Probability of entering your dream Graduation Program

Sean Xu

DATA1030 Fall22 S01

Hands-on Data Science

October 17th 2022

http://localhost:8908/notebooks/Desktop/brown/DATA1030-Fall2022/final%20project/Final%20project-data%20set%20of%20Graduate%20Admission.ipynb

Introduction

- Problem:

Whether you can enter your dream program?

What can you do to improve the probability?

- Target variables: Chance of Admitted

- Regression: Probability (0% - 100%)

- Kaggle: UCLA Database

```
In [108]: # target variable - Chance of Admit
          print(df['Chance of Admit'])
```

```
0      0.92
1      0.76
2      0.72
3      0.80
4      0.65
       ...
395    0.82
396    0.84
397    0.91
398    0.67
399    0.95
Name: Chance of Admit, Length: 400, dtype: float64
```

EDA

1.GRE Scores (out of 340)
2.TOEFL Scores (out of 120)
3.University Rating (out of 5)
4.Statement of Purpose (out of 5)
5.Letter of Recommendation Strength (out of 5)
6.Undergraduate GPA (out of 10)
7.Research Experience (either 0 or 1)

Target: Chance of Admit (ranging from 0 to 1)

- Continuous
- Ordinary
- Categorized

```
In [81]: df.describe()
```

Out[81]:

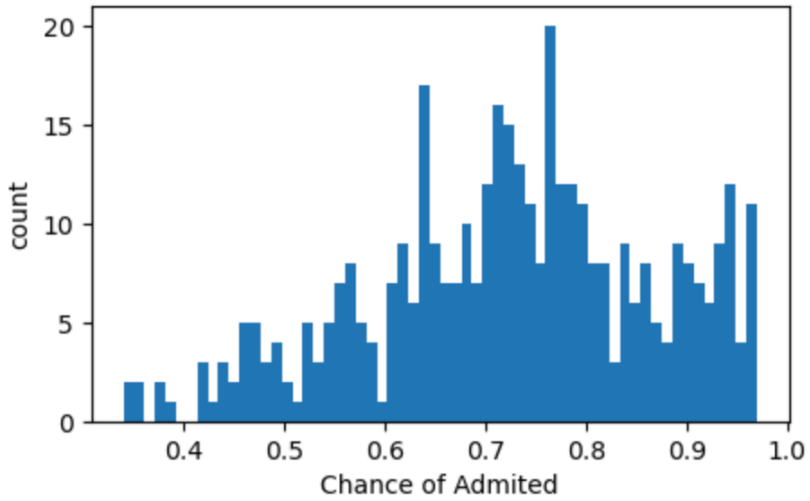
	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.547500	0.724350
std	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.498362	0.142609
min	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	0.340000
25%	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.000000	0.640000
50%	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.000000	0.730000
75%	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.000000	0.830000
max	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.970000

World average GRE score in 2022
Quantitative Reasoning: 153.66.
Verbal Reasoning: 150.37 in total = 301.03

