15장. 다중 회귀 분석 과제

In [1]:

```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in c:₩users₩ghkrg₩anaconda3₩envs₩data_mining
₩lib₩site-packages (0.11.1)
Requirement already satisfied: matplotlib>=2.2 in c:₩users₩ghkrg₩anaconda3₩envs₩data
_mining₩lib₩site-packages (from seaborn) (3.3.4)
Requirement already satisfied: scipy>=1.0 in c:₩users₩ghkrg₩anaconda3₩envs₩data_mini
ng₩lib₩site-packages (from seaborn) (1.6.3)
Requirement already satisfied: pandas>=0.23 in c:₩users₩ghkrg₩anaconda3₩envs₩data_mi
ning₩lib₩site-packages (from seaborn) (1.2.3)
Requirement already satisfied: numpy>=1.15 in c:₩users₩ghkrg₩anaconda3₩envs₩data_min
ing₩lib₩site-packages (from seaborn) (1.20.1)
Requirement already satisfied: cycler>=0.10 in c:₩users₩ghkrg₩anaconda3₩envs₩data_mi
ning₩lib₩site-packages (from matplotlib>=2.2->seaborn) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in c:₩users₩ghkrg₩anaconda3₩envs
₩data_mining₩lib₩site-packages (from matplotlib>=2.2->seaborn) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:₩users₩ghkrg₩anaconda3₩envs₩da
ta_mining₩lib₩site-packages (from matplotlib>=2.2->seaborn) (1.3.1)
Requirement already satisfied: pillow>=6.2.0 in c:₩users₩ghkrg₩anaconda3₩envs₩data_m
ining₩lib₩site-packages (from matplotlib>=2.2->seaborn) (8.1.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:₩users
₩ghkrg₩anaconda3₩envs₩data_mining₩lib₩site-packages (from matplotlib>=2.2->seaborn)
(2.4.7)
Requirement already satisfied: six in c:₩users₩ghkrg₩anaconda3₩envs₩data_mining₩lib
₩site-packages (from cycler>=0.10->matplotlib>=2.2->seaborn) (1.15.0)
Requirement already satisfied: pytz>=2017.3 in c:₩users₩ghkrg₩anaconda3₩envs₩data_mi
ning₩lib₩site-packages (from pandas>=0.23->seaborn) (2021.1)
```

1. 데이터셋

1.1 데이터셋 읽기 (Graduate Admission)

- 특징 : GRE Scores, TOEFL, CGPA, SOP Rating, LOR Rating, Research Papers, University Ratings
- 타겟 : Chance of Admit (대학원 입학 허가 확률)
- https://www.kaggle.com/mohansacharya/graduate-admissions (https://www.kaggle.com/mohansacharya/graduate-admissions)

In [2]:

```
import os
import pandas as pd

path1 = os.path.join('data', 'Admission_Predict_Ver1.1.csv')
path2 = os.path.join('data', 'Admission_Predict.csv')

data1 = pd.read_csv(path1)
data2 = pd.read_csv(path2)

dataset = pd.concat([data1, data2])

dataset.sample(5)
```

Out[2]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
322	323	314	107	2	2.5	4.0	8.27	0	0.72
372	373	336	119	4	4.5	4.0	9.62	1	0.95
131	132	303	105	5	5.0	4.5	8.65	0	0.77
149	150	311	106	2	3.5	3.0	8.26	1	0.79
344	345	295	96	2	1.5	2.0	7.34	0	0.47

1.2 Series No 컬럼 삭제 (Q1)

샘플마다 유일하게 존재하는 ID 역할을 하는 컬럼은 학습에 방해가 되므로 삭제하시오.

