In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import MinMaxScaler
from typing import Dict
```

In [2]:
```
data = pd.read_csv('Admission_Predict.csv')
data = data.dropna()
```

In [3]:
```
data
```

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
...	...	...	...	...	...	...	...	...	...
395	396	324	110	3	3.5	3.5	9.04	1	0.82
396	397	325	107	3	3.0	3.5	9.11	1	0.84
397	398	330	116	4	5.0	4.5	9.45	1	0.91
398	399	312	103	3	3.5	4.0	8.78	0	0.67
399	400	333	117	4	5.0	4.0	9.66	1	0.95

400 rows × 9 columns

In [37]:
```
data_y = data['Research']
data_y
```

In [37]:
```
data_y = data['Research']
data_y
```

Out[37]:
```
0      1
1      1
2      1
3      1
4      0
      ..
395    1
396    1
397    1
398    0
399    1
Name: Research, Length: 400, dtype: int64
```

In [38]:
```
data.dtypes
```

Out[38]:
```
Serial No.            int64
GRE Score             int64
TOEFL Score           int64
University Rating     int64
SOP                 float64
LOR                 float64
CGPA                float64
Research              int64
Chance of Admit     float64
dtype: object
```

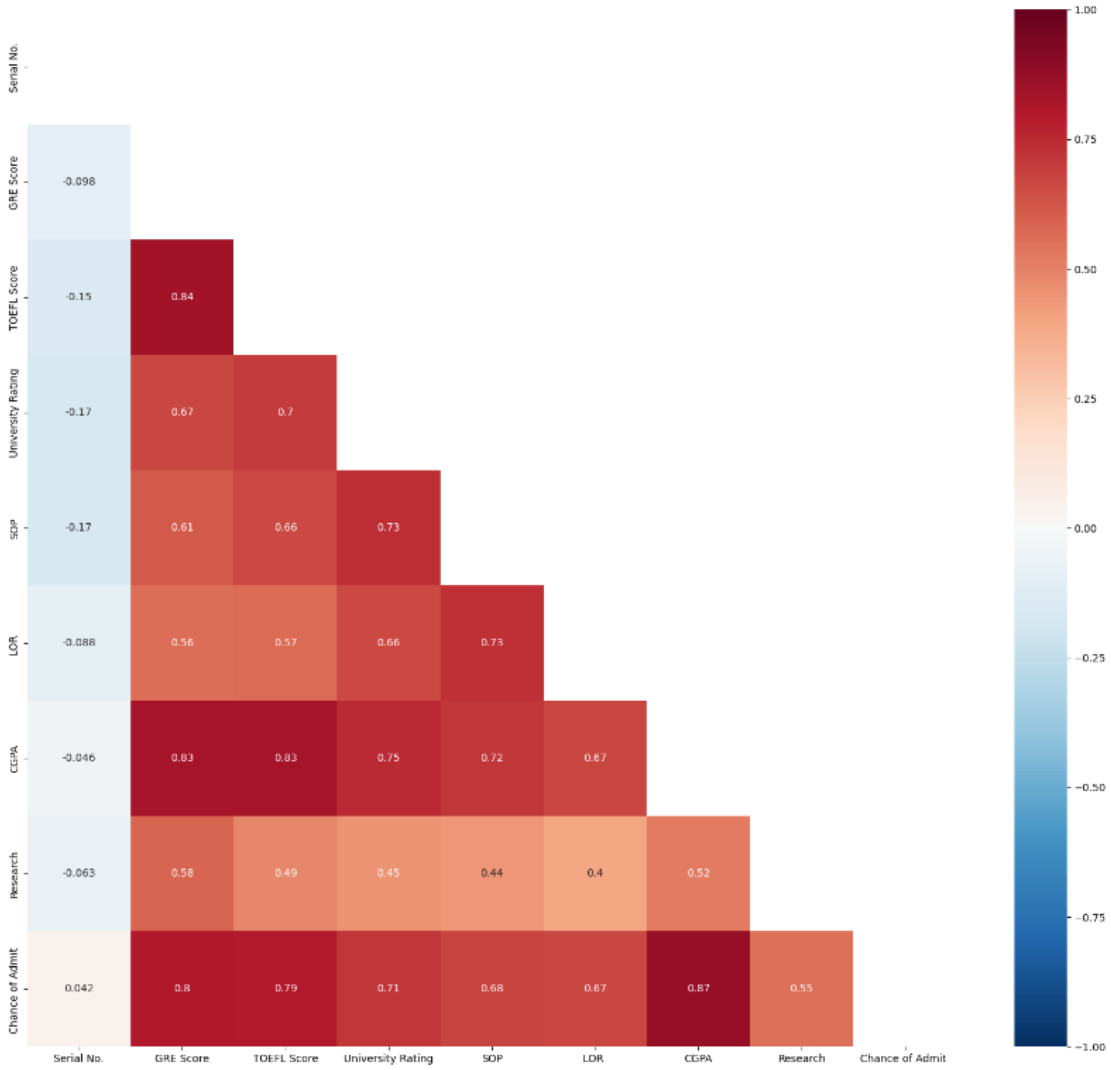
In [39]:
```
#Типы данных всех полей являются числовыми
```

In [40]:
```
data.duplicated().sum()
```

Out[40]:
```
0
```

In [41]:
```
#Дубликаты не обнаружены
```

In [42]:
```
plt.figure(figsize=(20, 18))
mask=np.triu(np.ones_like(data.corr(), dtype=bool))
sns.heatmap(data.corr(), mask=mask, annot=True, vmin=-1.0, vmax=1, center=0, cmap='RdBu_r')
```


Serial No.