Target Variable

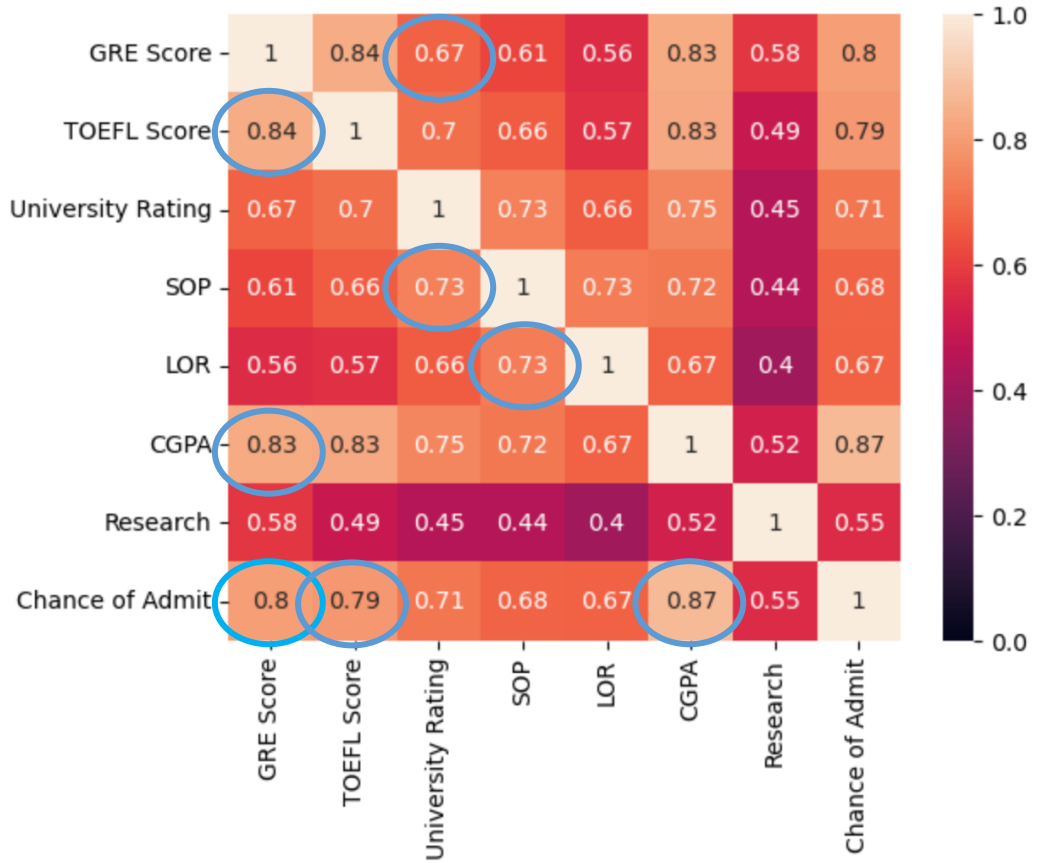
In [117]:
```
import matplotlib
from matplotlib import pylab as plt

print(df['Chance of Admit'].describe())
plt.figure(figsize=(5,3))

df['Chance of Admit'].plot.hist(bins = df['Chance of Admit'].nunique())
plt.xlabel('Chance of Admited')
plt.ylabel('count')
plt.show()
```

```
count    400.000000
mean       0.724350
std        0.142609
min        0.340000
25%        0.640000
50%        0.730000
75%        0.830000
max        0.970000
Name: Chance of Admit, dtype: float64
```



EDA

```
In [76]: import seaborn as sns
         sns.heatmap(df.corr(), vmin=0, vmax=1, annot = True)

Out[76]: <AxesSubplot:>
```



Not so surprising

- V.S. target variables

Highest three:
GPA (0.87) GRE (0.8) TOFEL (0.79)
Lowest:
Research (0.55)

- Hard skills

GPA and GRE and Tofel are all over 0.8

- Soft skills

LOR and SOP are all over 0.7

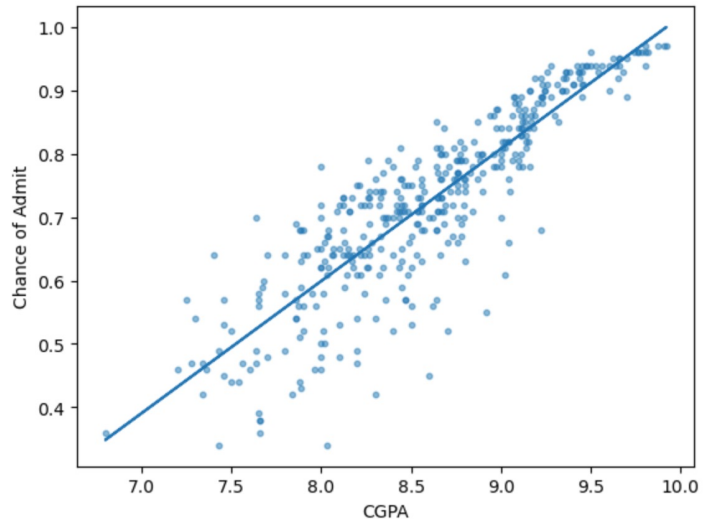
But what surprised me was
University ranking

EDA

Continuous vs. Continuous

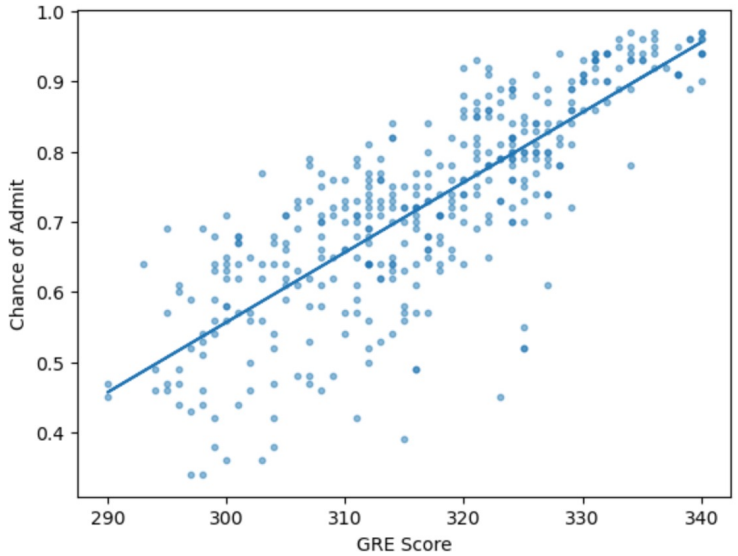
In [143]:
```
import numpy as np

plt.figure(figsize=(5,3))
df.plot.scatter('CGPA','Chance of Admit',s=10,alpha=0.5) # alpha=0.1,s=10
m, b = np.polyfit(df['CGPA'],df['Chance of Admit'] ,1)
plt.plot(df['CGPA'], m*df['CGPA']+b)
plt.show()
```

```
<Figure size 500x300 with 0 Axes>
```



In [144]:
```
plt.figure(figsize=(5,3))
df.plot.scatter('GRE Score','Chance of Admit',s=10,alpha=0.5) # alpha=0.1,s=10
m, b = np.polyfit(df['GRE Score'],df['Chance of Admit'] ,1)
plt.plot(df['GRE Score'], m*df['GRE Score']+b)
plt.show()
```

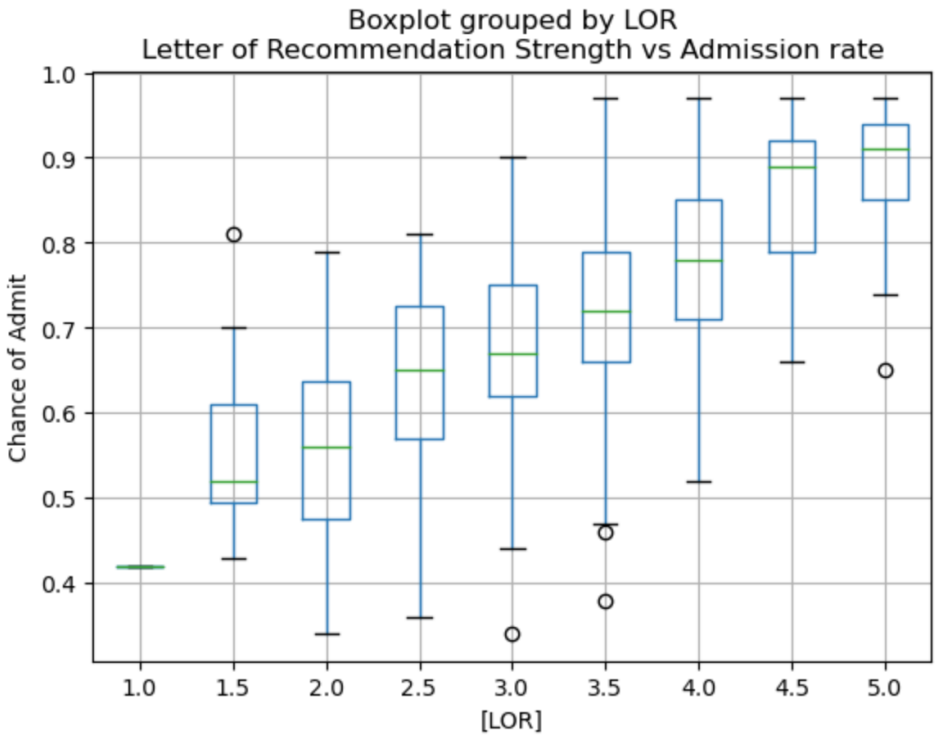
```
<Figure size 500x300 with 0 Axes>
```



EDA

Continuous vs. Ordinary

```
In [149]: df[['Chance of Admit','LOR']].boxplot(by='LOR')
          plt.ylabel('Chance of Admit')
          plt.title('Letter of Recommendation Strength vs Admission rate')
          plt.show()
```