In [3]:

```
# your code
dataset = dataset.drop('Serial No.', axis=1)
```

2. 데이터 탐색

2.1 요약 통계량

In [4]:

```
dataset.describe()
```

Out[4]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Ch
count	900.000000	900.000000	900.000000	900.000000	900.00000	900.000000	900.000000	900
mean	316.621111	107.288889	3.102222	3.385556	3.47000	8.586433	0.554444	0
std	11.369700	6.073968	1.143048	0.997612	0.91319	0.600822	0.497303	0
min	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0
25%	308.000000	103.000000	2.000000	2.500000	3.00000	8.140000	0.000000	0
50%	317.000000	107.000000	3.000000	3.500000	3.50000	8.570000	1.000000	0
75%	325.000000	112.000000	4.000000	4.000000	4.00000	9.052500	1.000000	0
max	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0

2.2 누락 데이터 확인

In [5]:

```
dataset.isnull().sum()
```

Out[5]:

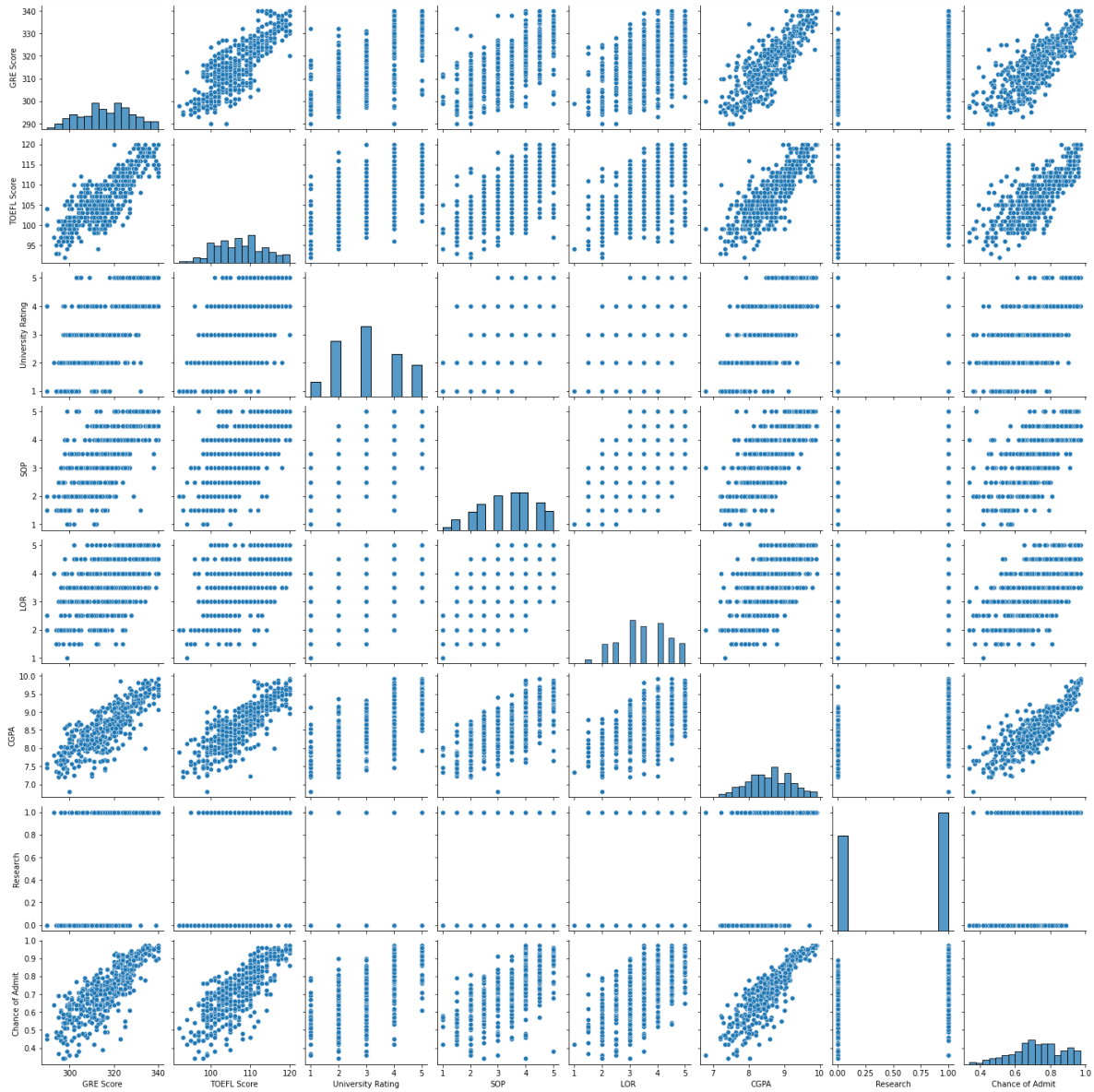
```
GRE Score          0
TOEFL Score        0
University Rating  0
SOP                0
LOR                0
CGPA               0
Research           0
Chance of Admit    0
dtype: int64
```

