Data Mining

선형 회귀 과제

2021년 4월 29일

1. 데이터셋

대학원 입학 승인 예측

선형 회귀 모델로 외국인 학생이 대학원 입학이 승인될 확률을 예측해보자.

대학원 입학 데이터셋 (Graduate Admission 2)

속성	설명	타입
Serial No.	일련 번호	Discrete
GRE Scores	GRE 점수 [0,340]	Discrete
TOEFL Scores	TOEFL 점수 [0, 120]	Discrete
University Rating	대학 순위 [1,5]	Ordinal
SOP	학업 계획서 (Statement of Purpose) 점수	Ordinal
LOR	추천서 (Letter of Recommendation) 점수	Ordinal
CGPA	학부 GPA [0,10]	Continuous
Research	연구 경험 (0 또는 1)	Categorical
Chance of Admit	대학원 입학 승인 확률 [0,1]	Continuous

- 대학원 입학이 승인될 확률을 예측하기 위해 만든 데이터셋
- 총 9개 컬럼
- Version 1 : 400 예제 (Admission_Predict.csv)
- Version 2 : 500 예제 (Admission_Predict_Ver1.1.csv)

두개합침

https://www.kaggle.com/mohansacharya/graduate-admissions

패키지 설치

Seaborn 설치

```
!pip install seaborn
```

데이터 읽기

```
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)
```

- 두 데이터 파일을 읽어서 합침 (900개 예제)

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
140	141	329	110	2	4.0	3.0	9.15	1	0.84
273	274	312	99	1	1.0	1.5	8.01	1	0.52
131	132	303	105	5	5.0	4.5	8.65	0	0.77
224	225	305	105	2	3.0	2.0	8.23	0	0.67
361	362	334	116	4	4.0	3.5	9.54	1	0.93

Series No 컬럼 삭제 (Q1)

샘플 별로 유일한 값을 갖는 Serial No 컬럼은 학습에 방해가 되므로 삭제하시오.

```
# your code
```

Pandas 행 & 열 추가/삭제

새로운 이름으로 열 추가

```
dataset['USA’] = 1
```

drop 함수로 행 삭제

```
dataset = dataset.drop(‘Origin’, axis=1)
```

axis=0은 row, 1은 column

drop 함수로 열 삭제

```
dataset = dataset.drop(Row_Index, axis=0)
```

→ serial No 제외

axis=0은 row, 1은 column

2. 데이터 탐색

요약 통계량 확인

```
dataset.describe()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	900.000000	900.000000	900.000000	900.000000	900.00000	900.000000	900.000000	900.000000
mean	316.621111	107.288889	3.102222	3.385556	3.47000	8.586433	0.554444	0.722900
std	11.369700	6.073968	1.143048	0.997612	0.91319	0.600822	0.497303	0.141722
min	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.340000
25%	308.000000	103.000000	2.000000	2.500000	3.00000	8.140000	0.000000	0.640000
50%	317.000000	107.000000	3.000000	3.500000	3.50000	8.570000	1.000000	0.730000
75%	325.000000	112.000000	4.000000	4.000000	4.00000	9.052500	1.000000	0.822500
max	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.970000

누락 데이터 확인

누락된 데이터 보기

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

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

Pandas 중복/누락 데이터 처리

중복 행 조회

