Predicting graduate admission

Random Forest Regressor

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv("Admission_Predict.csv")
df.head()
```

Out[2]:

	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

In [3]:
```
x=df.iloc[:,1:-1].values
y=df.iloc[:,-1].values
print(x)
```

```
[[337.   118.     4.   ...   4.5    9.65   1.  ]
 [324.   107.     4.   ...   4.5    8.87   1.  ]
 [316.   104.     3.   ...   3.5    8.     1.  ]
 ...
 [330.   116.     4.   ...   4.5    9.45   1.  ]
 [312.   103.     3.   ...   4.     8.78   0.  ]
 [333.   117.     4.   ...   4.     9.66   1.  ]]
```

In [4]:
```
print(y)
```

```
[0.92 0.76 0.72 0.8  0.65 0.9  0.75 0.68 0.5  0.45 0.52 0.84 0.78 0.62
 0.61 0.54 0.66 0.65 0.63 0.62 0.64 0.7  0.94 0.95 0.97 0.94 0.76 0.44
 0.46 0.54 0.65 0.74 0.91 0.9  0.94 0.88 0.64 0.58 0.52 0.48 0.46 0.49
 0.53 0.87 0.91 0.88 0.86 0.89 0.82 0.78 0.76 0.56 0.78 0.72 0.7  0.64
 0.64 0.46 0.36 0.42 0.48 0.47 0.54 0.56 0.52 0.55 0.61 0.57 0.68 0.78
 0.94 0.96 0.93 0.84 0.74 0.72 0.74 0.64 0.44 0.46 0.5  0.96 0.92 0.92
 0.94 0.76 0.72 0.66 0.64 0.74 0.64 0.38 0.34 0.44 0.36 0.42 0.48 0.86
 0.9  0.79 0.71 0.64 0.62 0.57 0.74 0.69 0.87 0.91 0.93 0.68 0.61 0.69
 0.62 0.72 0.59 0.66 0.56 0.45 0.47 0.71 0.94 0.94 0.57 0.61 0.57 0.64
 0.85 0.78 0.84 0.92 0.96 0.77 0.71 0.79 0.89 0.82 0.76 0.71 0.8  0.78
 0.84 0.9  0.92 0.97 0.8  0.81 0.75 0.83 0.96 0.79 0.93 0.94 0.86 0.79
 0.8  0.77 0.7  0.65 0.61 0.52 0.57 0.53 0.67 0.68 0.81 0.78 0.65 0.64
 0.64 0.65 0.68 0.89 0.86 0.89 0.87 0.85 0.9  0.82 0.72 0.73 0.71 0.71
 0.68 0.75 0.72 0.89 0.84 0.93 0.93 0.88 0.9  0.87 0.86 0.94 0.77 0.78
 0.73 0.73 0.7  0.72 0.73 0.72 0.97 0.97 0.69 0.57 0.63 0.66 0.64 0.68
 0.79 0.82 0.95 0.96 0.94 0.93 0.91 0.85 0.84 0.74 0.76 0.75 0.76 0.71
 0.67 0.61 0.63 0.64 0.71 0.82 0.73 0.74 0.69 0.64 0.91 0.88 0.85 0.86
 0.7  0.59 0.6  0.65 0.7  0.76 0.63 0.81 0.72 0.71 0.8  0.77 0.74 0.7
 0.71 0.93 0.85 0.79 0.76 0.78 0.77 0.9  0.87 0.71 0.7  0.7  0.75 0.71
```

```
0.72 0.73 0.83 0.77 0.72 0.54 0.49 0.52 0.58 0.78 0.89 0.7  0.66 0.67
0.68 0.8  0.81 0.8  0.94 0.93 0.92 0.89 0.82 0.79 0.58 0.56 0.56 0.64
0.61 0.68 0.76 0.86 0.9  0.71 0.62 0.66 0.65 0.73 0.62 0.74 0.79 0.8
0.69 0.7  0.76 0.84 0.78 0.67 0.66 0.65 0.54 0.58 0.79 0.8  0.75 0.73
0.72 0.62 0.67 0.81 0.63 0.69 0.8  0.43 0.8  0.73 0.75 0.71 0.73 0.83
0.72 0.94 0.81 0.81 0.75 0.79 0.58 0.59 0.47 0.49 0.47 0.42 0.57 0.62
0.74 0.73 0.64 0.63 0.59 0.73 0.79 0.68 0.7  0.81 0.85 0.93 0.91 0.69
0.77 0.86 0.74 0.57 0.51 0.67 0.72 0.89 0.95 0.79 0.39 0.38 0.34 0.47
0.56 0.71 0.78 0.73 0.82 0.62 0.96 0.96 0.46 0.53 0.49 0.76 0.64 0.71
0.84 0.77 0.89 0.82 0.84 0.91 0.67 0.95]
```

In [5]:
```
df.describe()
```

Out[5]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Rese
count	400.000000	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	8.598925	0.54
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.49
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.00
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.00
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.00
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.00
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.00

Exploratory Analysis

From these charts it looks like we have no missing values!

It seems as though Serial No. is just an index for students, which we can take out.

Two columns also have an added space in the label which we'll take out

We are also removing the blank sapces.

In [6]:
```
print(df.shape)
```