2.3 산포도 행렬

- 행렬의 대각 방향 : 히스토그램
- 행렬의 나머지 셀 : 두 변수 간의 상관성 분석을 위한 산포도

In [6]:

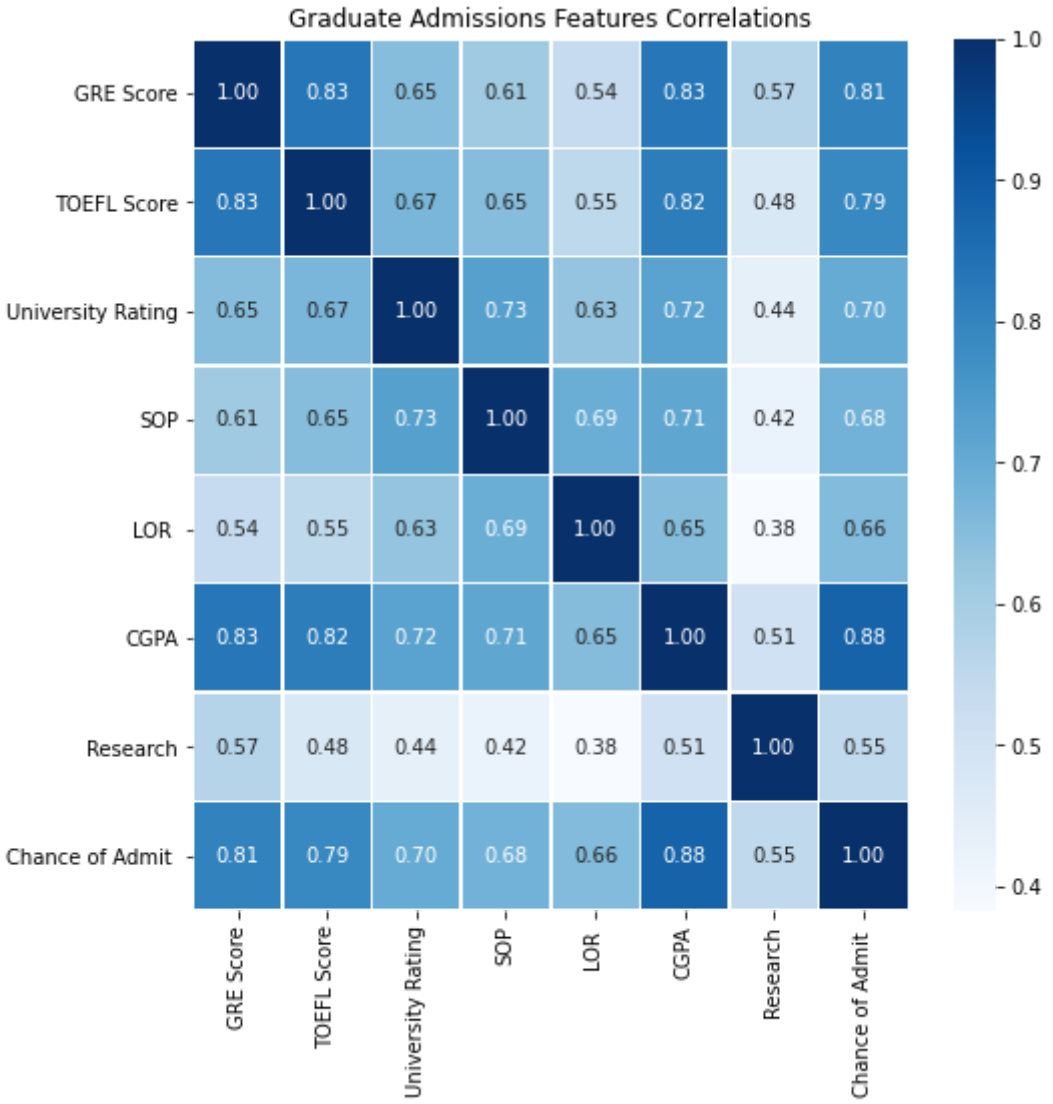
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.pairplot(dataset)
plt.show()
```



2.4 히트맵

In [7]:

```
fig, ax = plt.subplots(figsize=(8, 8))
sns.heatmap(dataset.corr(), linewidths=.5, annot=True, fmt=".2f", cmap='Blues')
plt.title('Graduate Admissions Features Correlations')
plt.show()
```



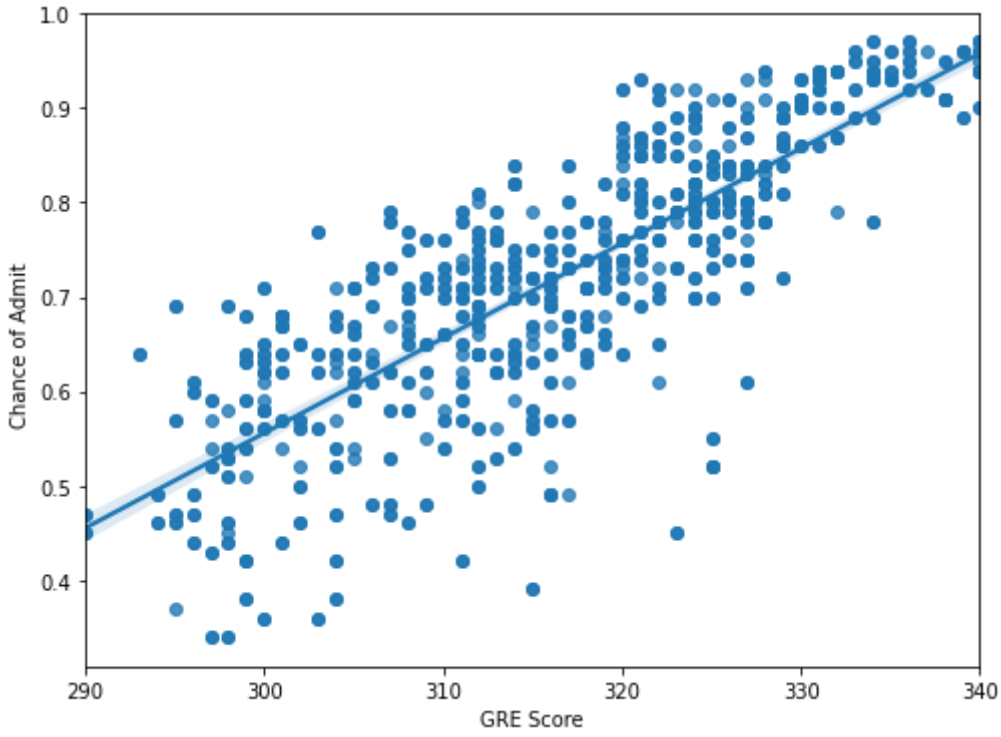
2.5 산포도와 단순 회귀선

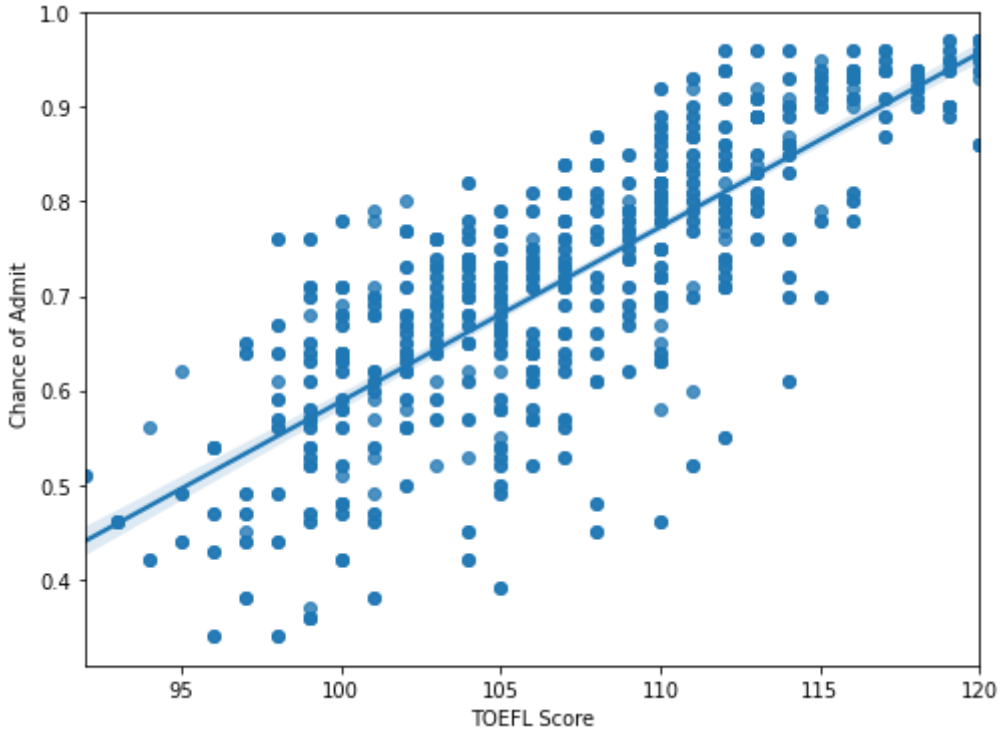
In [8]:

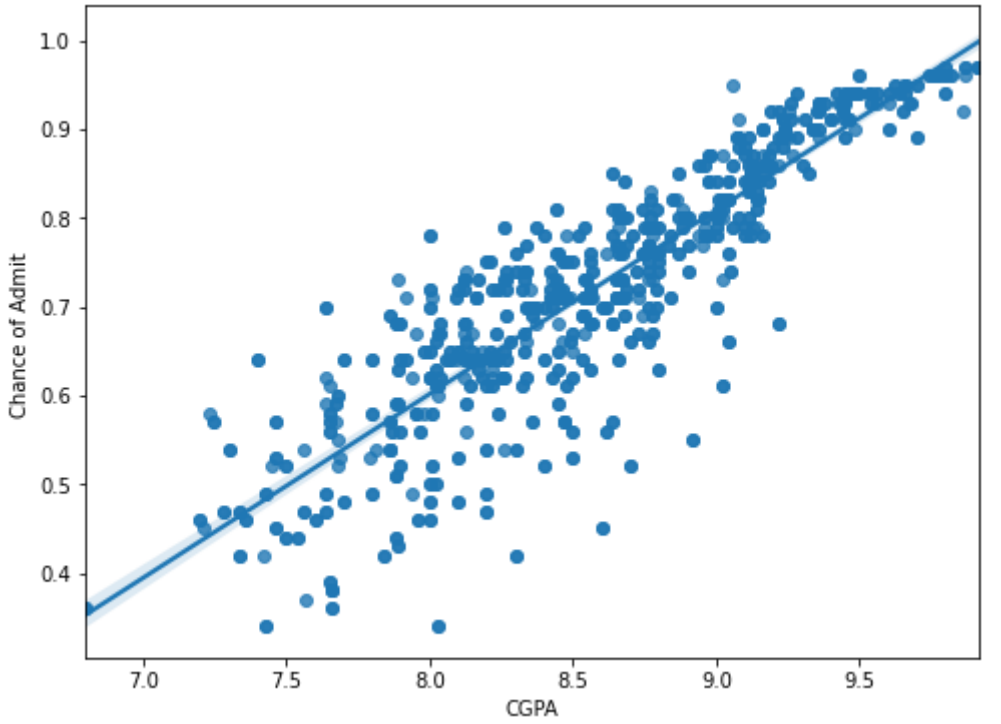
```
plt.subplots(figsize=(8,6))
sns.regplot(x="GRE Score", y="Chance of Admit ", data=dataset)
plt.subplots(figsize=(8,6))
sns.regplot(x="TOEFL Score", y="Chance of Admit ", data=dataset)
plt.subplots(figsize=(8,6))
sns.regplot(x="CGPA", y="Chance of Admit ", data=dataset)
```

Out[8]:

<AxesSubplot:xlabel='CGPA', ylabel='Chance of Admit '>




Chance of Admit
TOEFL Score
1.0
0.9
0.8
0.7
0.6
0.5
0.4
95
100
105
110
115
120


Chance of Admit
CGPA
1.0
0.9
0.8
0.7
0.6
0.5
0.4
7.0
7.5
8.0
8.5
9.0
9.5

3. 데이터 전처리

3.1 입력 및 타겟 데이터 추출

In [9]:

```
X = dataset.iloc[:,:-1].values
y = dataset.iloc[:,-1].values # Chance of Admit
```

입력 데이터에 상수 항에 대한 입력 1 추가

In [10]:

```
X = [[1.0] + list(row[:]) for row in X]
```

3.2 데이터셋 분리

In [11]:

```
import random
from scratch.machine_learning import train_test_split


random.seed(12)
X_train, X_test, y_train, y_test = train_test_split(X, y, 0.25)
print('train dataset :', len(X_train))
print('test dataset :', len(X_test))
```

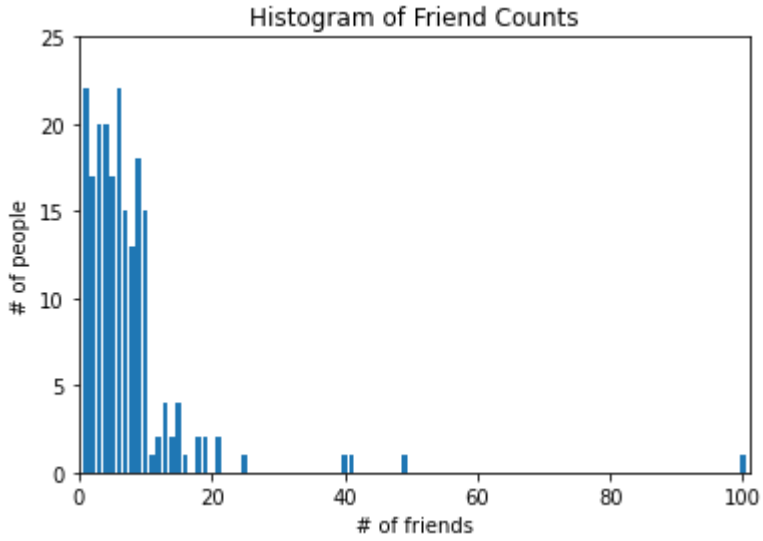
```
train dataset : 675
test dataset : 225
```

3.3 데이터 표준화 (Q2)

훈련 데이터의 평균과 표준 편차로 테스트 데이터를 표준화 하도록 normalization() 함수를 작성해 보시오.

In [12]:

```
from scratch.working_with_data import scale
from scratch.linear_algebra import vector_mean
from scratch.statistics import standard_deviation
from scratch.linear_algebra import Vector
from typing import List

def normalization(data: List[Vector],
                  means : Vector = None,
                  stdevs : Vector = None) -> List[Vector]:
    # your code
    dim = len(data[0])

    if means == None or stdevs == None:
        means, stdevs = scale(data)

    rescaled = [v[:] for v in data]

    for v in rescaled:
        for i in range(dim):
            if stdevs[i] > 0:
                v[i] = (v[i] - means[i]) / stdevs[i]

    return rescaled, means, stdevs
```



In [13]:

```
X_train_normed, X_train_means, X_train_stdevs = normalization(X_train)
X_test_normed, _, _  = normalization(X_test, X_train_means, X_train_stdevs)
```

4. 선형 회귀 분석 (Linear Regression)

4.1 예측 (Q3)

모델 예측 코드를 작성해 보시오.

In [14]:

```
# your code
from scratch.linear_algebra import dot, Vector

def predict(x: Vector, beta: Vector) -> float:
    return dot(x, beta)
```

4.2 손실 함수 (Q4)

손실 함수와 그래디언트를 작성해 보시오.

In [15]:

```
from typing import List

def error(x: Vector, y: float, beta: Vector) -> float:
    return predict(x, beta) - y
```

In [16]:

```
def squared_error(x: Vector, y: float, beta: Vector) -> float:
    return error(x, y, beta) ** 2
```

In [17]:

```
# your code
def sqerror_gradient(x: Vector, y: float, beta: Vector) -> Vector:
    err = error(x, y, beta)
    return [2 * err * x_i for x_i in x]
```

4.3 모델 훈련 (Q5)

최소 자승법

선형 회귀의 최소 자승법을 경사 하강법으로 구현하시오.

In [18]:

```
# your code
import random
import tqdm
from scratch.linear_algebra import vector_mean
from scratch.gradient_descent import gradient_step

def least_squares_fit(xs: List[Vector],
    ys: List[float],
    learning_rate: float = 0.001,
    num_steps: int = 1000,
    batch_size: int = 1) -> Vector:

    # Start with a random guess
    guess = [random.random() for _ in xs[0]]
    for _ in tqdm.trange(num_steps, desc="least squares fit"):
        for start in range(0, len(xs), batch_size):
            batch_xs = xs[start:start+batch_size]
            batch_ys = ys[start:start+batch_size]
            gradient = vector_mean([sqerror_gradient(x, y, guess) for x, y in zip(batch_xs, batch_y
            guess = gradient_step(guess, gradient, -learning_rate)

    return guess
```

In [19]:

```
learning_rate = 0.001

beta = least_squares_fit(X_train_normed, y_train, learning_rate, 5000, 50)
```

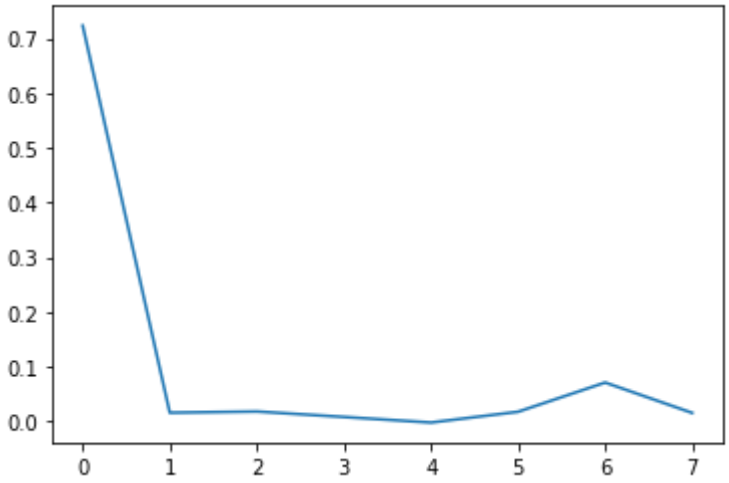
```
least squares fit: 100%|██████████| 5000/5000 [00:17<00:00, 281.21it/s]
```

β 확인

In [20]:

```
print("beta = ", beta)
plt.plot(beta)
plt.show()
```

```
beta =  [0.7230369263767936, 0.01589118373524194, 0.018101419703239086, 0.0081133607
37764688, -0.002073350176785202, 0.017606264994132734, 0.0710238708099809, 0.0155116
99194081541]
```



4.4 모델 테스트 (Q6)

테스트 데이터를 이용해서 예측을 해보고 SSE를 계산해 보시오.

In [21]:

```
def test(xs: List[Vector], ys: List[float], beta : Vector) -> float:

    # your code
    pred_y = []
    sse_list = []
    for i in xs:
        pred_y.append(dot(i, beta))
        sse_list.append(dot(i, beta))

    c = 0
    for j in ys:
        sse_list[c] -= j
        sse_list[c] = sse_list[c] ** 2
        c += 1

    SSE = sum(sse_list)

    return pred_y, SSE
```

In [22]:

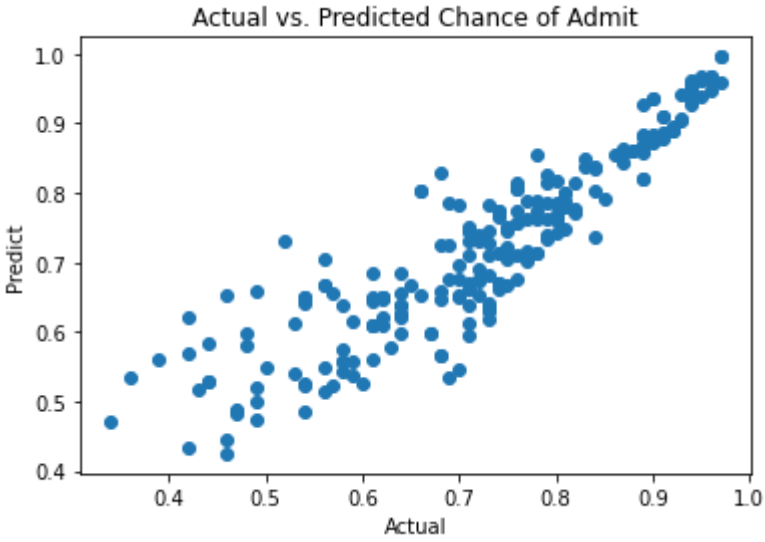
```
pred_y, SSE = test(X_test_normed, y_test, beta)
print("SSE = ", SSE)
```

```
SSE =  0.8671732654574079
```

실제 값과 예측 값의 상관 관계

In [23]:

```
plt.scatter(y_test, pred_y)
plt.title("Actual vs. Predicted Chance of Admit")
plt.xlabel("Actual")
plt.ylabel("Predict")
plt.show()
```



4.5 모델 적합도 (goodness-of-fit)

결정계수

In [24]:

```
from scratch.simple_linear_regression import total_sum_of_squares

def multiple_r_squared(xs: List[Vector], ys: Vector, beta: Vector) -> float:
    sum_of_squared_errors = sum(error(x, y, beta) ** 2
                                for x, y in zip(xs, ys))

    return 1.0 - sum_of_squared_errors / total_sum_of_squares(ys)
```

In [25]:

```
r_squared = multiple_r_squared(X_test_normed, y_test, beta)
print(r_squared)
```

0.8189390601208067