РК №1

Рубежный контроль №1. Ищенко А.С. Вариант №6: задание 1, датасет 6

Задание:

Для заданного набора данных проведите корреляционный анализ. В случае наличия пропусков в данных удалите строки или колонки, содержащие пропуски. Сделайте выводы о возможности построения моделей машинного обучения и о возможном вкладе признаков в модель.

Набор данных состоит из данных, необходимых для прогнозирования поступления в аспирантуру в Индии. Набор содержит следующие данные:

Баллы GRE (из 340) Баллы TOEFL (из 120) Рейтинг университета (из 5) Заявление о целях (из 5) Рекомендательное письмо (из 5) Средний балл бакалавриата (из 10) Исследовательский опыт (0 или 1) Шанс допуска (от 0 до 1)

In [2]:
```
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.neighbors import KNeighborsRegressor, KNeighborsClassifier

from sklearn.model_selection import GridSearchCV
from sklearn.metrics import f1_score
from sklearn.preprocessing import OrdinalEncoder
from sklearn import preprocessing
from sklearn import utils

%matplotlib inline
sns.set(style="ticks")
```

In [3]:
```
df = pd.read_csv('Admission_Predict_Ver1.1.csv')
```

In [4]:
```
df.shape
```

Out[4]:
```
(500, 9)
```

In [4]:
```
df.head(5)
```

Out[4]:

	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 [5]:
```
df.columns
```

Out[5]:
```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

Название последней колонки имеет незаметный лишний пробел в конце строки, уберем его

In [6]:
```
df.rename(columns={'Chance of Admit ': 'Chance of Admit'}, inplace=True)
```

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

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

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         500 non-null    int64  
 1   GRE Score          500 non-null    int64  
 2   TOEFL Score        500 non-null    int64  
 3   University Rating  500 non-null    int64  
 4   SOP                500 non-null    float64
 5   LOR                500 non-null    float64
 6   CGPA               500 non-null    float64
 7   Research           500 non-null    int64  
 8   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 35.3 KB
```

In [9]:
```
df.dtypes
```

Out[9]:
```
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
```

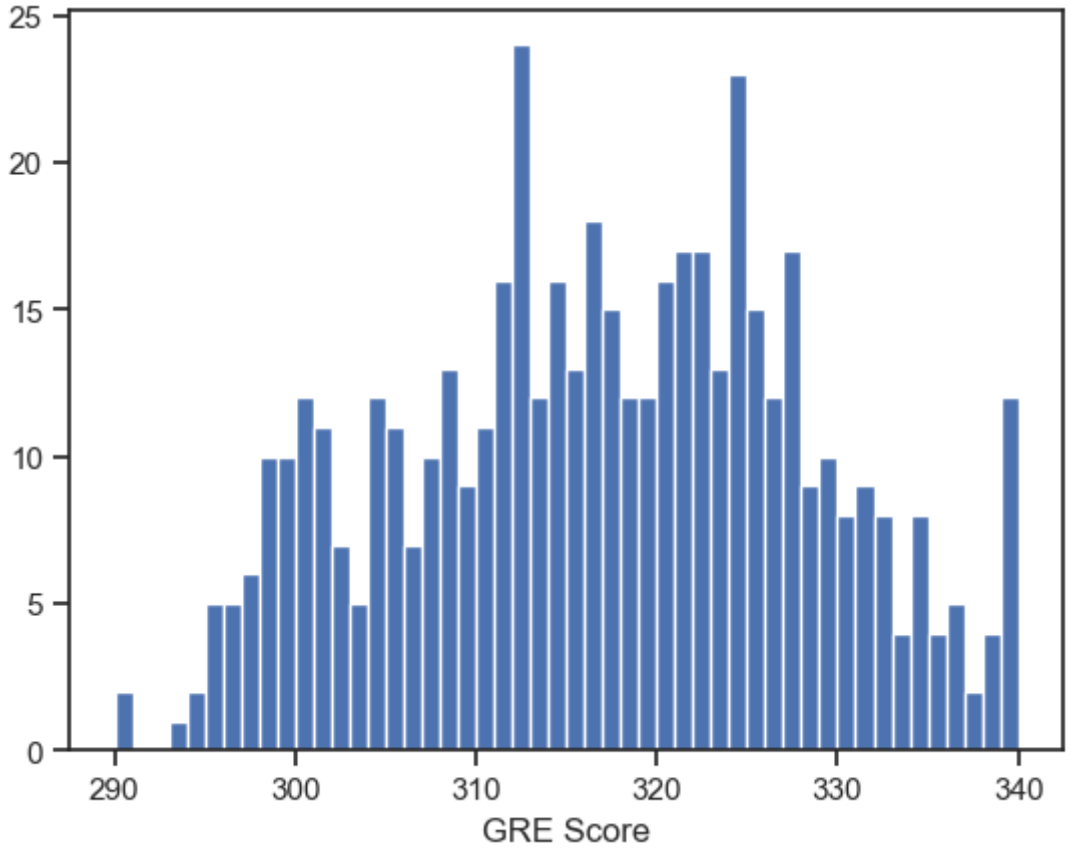
In [10]:
```
print('Число пропусков')
df.isna().sum()
```