- If we want the chance of admission >90%
  - Need LOR strength more than 3
- Although LOR increase will increase mean
  - Huge change 4 – 4.5
  - Not much change 4.5-5

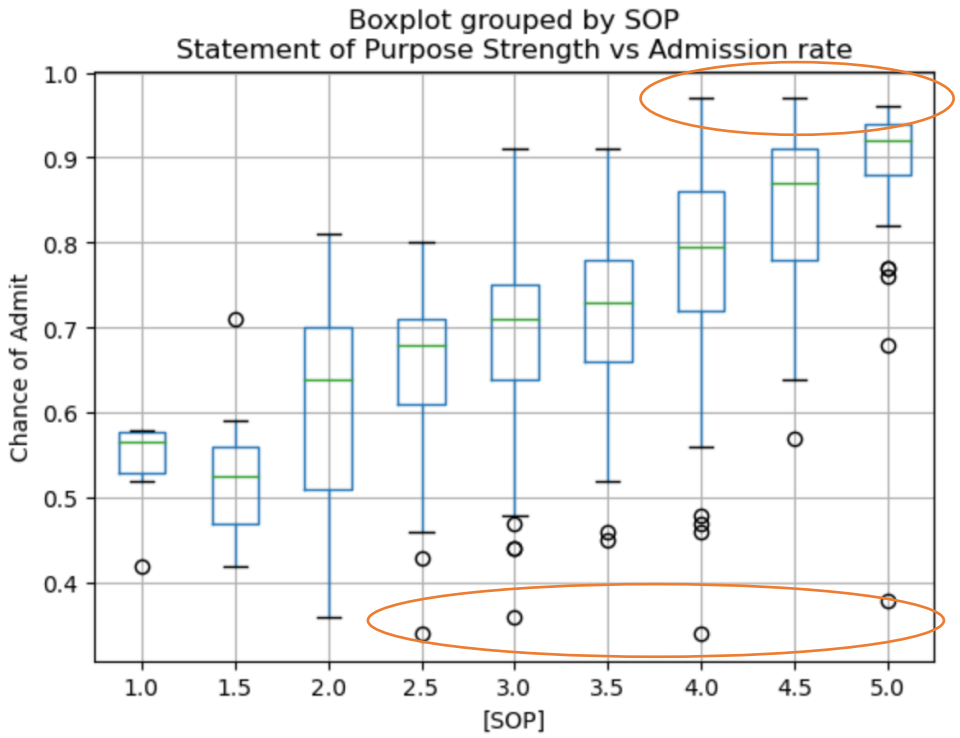
```
print(df['LOR'].value_counts())
```

```
3.0    85
4.0    77
3.5    73
4.5    45
2.5    39
2.0    38
5.0    35
1.5     7
1.0     1
Name: LOR, dtype: int64
```

EDA

Continuous vs. Ordinary

```
df[['Chance of Admit','SOP']].boxplot(by='SOP')
plt.ylabel('Chance of Admit')
plt.title('Statement of Purpose Strength vs Admission rate')
plt.show()
```



- Something Same as LOR
- The maximum chance of admission of 5.0 is lower than 4.5
- Much more outliers
  - Even in 5.0 SOP, have a 40% chance of admission

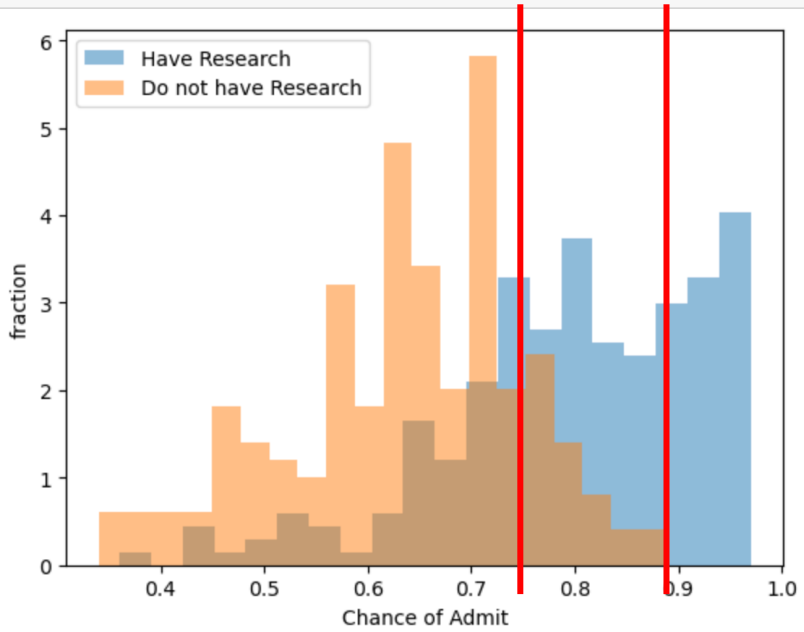
```
In [152]: print(df['SOP'].value_counts())
```

```
4.0    70
3.5    70
3.0    64
4.5    53
2.5    47
5.0    37
2.0    33
1.5    20
1.0     6
Name: SOP, dtype: int64
```

EDA Continuous vs. categorized

- Most of the people do not have research has less than 75% chance of admit
- If you want chance of admit > 90% need research

```
In [161]: categories = df['Research'].unique()

for c in categories:
    if (c == 0):
        label_graph = 'Do not have Research'
    else:
        label_graph = 'Have Research'
    plt.hist(df[df['Research']==c]['Chance of Admit'],alpha=0.5,label = label_graph,bins=20,density=True)
plt.legend()
plt.ylabel('fraction')
plt.xlabel('Chance of Admit')
plt.show()
```



Splitting Data

- One thing I notice:

In [182]:
```
from sklearn.model_selection import train_test_split
def basic_split(X,y,train_size,val_size,test_size,random_state):
    # test the inputs
    if ((train_size + val_size + test_size) == 1) & isinstance(random_state,int) == True:
        print ('input is correct')
    else:
        print ('input is wrong')
    # perform basic split
    X_train, X_other, y_train, y_other = train_test_split(X,y, train_size = train_size, \
                                                          stratify = X['University Rating'], random_state = random_state)
    X_val, X_test, y_val, y_test = train_test_split(X_other,y_other,train_size= val_size/(1-train_size),\
                                                    stratify = X_other['University Rating'], random_state = random_stat
    # test the outputs
    print ('If we use stratification:')
    print(np.unique(X_train['University Rating'],return_counts=True))
    print(np.unique(X_val['University Rating'],return_counts=True))
    print(np.unique(X_test['University Rating'],return_counts=True))

    return X_train, y_train, X_val, y_val, X_test, y_test

y = df['Chance of Admit']
X = df.loc[:, df.columns != 'Chance of Admit']
random_state = 42
X_train, y_train, X_val, y_val, X_test, y_test = basic_split(X,y,0.6,0.2,0.2,random_state)
```

```
input is correct
If we use stratification:
(array([1, 2, 3, 4, 5]), array([16, 64, 80, 44, 36]))
(array([1, 2, 3, 4, 5]), array([ 5, 21, 27, 15, 12]))
(array([1, 2, 3, 4, 5]), array([ 5, 22, 26, 15, 12]))
```

Size of Data set is small, it has only 400 rows

Missing Values

- It is a well designed and organized data set

```
In [16]: perc_missing_value = df.isnull().sum(axis=0)/df.shape[0]
         print('percentage of missing value:', perc_missing_value)
```

```
percentage of missing value: GRE Score            0.0
TOEFL Score           0.0
University Rating     0.0
SOP                   0.0
LOR                   0.0
CGPA                  0.0
Research              0.0
Chance of Admit       0.0
dtype: float64
```

Preprocessing

- Standardized GRE, TOFEL, and GPA
- Use MinMax scaler

In [23]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scaler.fit(X_train.iloc[:, 0:2])
X_train.iloc[:, 0:2] = scaler.transform(X_train.iloc[:, 0:2])
scaler.fit(X_train.iloc[:, 5:6])
X_train.iloc[:, 5:6] = scaler.transform(X_train.iloc[:, 5:6])

print("after normalization: ")
print(X_train.head())
```

```
after normalization:
     GRE Score  TOEFL Score  University Rating  SOP  LOR       CGPA  Research
375       0.28     0.321429                  2  2.0  2.5   0.275641         0
141       0.84     0.928571                  2  4.5  3.5   0.820513         1
349       0.46     0.321429                  3  2.5  3.0   0.397436         0
163       0.54     0.464286                  3  3.5  3.0   0.564103         0
72        0.62     0.678571                  5  5.0  5.0   0.849359         1
```