GRE Score
TOEFL Score
University Rating
SOP
LOR
CGPA
Research
Chance of Admit
-0.098
-0.15
0.84
-0.17
0.67
0.7
-0.17
0.61
0.66
0.73
-0.088
0.56
0.57
0.66
0.73
-0.046
0.83
0.83
0.75
0.72
0.67
-0.063
0.58
0.49
0.45
0.44
0.4
0.52
0.042
0.8
0.79
0.71
0.68
0.67
0.87
0.55
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

#Как видно из графика, наибольшую кореляцию с целевым признаком имеют GRE SCORE, CGPA, TOEFL SCORE и University Rating. Эти признаки будут наиболее информативны при построении моделей машинного обучения.

In [43]:
```
data_y.value_counts()
```

Out[43]:
```
1    219
0    181
Name: Research, dtype: int64
```

In [44]:
```
data_X_train, data_X_test, data_y_train, data_y_test = train_test_split(data[['GRE Score','CGPA','University Rating','TOEFL Score
```

Разобьем исходную выборку на обучающую и тестовую

In [45]:
```
mms = MinMaxScaler()
```

Проиведем масштабирование данных

In [46]:
```
data_X_train_scaled = mms.fit_transform(data_X_train)
data_X_test_scaled = mms.fit_transform(data_X_test)
```

Была обучена модель логической регрессии

In [47]:
```
cl=LogisticRegression(multi_class='multinomial')
```

In [48]:
```
cl.fit(data_X_train_scaled, data_y_train)
```

Out[48]:
```
LogisticRegression(multi_class='multinomial')
```

In [49]:
```
pred_data_y_test = cl.predict(data_X_test_scaled)
pred_data_y_test
```

Out[49]:
```
array([0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1], dtype=int64)
```

Для оценки качества моделей машинного обучения были использованы метрики accuracy и F1-мера

Для оценки качества моделей машинного обучения были использованы метрики accuracy и F1-мера

In [50]:
```
accuracy_score(data_y_test, pred_data_y_test)
```

Out[50]:
```
0.825
```

In [53]:
```
f1_score(data_y_test, pred_data_y_test, average=None)
```

Out[53]:
```
array([0.825, 0.825])
```

Была обучена модель случайного леса

In [54]:
```
data_rl_cf = RandomForestClassifier(random_state=2)
data_rl_cf.fit(data_X_train_scaled, data_y_train)
```

Out[54]:
```
RandomForestClassifier(random_state=2)
```

In [55]:
```
pred_data_rf_y_test = data_rl_cf.predict(data_X_test_scaled)
pred_data_rf_y_test
```

Out[55]:
```
array([0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1,
       1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1], dtype=int64)
```

In [56]:
```
accuracy_score(data_y_test, pred_data_rf_y_test)
```

Out[56]:
```
0.7875
```

In [57]:
```
print_accuracy_score_for_classes(data_y_test, pred_data_rf_y_test)
```

```
Метка    Accuracy
0        0.8571428571428571
1        0.7333333333333333
```

In [58]:
```
f1_score(data_y_test, pred_data_rf_y_test, average=None)
```

Out[58]:
```
array([0.77922078, 0.79518072])
```

Таким образом, каждая из моделей машинного обучения классифицирует данные с довольно высокой точностью. Модель случайного леса классифицирует лучше модели логистической регрессии.