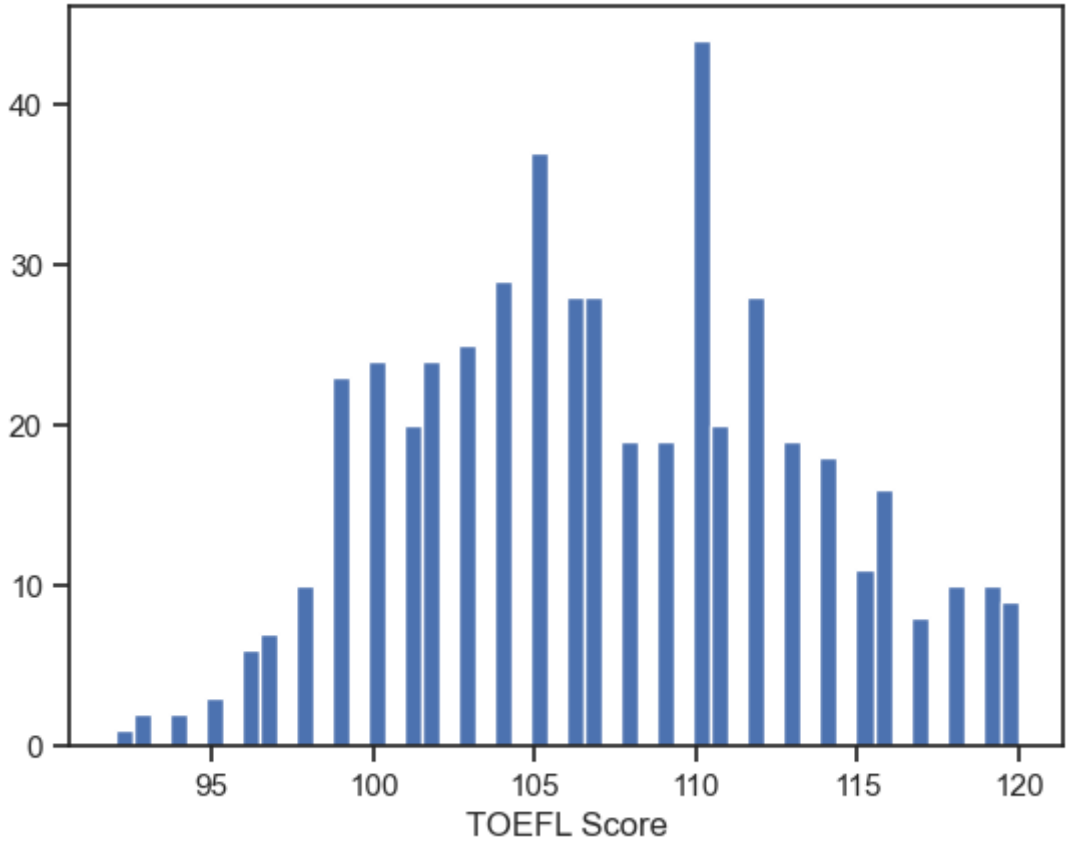
```
Число пропусков
```

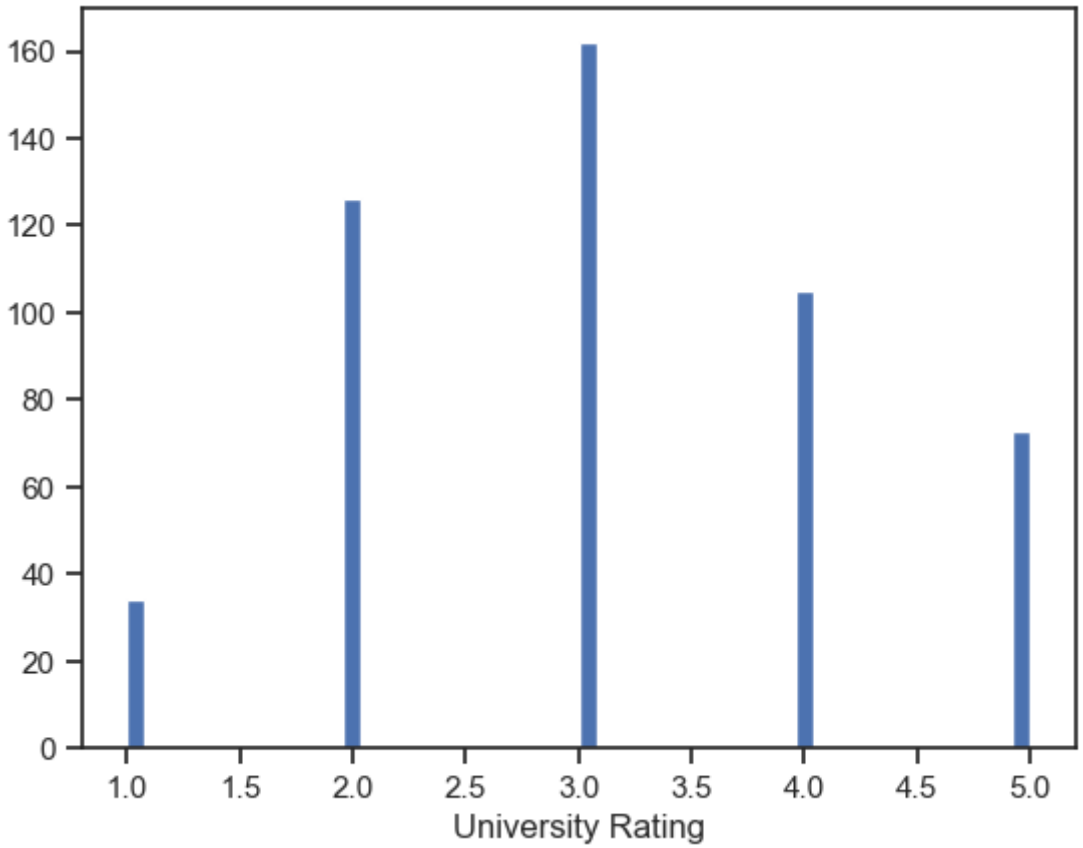
Out[10]:
```
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

