ADMISSION PREDICTION BY USING MACHINE LEARNING

```
In [29]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sns
         import random
```

```
In [7]: df=pd.read_csv("Admission_Predict_ver1.1.csv")
```

```
In [8]: df.columns
```

```
Out[8]: Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating',
        'SOP',
               'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
              dtype='object')
```

```
In [4]: #df.head(2)
        df.columns
```

```
Out[4]: Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating',
        'SOP',
               'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
              dtype='object')
```

```
In [10]: len(df.columns)
```

```
Out[10]: 9
```

Dropping unwanted columns (Example:Art No ,

Correspondence Address , volume)

```
In [6]: df.drop(["Chance of Admit "],axis=1,inplace=True)
```

```
In [7]: df.columns #Art. No.","Correspondence Address have been dropped from th
        e original dataframe since inplace = ture in above step
```

```
Out[7]: Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating',
        'SOP',
               'LOR ', 'CGPA', 'Research'],
              dtype='object')
```

```
In [8]: len(df.columns) # length has been reduced by 2 since 2 columns has been
         dropped form the dataset
```

```
Out[8]: 8
```

Check if there are fields contianing null values

```
In [9]: df.isnull().any()
```

```
Out[9]: Serial No.           False
        GRE Score            False
        TOEFL Score          False
        University Rating    False
        SOP                  False
        LOR                  False
        CGPA                 False
        Research             False
        dtype: bool
```

```
In [10]: df.describe(include='all')
```

```
Out[10]:
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Res
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.0
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.5
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.4
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.0
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.0
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.0
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.0
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.0

Renaming columns

In [11]:

```
df.rename(columns={"Serial No.":"Serial_No","GRE Score":"GRE_Score","TO
EFL Score":"TOEFL_Score","University Rating":"University_Rating","Chanc
e of Admit ":"Chance_of_Admit"},inplace=True)
```

In [12]:

```
df.columns
```

Out[12]:

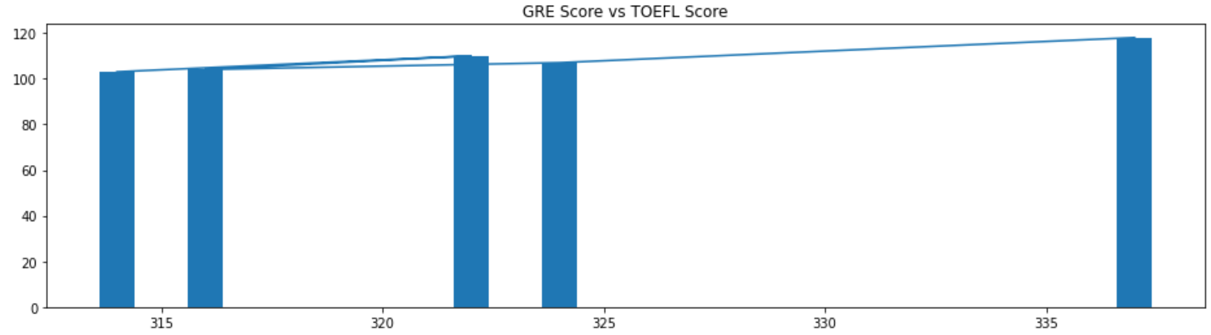
```
Index(['Serial_No', 'GRE_Score', 'TOEFL_Score', 'University_Rating', 'S
OP',
       'LOR ', 'CGPA', 'Research'],
      dtype='object')
```

Comparison between GRE Score and TOEFL Score

In [13]:

```
x = df.GRE_Score
y = df.TOEFL_Score
```

```
plt.figure(figsize=(16,4))
plt.bar(x[:5],y[:5])
plt.plot(x[:5],y[:5])
plt.title('GRE Score vs TOEFL Score')
plt.savefig("output.jpg")
```



In [14]:

```
df.isnull().any()
```

Out[14]:

```
Serial_No            False
GRE_Score            False
TOEFL_Score          False
University_Rating    False
SOP                  False
LOR                  False
CGPA                 False
Research             False
dtype: bool
```

Number of Students who have scored more than 300

In [15]:

```
df1 = df.where(df.GRE_Score>300)
df1.Serial_No.count()
```

Out[15]:

```
447
```

Minimum GRE Score and TOEFL Score to get admission in 5 rating university

In [16]:

```
df1 = df.where(df.University_Rating==5)
print("Min GRE Score : ",df1.GRE_Score.min())

df1 = df.where(df.University_Rating==5)
print("Min TOEFL Score : ",df1.TOEFL_Score.min())
```

```
Min GRE Score :  303.0
Min TOEFL Score :  101.0
```

Minimum , Maximum , Median, Mean of the GRE and TOEFL Scores

In [17]:

```
print("Minimum GRE Score : ",df.GRE_Score.min())
print("Maximum GRE Score : ",df.GRE_Score.max())
print("Median GRE Score : ",df.GRE_Score.median())
print("Mean GRE Score : ",df.GRE_Score.mean())
print("*******************************")
print("Minimum TOEFL Score : ",df.TOEFL_Score.min())
print("Maximum TOEFL Score : ",df.TOEFL_Score.max())
print("Median TOEFL Score : ",df.TOEFL_Score.median())
print("Mean TOEFL Score : ",df.TOEFL_Score.mean())
```

```
Minimum GRE Score :  290
Maximum GRE Score :  340
Median GRE Score :  317.0
Mean GRE Score :  316.472
*******************************
Minimum TOEFL Score :  92
Maximum TOEFL Score :  120
Median TOEFL Score :  107.0
Mean TOEFL Score :  107.192
```

In [15]:
```
