train)

rf_pred = rf.predict(X_test)

print("Linear Regression R2:", r2_score(y_test, y_pred))
print("Random Forest R2:", r2_score(y_test, rf_pred))
print("Random Forest MAE:", mean_absolute_error(y_test, rf_pred))
```

```
Linear Regression R2: 0.812948382648416
Random Forest R2: 0.8655385576975195
Random Forest MAE: 0.04110250000000003
```

```
plt.figure(figsize=(8,6))
sns.heatmap(admission.corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```




```
importance = pd.DataFrame({
    "Feature": X.columns,
    "Importance": rf.feature_importances_
})

importance.sort_values(by="Importance", ascending=False)
```

```
              Feature  Importance
6                CGPA    0.768517
0           Serial No    0.070975
1           GRE Score    0.067159
2         TOEFL Score    0.035885
4                 SOP    0.019993
5                LOR     0.018013
3  University Rating    0.010036
7            Research    0.009422
```

```
admission.shape
```

```
(400, 9)
```

Conclusion**** CGPA is the most important factor affecting admission chances. Research experience positively impacts admission probability. Linear Regression provides good performance. Random Forest improves prediction accuracy further. The model can be used to estimate admission probability based on student profile.