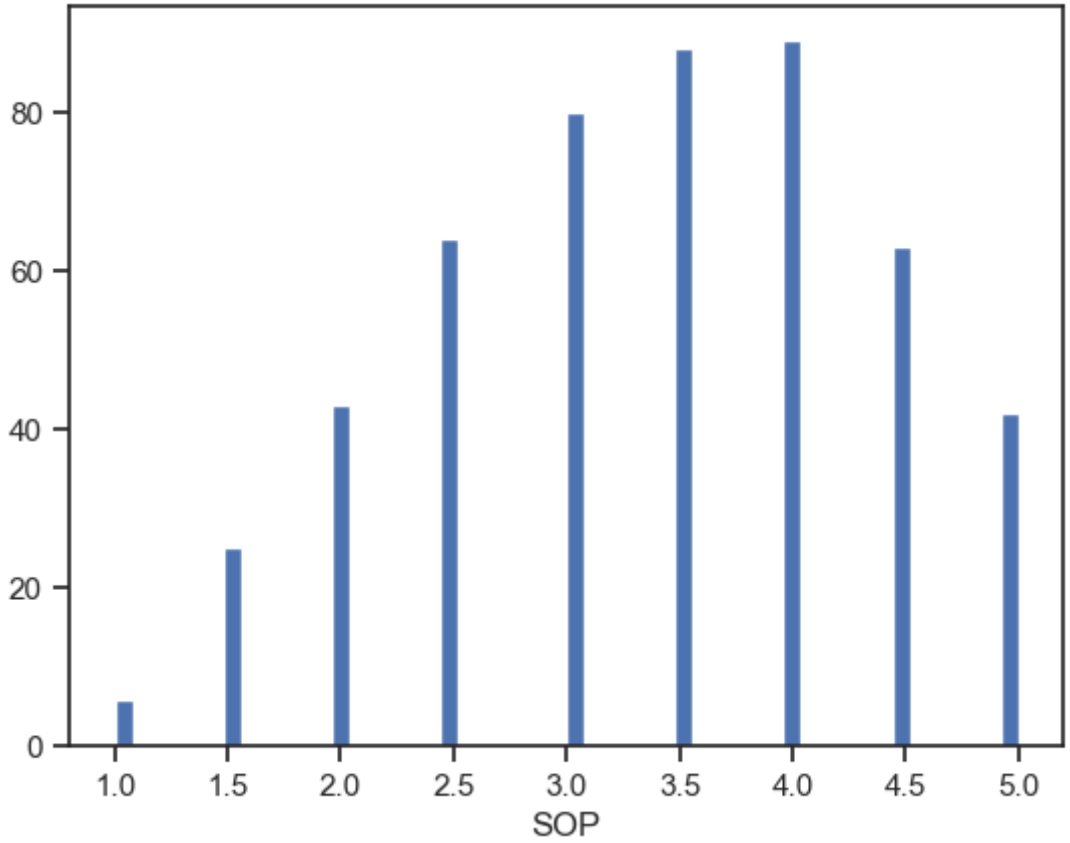
Визуализация данных

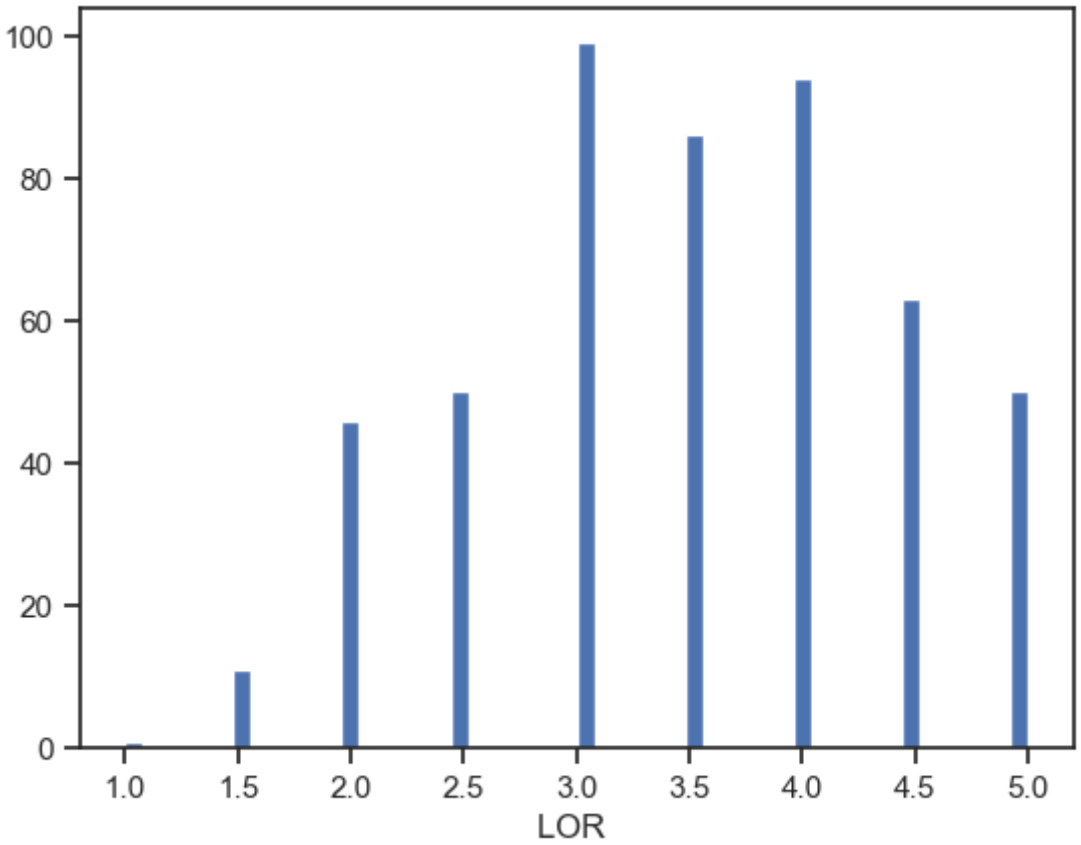
In [13]:
```
df_new = df.drop(columns='Serial No.')