df = df.drop('Serial No.',axis = 1)
```

In [16]:
```
df.head()
```

Out[16]:

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

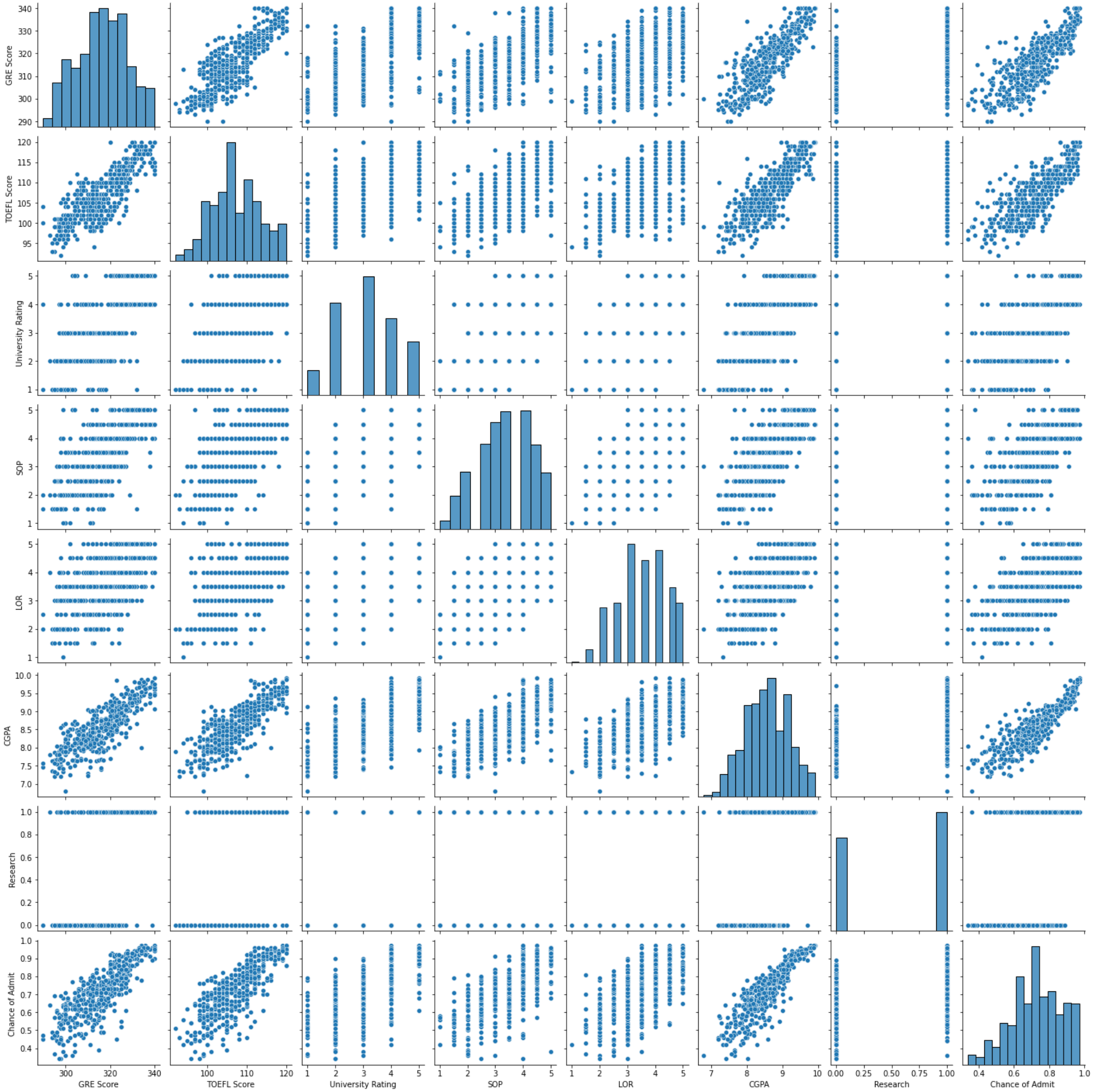
In [17]:
```
df.describe()
```

Out[17]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Cha of A
count	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.00
mean	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.72
std	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.14
min	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.34
25%	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.63
50%	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.72
75%	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.82
max	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.97

In [18]:
```
sns.pairplot(df)
```

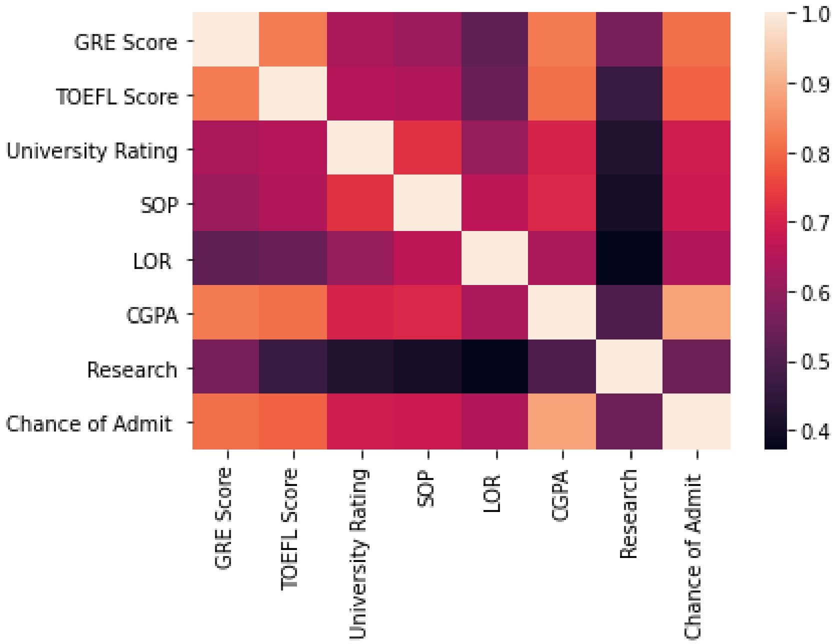
Out[18]:
```
<seaborn.axisgrid.PairGrid at 0x278a00ae9a0>
```



In [19]:

```
corr = df.corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
```

Out[19]: <AxesSubplot:>



In [20]:
```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV,train_test_split
from sklearn.metrics import mean_absolute_error
```

In [22]:
```
X = df.drop('Chance of Admit ',axis = 1)
y = df['Chance of Admit ']

X_train,X_val,y_train,y_val = train_test_split(X,y,test_size = .25,rand
om_state = 123)
```

In [23]:
```
lin_model = LinearRegression()
```

In [24]:
```
lin_model.fit(X_train,y_train)
```

Out[24]:
```
LinearRegression()
```

In [25]:
```
print('Mean absolute error for linear model: %0.4f' %mean_absolute_erro
r(y_val,lin_model.predict(X_val)))
```

```
Mean absolute error for linear model: 0.0423
```

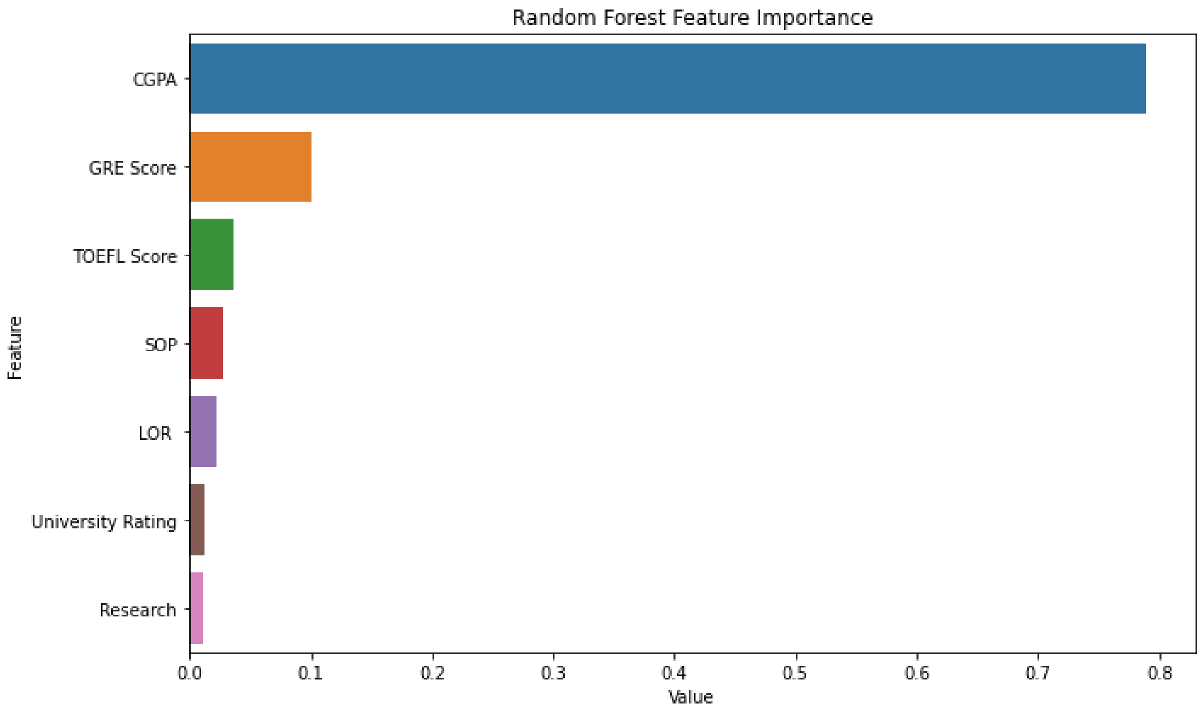
In [26]:
```
rf_model = RandomForestRegressor(n_estimators = 100,random_state = 42)
rf_model.fit(X_train,y_train)
```

Out[26]:
```
RandomForestRegressor(random_state=42)
```

In [27]:
```
print('Mean absolute error for linear model: %0.4f' %mean_absolute_erro
r(y_val,rf_model.predict(X_val)))
```

```
Mean absolute error for linear model: 0.0419
```

In [30]:
```
feature_importance = pd.DataFrame(sorted(zip(rf_model.feature_importanc
es_, X.columns)), columns=['Value','Feature'])
plt.figure(figsize=(10, 6))
sns.barplot(x="Value", y="Feature", data=feature_importance.sort_values
(by="Value", ascending=False))
plt.title('Random Forest Feature Importance')
plt.tight_layout()
```


Random Forest Feature Importance
Feature
CGPA
GRE Score
TOEFL Score
SOP
LOR
University Rating
Research
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Value