```
dataset.duplicated()
```

중복 행 제거

```
dataset.drop_duplicates()
```

누락 데이터 조회

```
dataset.isna()
```

누락 데이터 개수 합산

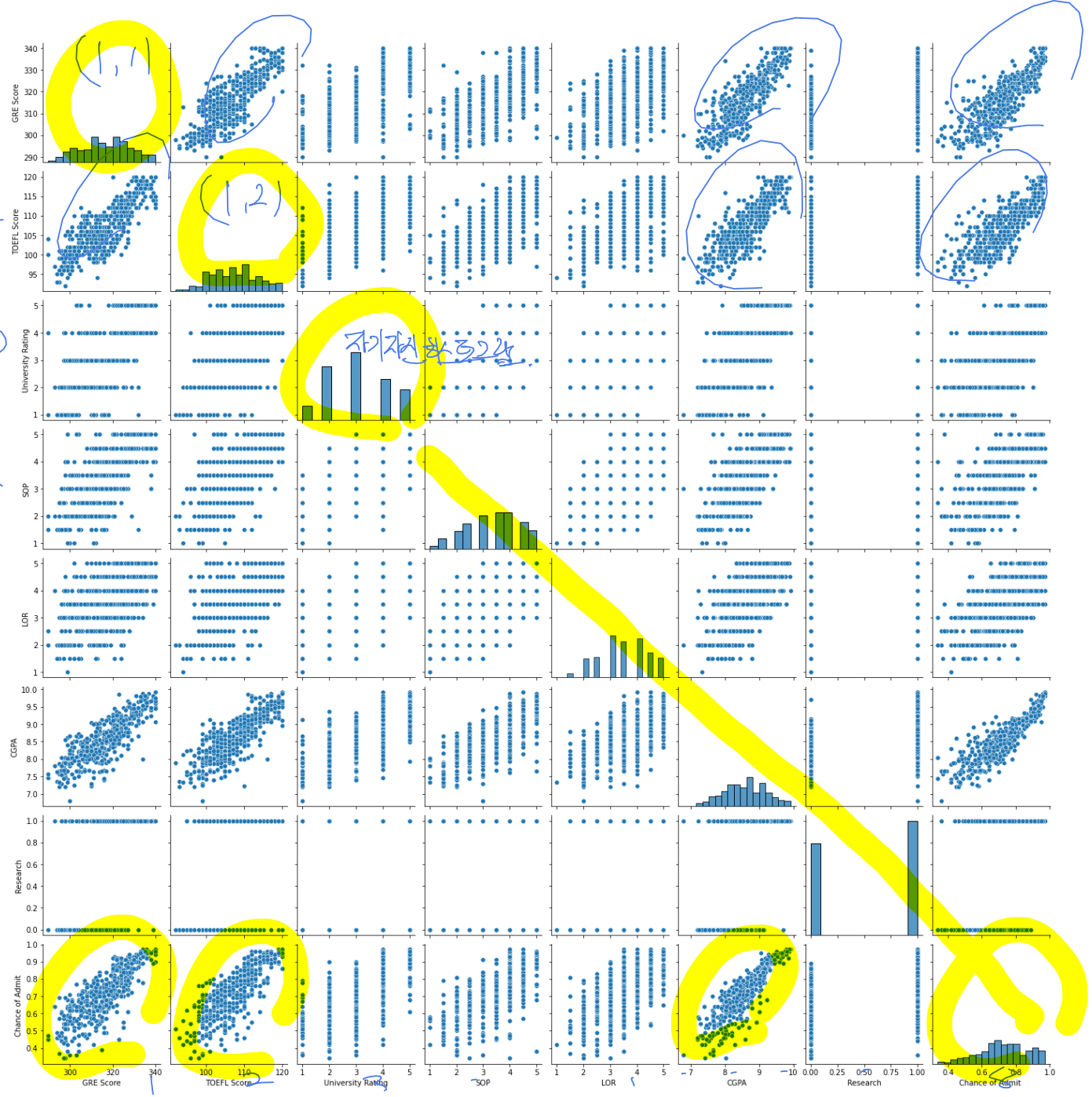
```
dataset.isna().sum()
```

누락 데이터 제거/채우기

```
dataset = dataset.dropna()
dataset = dataset.fillna(0)
```

산포도 행렬

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



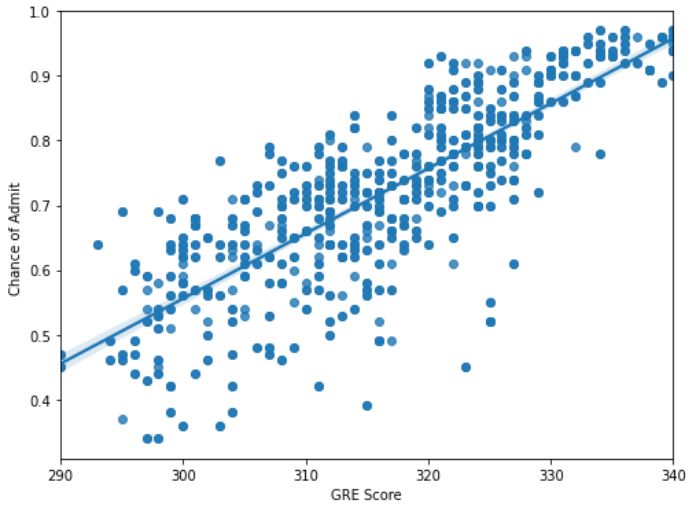
히트맵

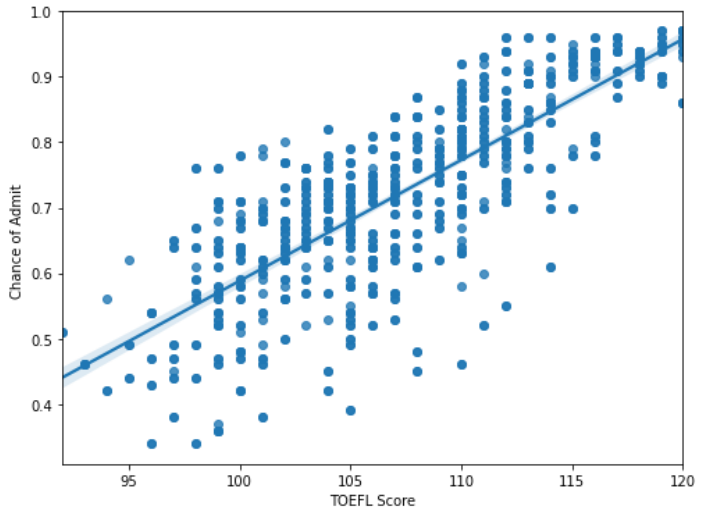
```
fig, ax = plt.subplots(figsize=(8, 8))
sns.heatmap(dataset.corr(), linewidths=.5, annot=True,
 fmt=".2f", cmap='Blues')
plt.title('Graduate Admissions Features Correlations')
plt.show()
```

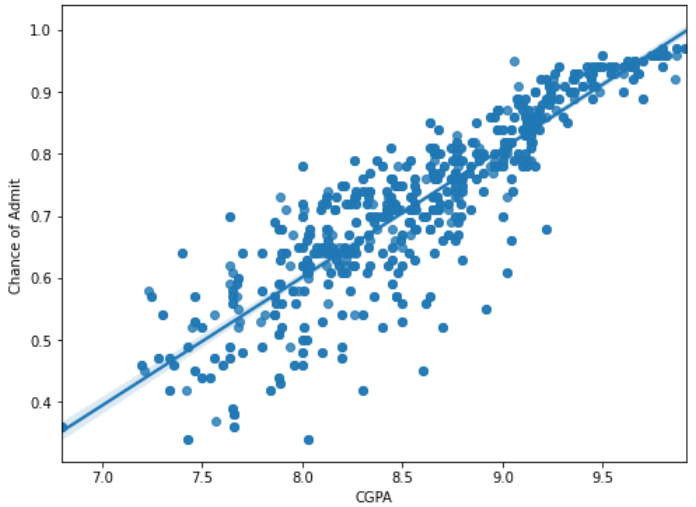
변수들의 상관관계를 색깔로 알아볼

산포도와 단순 회귀선

```
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)
```







3. 데이터 전처리

입력 및 타겟 데이터 추출

데이터프레임에서 데이터 추출

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

입력 데이터에 상수 항에 대한 입력 1 추가

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

데이터셋 분리

```
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

데이터 표준화 (Q2)

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

scale과 rescale을 이용.

표준 정규 분포로 정규화

```
from scratch.working_with_data import scale

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

    return rescaled, means, stdevs
```

훈련 데이터 및 테스트 데이터 표준화

means, stdvs가 None일 경우 작동구현

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

↳ 들어올경우 모든 data!

4. 선형 회귀

예측 (Q3)

모델 예측 코드를 작성해 보시오.

모델 예측

```
# your code
```

16장 17절 실제예측 코드 보기

손실 함수 (Q4)

손실 함수와 그래디언트를 구현해 보시오.

잔차

error 발생.

```
# your code
```

SSE

```
# your code
```

β에 대해 그래디언트 계산

```
# your code
```

모델 훈련 (Q5)

선형 회귀의 최소 자승법을 경사 하강법으로 구현하시오.

```
# your code
```

모델 훈련

선형 회귀

```
learning_rate = 0.001

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

계수 출력

```
print("beta = ", beta)
```

beta = [0.7230369263742981, 0.0158911832078850 18, 0.0181014198098116 25, 0.0081133606813682 24, -0.0020733502459507765, 0.017606264910547 65, 0.0710238713464783 6, 0.0155116992690851 75]

모델 테스트 (Q6)

테스트 데이터를 이용해서 모델 예측을 해보고 SSE를 계산해 보시오.

테스트 데이터로 예측 및 SSE 계산

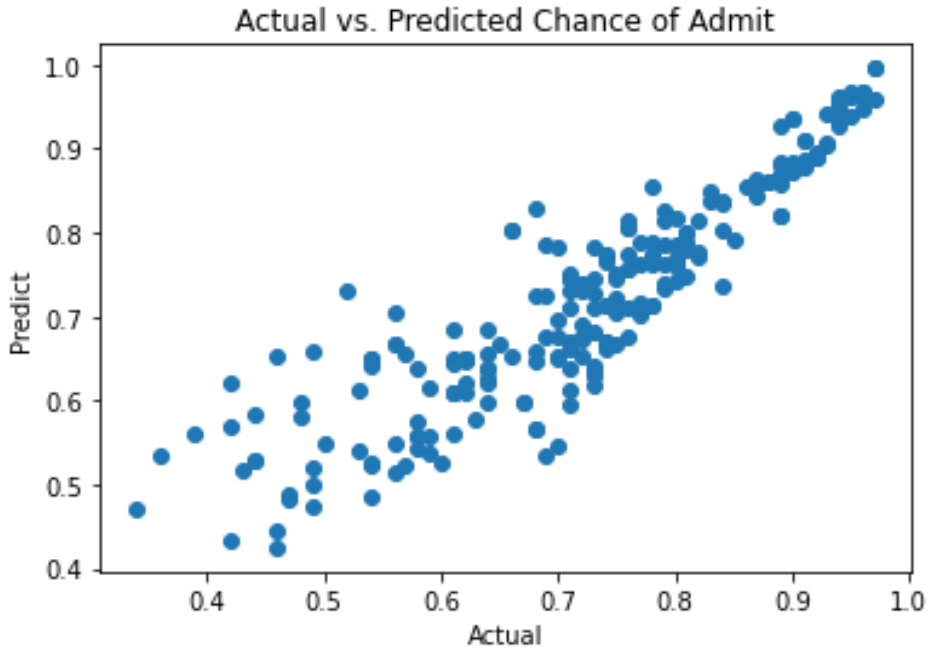
```
def test(xs: List[Vector], ys: List[float], beta : Vector) -> float:

    # your code

    return pred_y, SSE
```

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

SSE = 0.8671732654574079

예측 결과와 실제 값의 상관 관계

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



모델 적합도 (goodness-of-fit)

결정계수

```
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)
```

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

0.8189390601208067

Thank you!