for column in df_new:
    plt.hist(df_new[column], 50)
    plt.xlabel(column)
    plt.show()
```

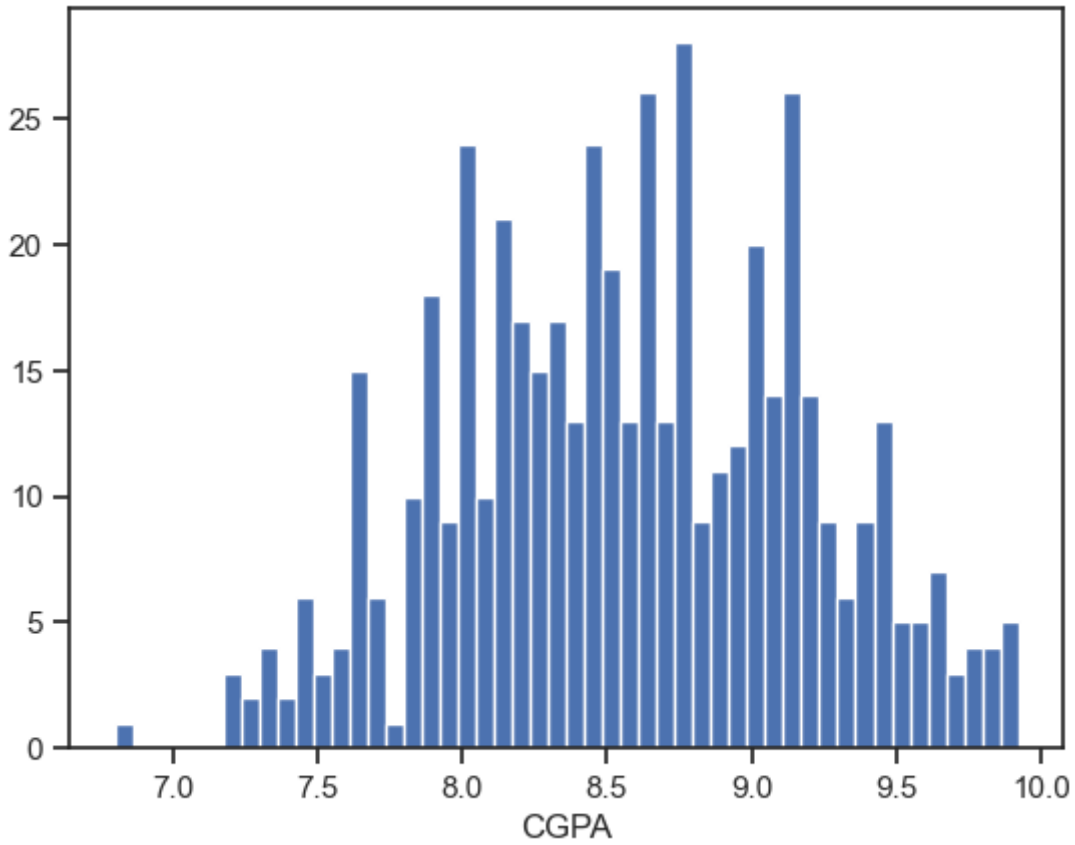


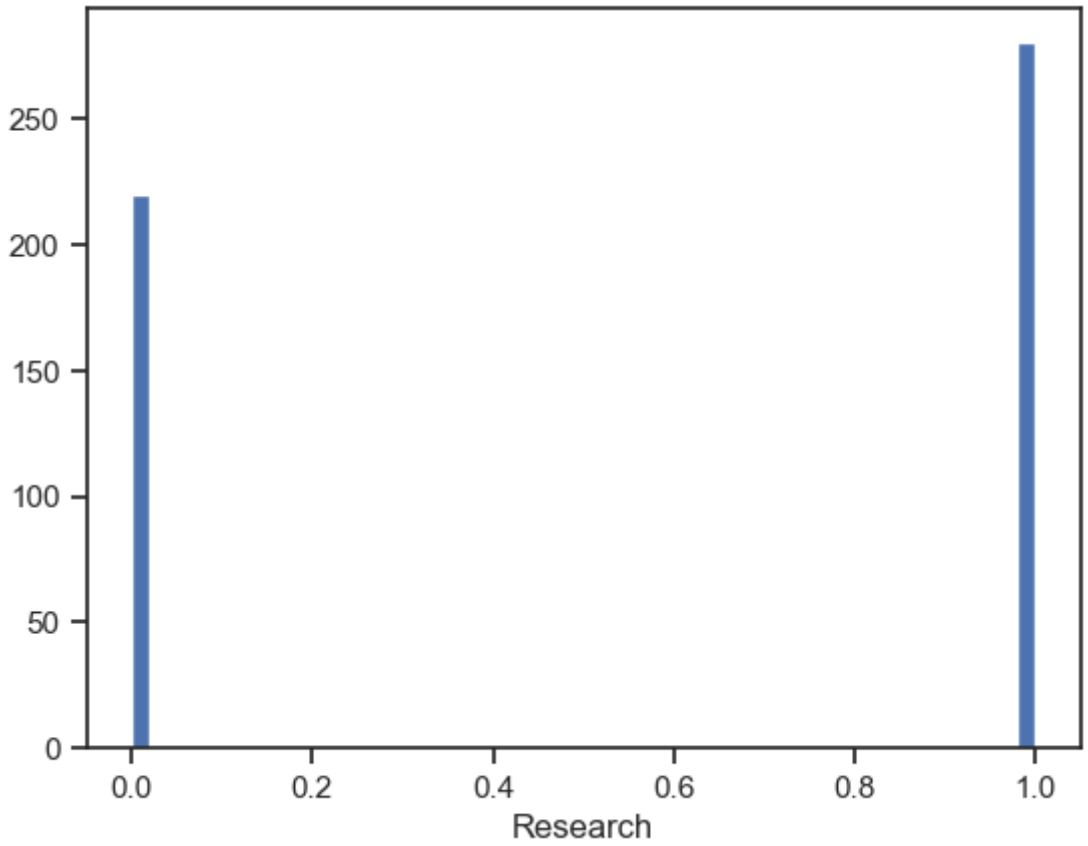

40
30
20
10
0
95
100
105
110
115
120
TOEFL Score

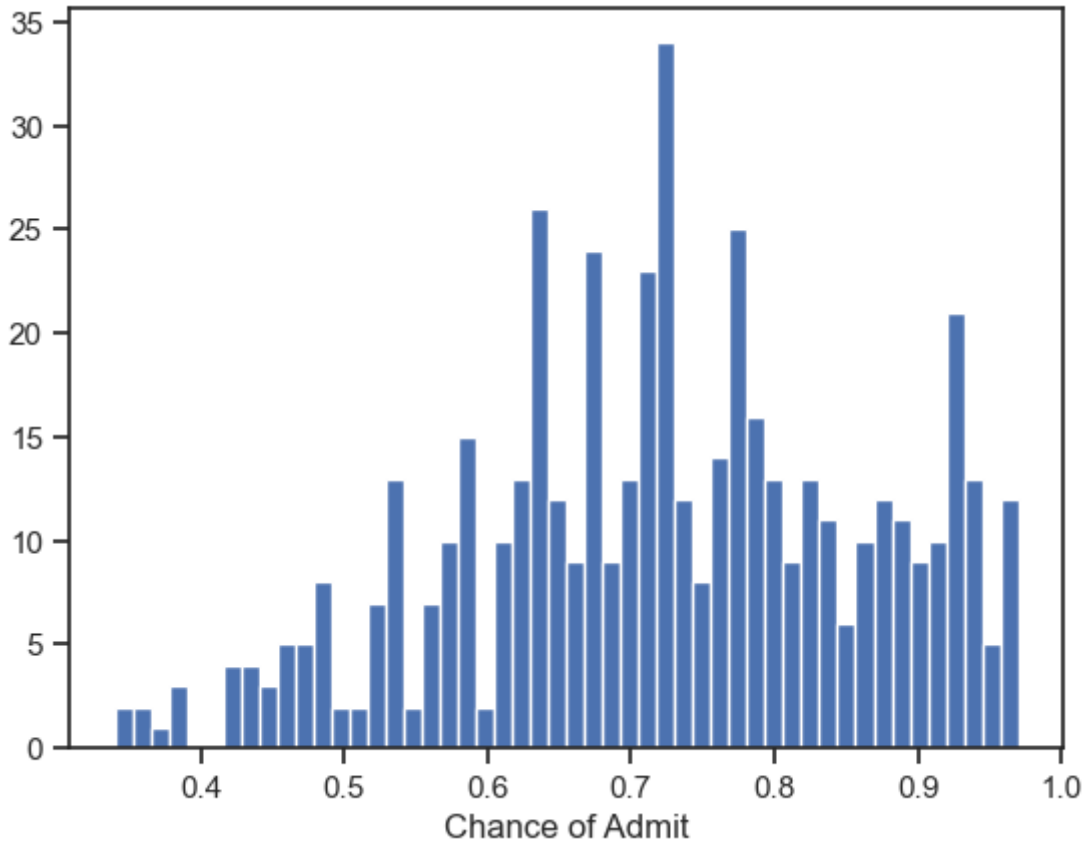

160
140
120
100
80
60
40
20
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
University Rating


80
60
40
20
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
SOP


100
80
60
40
20
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
LOR


25
20
15
10
5
0
7.0
7.5
8.0
8.5
9.0
9.5
10.0
CGPA


250
200
150
100
50
0
0.0
0.2
0.4
0.6
0.8
1.0
Research

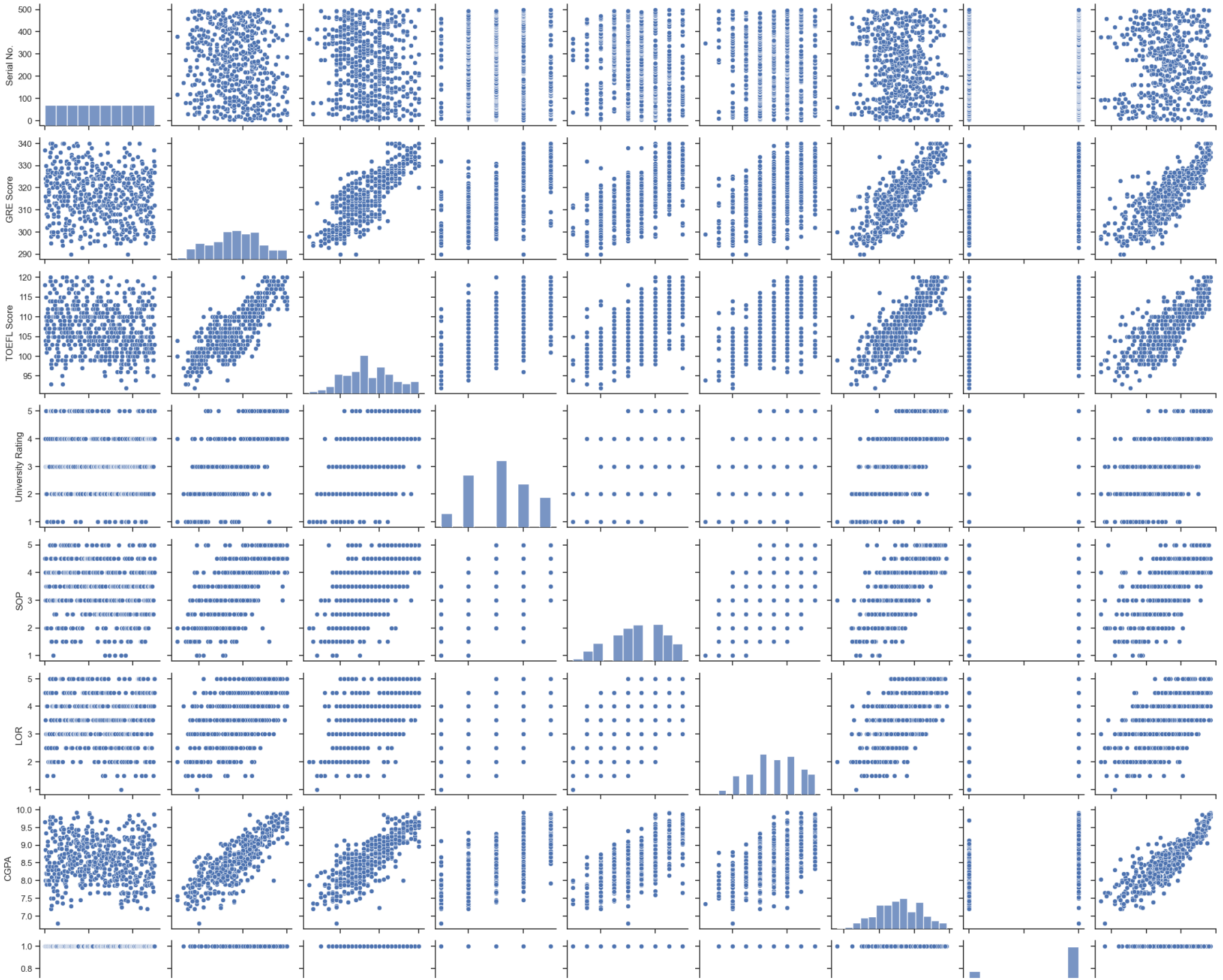


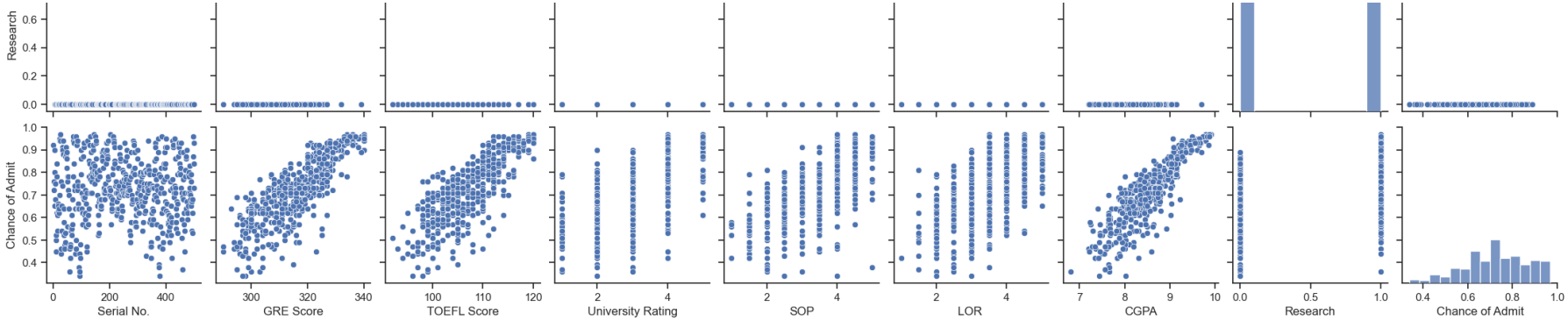
Парный анализ

Все данные в наборе являются числовыми, поэтому в подготовке для парного анализа не нуждаются

In [14]:
```
sns.pairplot(df)
```

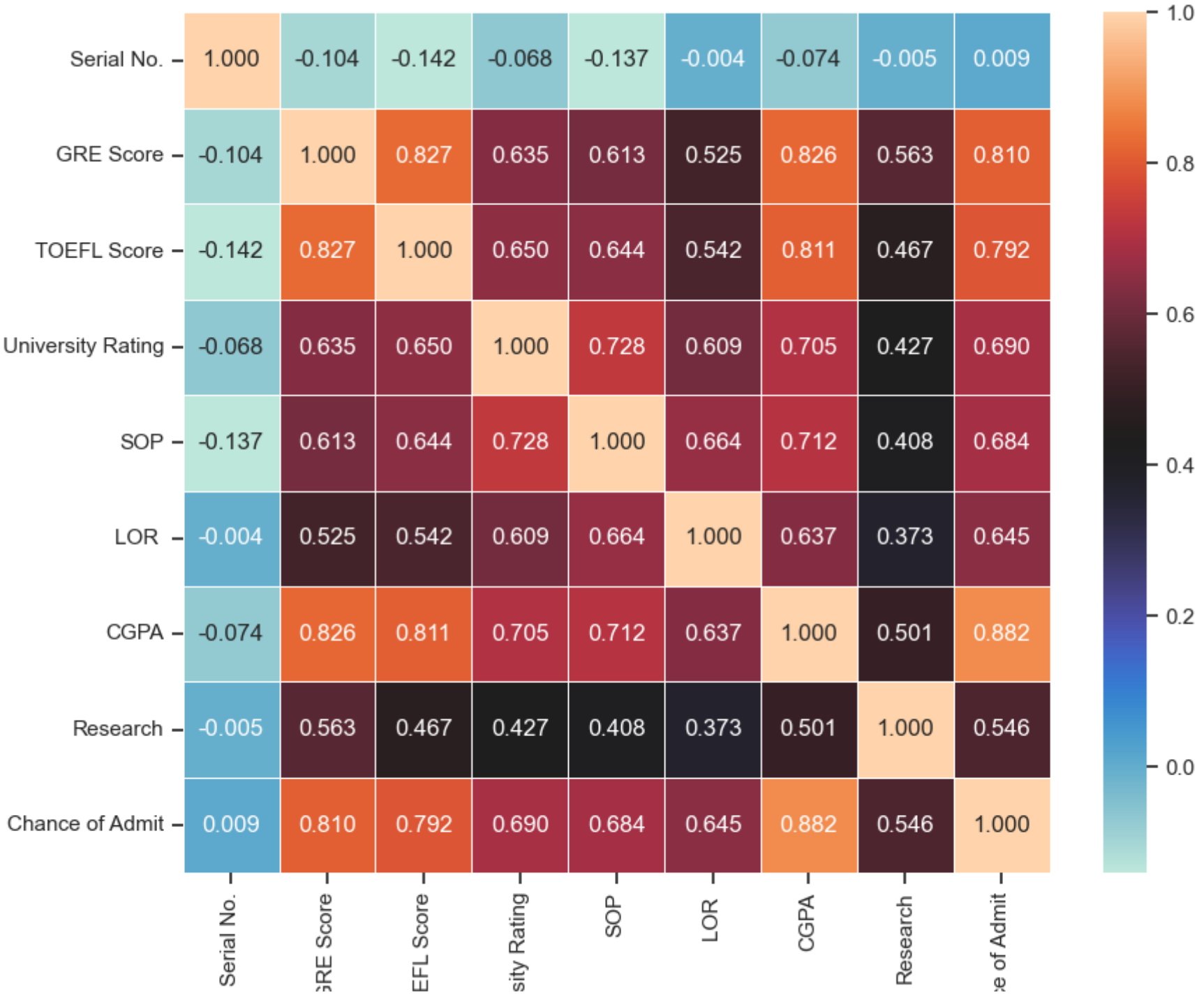
Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x7fcb27593940>
```


Serial No.
GRE Score
TOEFL Score
University Rating
SOP
LOR
CGPA


Research
Chance of Admit
Serial No.
GRE Score
TOEFL Score
University Rating
SOP
LOR
CGPA
Research
Chance of Admit

Тепловая карта

In [15]:
```
f,ax = plt.subplots(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, linewidths=.5, fmt= '.3f',ax=ax,cmap="icefire");
```


Serial No.
GRE Score
TOEFL Score
University Rating
SOP
LOR
CGPA
Research
Chance of Admit
1.000 -0.104 -0.142 -0.068 -0.137 -0.004 -0.074 -0.005 0.009
-0.104 1.000 0.827 0.635 0.613 0.525 0.826 0.563 0.810
-0.142 0.827 1.000 0.650 0.644 0.542 0.811 0.467 0.792
-0.068 0.635 0.650 1.000 0.728 0.609 0.705 0.427 0.690
-0.137 0.613 0.644 0.728 1.000 0.664 0.712 0.408 0.684
-0.004 0.525 0.542 0.609 0.664 1.000 0.637 0.373 0.645
-0.074 0.826 0.811 0.705 0.712 0.637 1.000 0.501 0.882
-0.005 0.563 0.467 0.427 0.408 0.373 0.501 1.000 0.546
0.009 0.810 0.792 0.690 0.684 0.645 0.882 0.546 1.000
Serial No.
GRE Score
EFL Score
sity Rating
SOP
LOR
CGPA
Research
e of Admit
1.0
0.8
0.6
0.4
0.2
0.0

основании этого можно сделать модель предсказания шанса поступления судя по этим параметрам.

Для этого можно использовать классифкацию (или регрессию) на основе метода k-ближайших соседей (также можно использовать SVM и дерево решений).

Найдем наилучшее количество соседей для метода k-ближайших соседей

In [22]:
```
data_train, data_test, train_target, test_target = train_test_split(df, df['Chance of Admit'], test_size=0.1, random_s
dtest = data_test[['GRE Score','University Rating', 'Chance of Admit']]

lab = preprocessing.LabelEncoder()
test_target_transformed = lab.fit_transform(test_target)

knn = KNeighborsClassifier()
parametrs = {'n_neighbors': range(1, 10)}
grid = GridSearchCV(knn, parametrs, cv=4, scoring='f1')
grid.fit(dtest, test_target_transformed)
grid.best_params_
```

Out[22]:
```
{'n_neighbors': 1}
```

In [ ]:
```

```