```
(400, 9)
```

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

Out[7]:
```
<bound method DataFrame.info of      Serial No.  GRE Score  TOEFL Score  Un
iversity 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
..            ...        ...          ...                ...  ...   ...   ...
395           396        324          110                  3  3.5   3.5  9.04
396           397        325          107                  3  3.0   3.5  9.11
397           398        330          116                  4  5.0   4.5  9.45
398           399        312          103                  3  3.5   4.0  8.78
399           400        333          117                  4  5.0   4.0  9.66
```

```
     Research  Chance of Admit
0           1              0.92
1           1              0.76
2           1              0.72
3           1              0.80
4           0              0.65
..        ...               ...
395         1              0.82
396         1              0.84
397         1              0.91
398         0              0.67
399         1              0.95

[400 rows x 2 columns]
```

In [8]:
```
# to directly know the number of missing values in a data frame
df.isna().sum()
```

Out[8]:
```
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
```

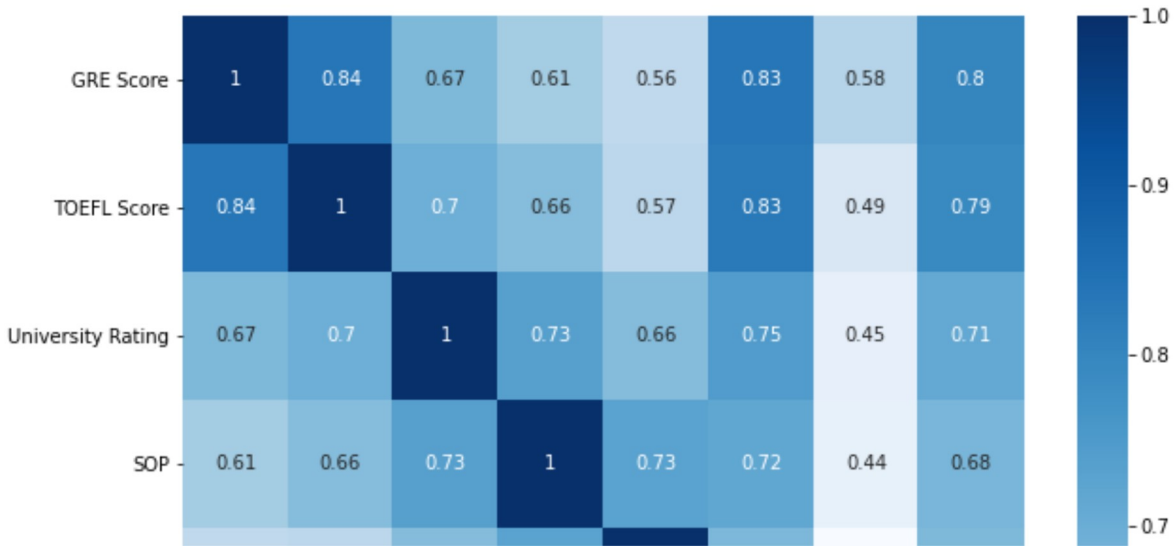
In [9]:
```
df.rename(columns = {'Chance of Admit ':'Chance of Admit', 'LOR ':'LOR'}, i
df.drop(labels='Serial No.', axis=1, inplace=True)
```

Let's plot a heatmap to see the correlation of all the features compared to Chance to Admit:

EDA

In [10]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(df.corr(), annot=True, cmap='Blues')
```

Out[10]:
```
<AxesSubplot:>
```



Preparing Data for Machine Learning

Now that we understand our dataset, it's time to implement machine learning methods to predict future applicant's chances of admission.

First we have to prepare our data, by splitting it into training and testing data. We'll also scale our data, from 0 to 1, to receive more accurate predictions.

In [11]:
```
from sklearn.model_selection import train_test_split
targets = df['Chance of Admit']
features = df.drop(columns = {'Chance of Admit'})

X_train, X_test, y_train, y_test = train_test_split(features, targets, test
```

In [12]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

Random Forest

In [13]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import accuracy_score,confusion_matrix
forest = RandomForestRegressor(n_estimators=110,max_depth=6,random_state=0)
forest.fit(X_train, y_train)
y_predict = forest.predict(X_test)
forest_score = forest.score(X_test, y_test)*100
print(forest_score)
```

```
81.35681504113325
```

In [ ]:

In [ ]: