In [1]:

```
import pandas as pd
import random
import seaborn as sns
import matplotlib.pyplot as plt

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
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LogisticRegression
```

In [2]:

```
adm=pd.read_csv(r"C:\Users\HP\Documents\Admission_Predict_Ver1.1.csv")
```

In [3]:

```
adm.head()
```

Out[3]:

| | 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 [4]:

```
adm= adm.drop('Serial No.',axis = 1)
```

In [5]:

```
adm.head()
```

Out[5]:

| | 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 [6]:

```
adm.describe()
```

Out[6]:

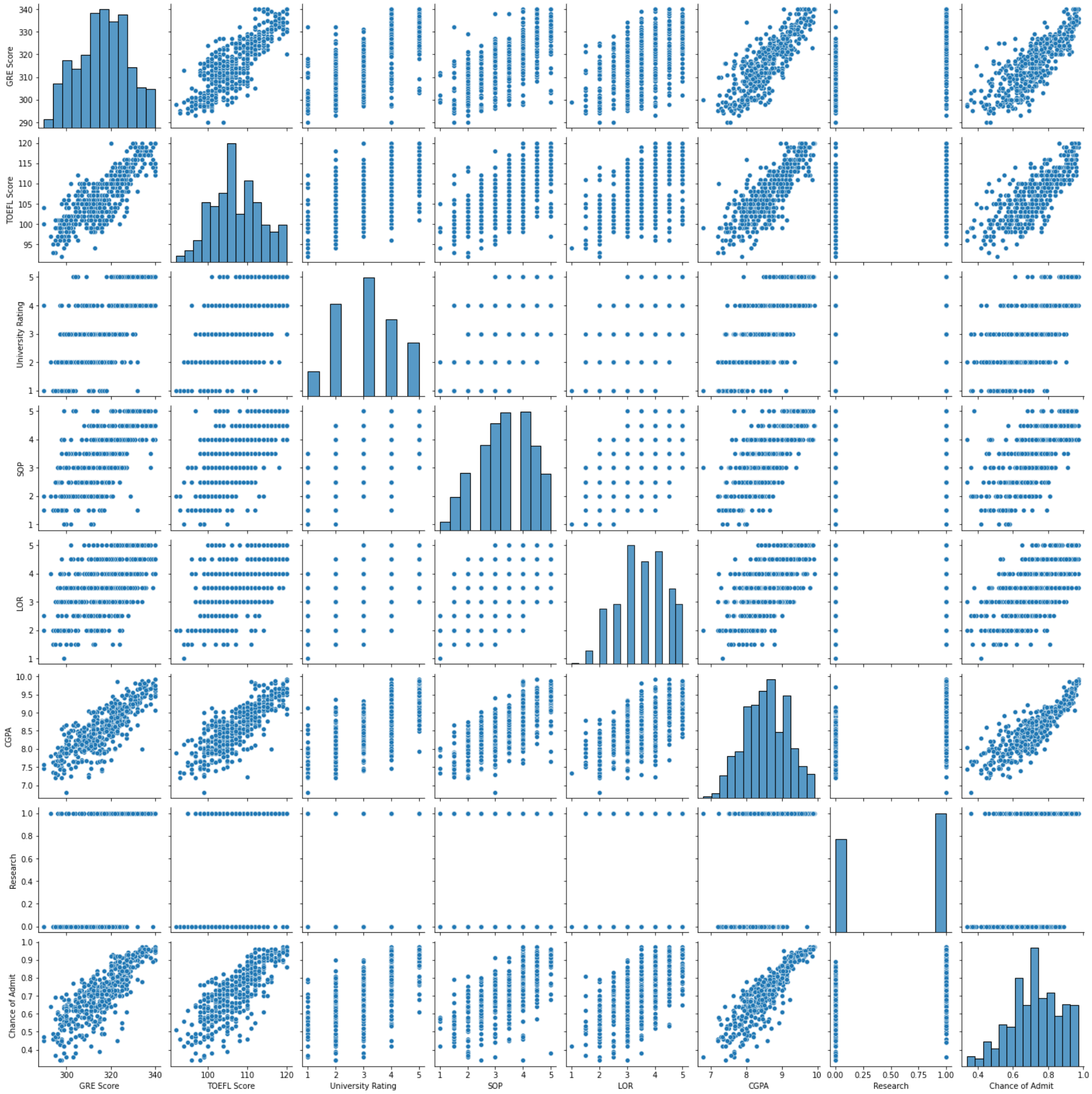
| | GRE Score | TOEFL Score | University Rating | SOP | LOR | CGPA | Research | Chan of Adr |
|---|---|---|---|---|---|---|---|---|
| **count** | 500.000000 | 500.000000 | 500.000000 | 500.000000 | 500.00000 | 500.000000 | 500.000000 | 500.000 |
| **mean** | 316.472000 | 107.192000 | 3.114000 | 3.374000 | 3.48400 | 8.576440 | 0.560000 | 0.721 |
| **std** | 11.295148 | 6.081868 | 1.143512 | 0.991004 | 0.92545 | 0.604813 | 0.496884 | 0.141 |
| **min** | 290.000000 | 92.000000 | 1.000000 | 1.000000 | 1.00000 | 6.800000 | 0.000000 | 0.340 |
| **25%** | 308.000000 | 103.000000 | 2.000000 | 2.500000 | 3.00000 | 8.127500 | 0.000000 | 0.630 |
| **50%** | 317.000000 | 107.000000 | 3.000000 | 3.500000 | 3.50000 | 8.560000 | 1.000000 | 0.720 |
| **75%** | 325.000000 | 112.000000 | 4.000000 | 4.000000 | 4.00000 | 9.040000 | 1.000000 | 0.820 |
| **max** | 340.000000 | 120.000000 | 5.000000 | 5.000000 | 5.00000 | 9.920000 | 1.000000 | 0.970 |

In [7]:

```
sns.pairplot(adm)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x1d27d4d8>
```



In [8]:

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

x_train,x_test,y_train,y_test = train_test_split(X,y,test_size = .3,random_state
```

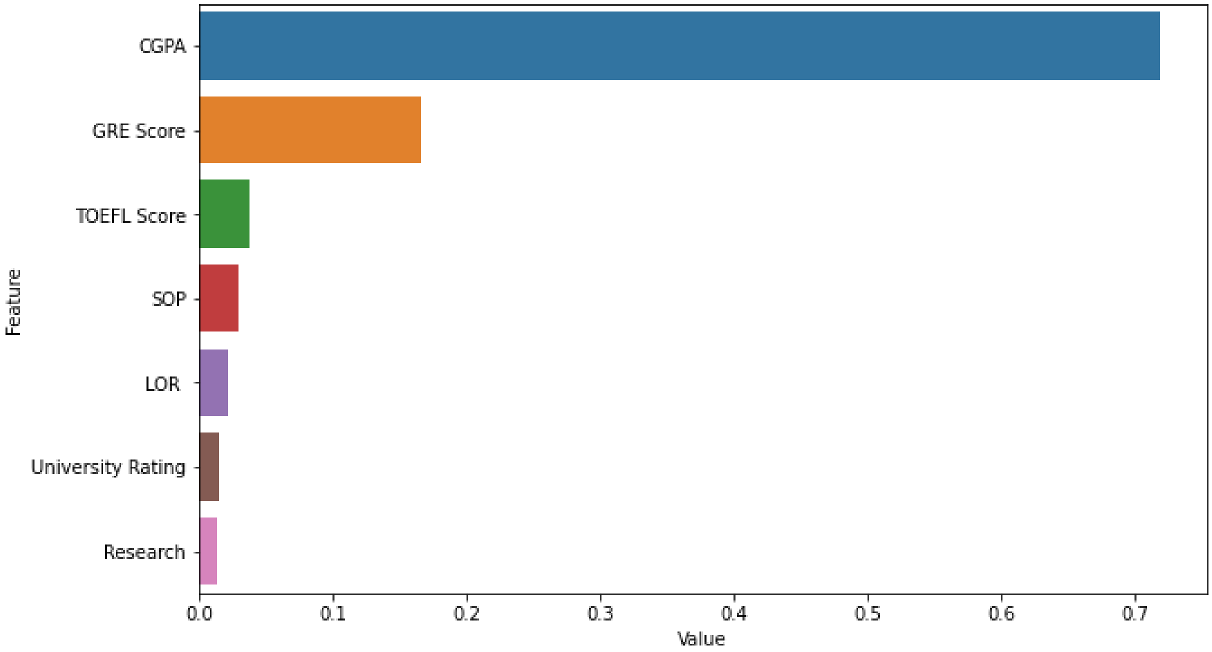
In [9]:

```
rf_model = RandomForestRegressor(n_estimators = 100,random_state = 42)
rf_model.fit(x_train,y_train)
mae=mean_absolute_error(y_test,rf_model.predict(x_test))
print('Mean absolute error for RF model: %0.4f' %mae)
```

```
Mean absolute error for RF model: 0.0438
```

In [10]:
```
feature_importance = pd.DataFrame(sorted(zip(rf_model.feature_importances_, X.col
plt.figure(figsize=(10, 6))
sns.barplot(x="Value", y="Feature", data=feature_importance.sort_values(by="Value
```

Out[10]:
```
<AxesSubplot:xlabel='Value', ylabel='Feature'>
```



In [11]:
```
model = LinearRegression()

model.fit(x_train[['GRE Score']], y_train)
```

Out[11]:
```
LinearRegression()
```

In [12]:
```
intercept = model.intercept_
coeff = model.coef_
intercept
```

Out[12]:
```
-2.571301143220596
```

In [13]:
```
coeff
```

Out[13]:
```
array([0.01040764])
```

In [14]:
```
print('Admit = {0:0.2f} + ({1:0.2f} x GRE Score)'.format(intercept, coeff[0]))
```

```
Admit = -2.57 + (0.01 x GRE Score)
```

In [15]:
```
admit = -2.57 + (0.01 * x_train['GRE Score'])
```

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

Out[16]:
```
5      0.73
116    0.42
45     0.65
16     0.60
462    0.50
Name: GRE Score, dtype: float64
```

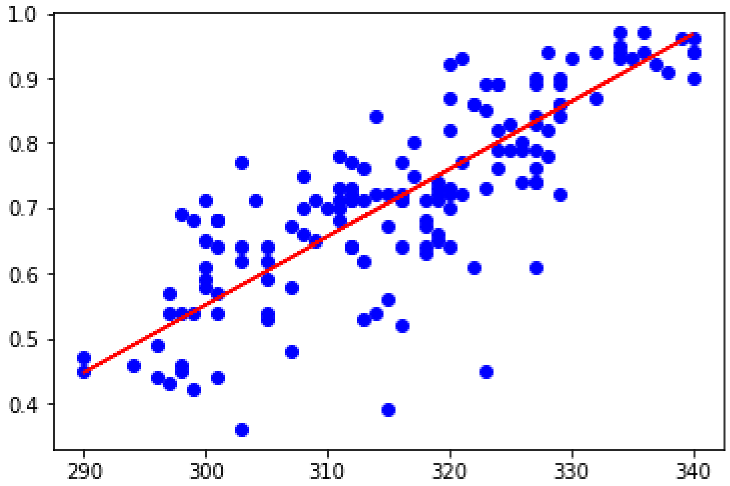
In [17]:
```

pred = model.predict(x_test[['GRE Score']])
pred.shape
```

Out[17]:
```
(150,)
```

In [18]:
```
xtest = x_test['GRE Score']
```

In [19]:
```
plt.scatter(xtest,y_test,color='b')
plt.plot(xtest,pred,color='r')
```

Out[19]:
```
[<matplotlib.lines.Line2D at 0x68c1af0>]
```



In [20]:
```

ad = intercept + (coeff[0] * x_train['GRE Score'])
ad.head()
```

Out[20]:
```
5      0.863221
116    0.540584
45     0.779960
16     0.727922
462    0.623845
Name: GRE Score, dtype: float64
```

In [ ]: