

# 수강신청 결과 예측과 웹 구현

웹을 통한 마일리지 수강 신청 가이드라인 제공

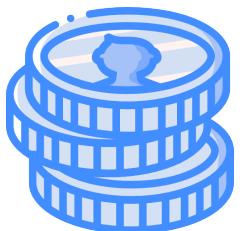


안주영 | 이용하 | 정현우 | 최민태 | 한승희

# 목차

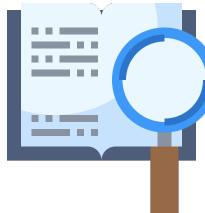


서론



목표

본론



크롤링



머신러닝



DB



웹

결론

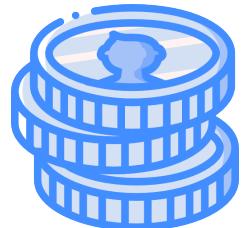


결론

# 목차



서론



목표

본론



크롤링



머신러닝

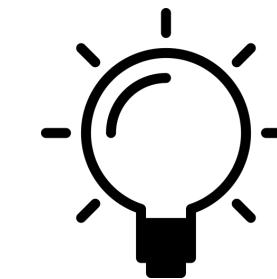


DB



웹

결론



결론

# 서론



## 연세대학교 수강신청 제도



### Y-CES 마일리지 수강신청 제도

“마일리지 할당” 및 대기 순번제를 이용한 온라인 수강신청 방법

2015년 06월 특허 등록 및 공고

마일리지 제도는 수강신청 과열과 서버 다운 등의 문제를 해결하기 위한 연세대학교의 새로운 수강신청 제도이다.

# 본론

## Step 3) 머신러닝

### 4. 모델 학습

#### 클래스 생성

##### CRpredict

```
class CRpredict:  
    def __init__(self, origin_df, course_code, instructor, quota, max_mile  
age,  
                 mileage, major_quota, second_major, grade_quota_1, grade_  
quota_2,  
                 grade_quota_3, grade_quota_4, enrolled_courses, graduatio  
n, major, double_major,  
                 first_enroll, credits_rate, previous_credits_rate, grad  
e):
```

추후의 머신러닝을 위해 “CRpredict”라는 클래스를 자체적으로 만들었다.

클래스를 만들어 놓으면, 계속 모델을 사용하기 편리하고  
parameter 조정 등을 유연하게 할 수 있다.

## 서론



## 연세대학교 수강신청 제도



## Y-CES 마일리지 수강신청 제도



학생에게 학기 별로 일정량의 마일리지를 제공하고, 학생은 개별 과목 수강을 희망하는 정도만큼 과목 별로 마일리지를 배분하는 제도이다.

## 서론



## 연세대학교 수강신청 제도



## Y-CES 마일리지 수강신청 제도

국내 다수의 대학들은 **선착순 방식**을 표방하는 기준안을 수강신청 제도로 채택하고 있다. 기준안이 자체적이고 고질적인 **문제**를 내포하고 있음에도 이를 대체할 만한 **표 족한 방법**이 없기 때문이다.

수강신청의 새로운 패러다임



근본적인 문제점 완전한 해결



# 서론

## 연세대학교 수강신청 제도

여전히 남아있는 고질적인 문제는 잠시 뒤로 하자.

**“수강신청은 매학기 찾아온다.”**

따라서 주어진 상황에서 최고의 선택을 하기 위한 **가이드라인**이 필요하다.



마일리지 수강신청 데이터를 이용한 머신러닝 모델로 마일리지 수강신청 **합격 확률** (합 · 불 여부)를 예측하고, **최적의 마일리지 배분** 의사결정을 돋는다.

# 서론

## 예측 목표

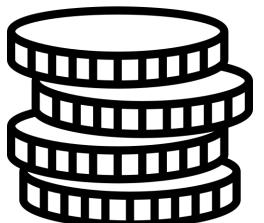
수강신청 데이터가 적재된 DB 서버와 연동된 Web 상에서 직접 강의를 선택하여 자신의 정보에 따라 예측된 마일리지 수강 신청 성공 여부를 볼 수 있도록 구현



# 목차

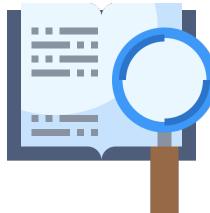


서론



목표

본론



크롤링



DB

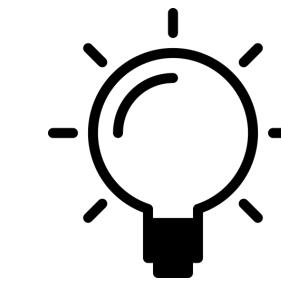


머신러닝



웹

결론



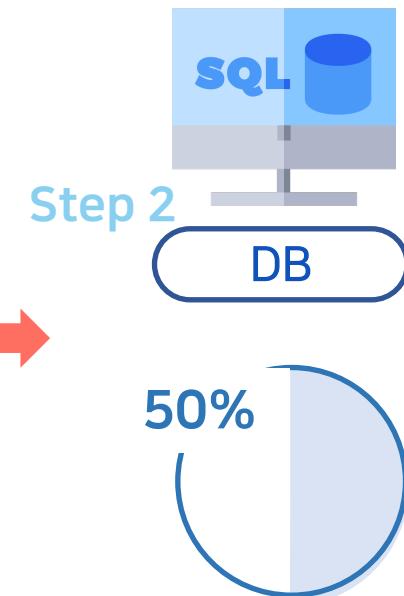
결론

# 본론

## 과정



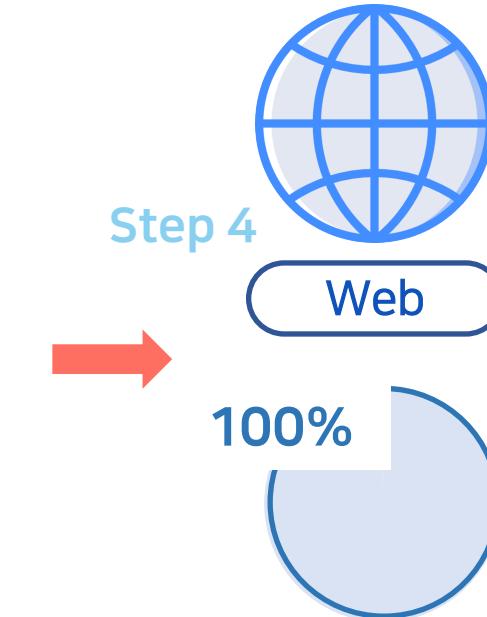
1. 연세포탈에서 **수강 신청 정보 크롤링**



2. MySQL DB 구축  
3. 크롤링한 데이터 MySQL 서버에 넣기



4. **머신러닝**을 통해  
마일리지 우선순위  
정보에 따른 **예측 모델링**



5. **Flask**를 통해 DB 서버와  
연동된 **Web** 구현  
6. 사용자가 입력한 정보에  
따른 **합·불 여부** 출력

# 본론

## 프레임워크



Selenium

BeautifulSoup

BeautifulSoup



MySQL



pyMySQL



Scikit-learn



XGBoost



Flask



HTML

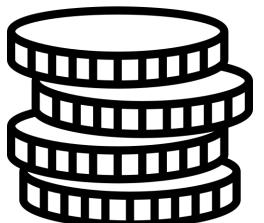


CSS

# 목차

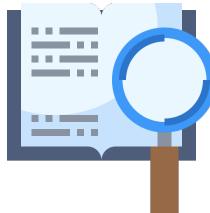


서론



목표

본론



크롤링



DB

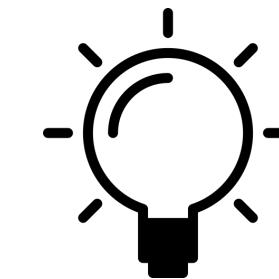


머신러닝



웹

결론



결론

# 본론

Step 1) 크롤링

## 연세포탈 수강편람 크롤링

연세포탈의 [수강편람]에서

-마일리지 수강신청결과(수업 별)



meta\_data

-마일리지 수강신청결과(개인 별)



mile\_data

구분하여 크롤링

# 본론

## Step 1) 크롤링

### 연세포탈 수강편람 크롤링

#### 수강편람 조회

\* 마일리지 수강신청결과

학정번호-분반-실습	과목명	학점	담당교수	강의시간	강의실	정원	참여 인원	선정사 정원 (2전공포함)	학년별정원				교환학생 기능여부	Mile
									1	2	3	4		
CSI2102-01-00	객체지향프로그래밍	3	김선현	월6/수2,3	공A528/공D504	72	76	36 (Y)	0	0	0	0	X	

\* 정원, 전공자정원, 학년별정원은 수강신청기간동안 적용된 값이며, 추가수강신청 및 수강변경 기간에 적용되는 학년별 정원 등의 값(잔여석)은 과목 개설학과의 결정에 따라 변경될 수 있으므로 개설과목에서 검색하여 확인하시기 바랍니다.

마일리지 순위 결과

순위	마일리지	전공자/복수전공자 (전공자정원포함여부)	신청 과목수	졸업신청	초수강여부	총이수학점/ 졸업이수학점	직전학기이수학점/ 학기당수강학점	학년	수강여부	비고
1	36	N (N)	6	N	Y	0.7592	0.7894	4	O	
2	36	N (N)	6	N	Y	0.7571	1.0000	4	O	
3	36	N (N)	6	N	Y	0.5592	0.9473	3	O	
4	36	N (N)	6	N	Y	0.5333	0.8157	3	O	
5	36	N (N)	6	N	Y	0.3714	0.9473	2	O	
6	36	N (N)	6	N	Y	0.3185	1.0000	2	O	
7	36	N (N)	6	N	Y	0.3000	1.0000	2	O	
8	36	N (N)	6	N	Y	0.2666	0.9473	2	O	
9	36	N (N)	6	N	Y	0.2592	0.9736	2	O	
10	35	N (N)	6	N	Y	0.5777	0.7894	3	O	
11	35	N (N)	6	N	Y	0.5037	0.8421	3	O	
12	32	N (N)	6	N	Y	0.5285	0.8157	3	O	
13	32	N (N)	5	N	Y	0.5148	0.7894	3	O	
14	32	N (N)	5	N	Y	0.3412	1.0000	2	O	
15	30	Y (Y)	6	N	Y	0.6629	0.9473	3	O	
16	30	Y (Y)	6	N	Y	0.6333	1.0000	3	O	
17	30	N (N)	6	N	Y	0.5464	1.0000	3	O	
18	28	N (N)	6	N	Y	0.2407	0.8421	2	O	
19	27	Y (Y)	5	N	N	0.8000	0.9473	4	O	
20	25	N (N)	6	N	Y	0.3015	1.0000	2	O	
21	24	N (N)	6	N	Y	0.4370	1.0000	3	O	
22	24	N (N)	5	N	Y	0.4535	0.6578	3	O	
23	24	N (N)	3	N	Y	0.5079	1.0000	3	O	

meta\_data

mile\_data

# 본론

## Step 1) 크롤링

### 연세포탈 수강편람 크롤링

#### 수강편람 조회

\* 마일리지 수강신청결과

학정번호-분반-실습	과목명	학점	담당교수	강의시간	강의실	정원	참여 인원	선정사 정원 (2전공포함)	학년별정원				교환학생 가능여부	Mileage
									1	2	3	4		
CSI2102-01-00	객체지향프로그래밍	3	김선현	월6/수2,3	공A528/공D504	72	76	36(Y)	0	0	0	0	X	

\* 정원, 전공자정원, 학년별정원은 수강신청기간동안 적용된 값이며, 추가수강신청 및 수강변경 기간에 적용되는 학년별 정원 등의 값(잔여석)은 과목 개설학과의 결정에 따라 변경될 수 있으므로 개설과목에서 검색하여 확인하시기 바랍니다.

마일리지 순위 결과

순위	마일리지	전공자/복수전공자 (전공자정원포함여부)	신청 과목수	졸업신청	초수강여부	총이수학점/ 졸업이수학점	직전학기이수학점/ 학기당수강학점	학년	수강여부	비고
1	36	N (N)	6	N	Y	0.7592	0.7894	4	0	
2	36	N (N)	6	N	Y	0.7571	1.0000	4	0	
3	36	N (N)	6	N	Y	0.5592	0.9473	3	0	
4	36	N (N)	6	N	Y	0.5333	0.8157	3	0	
5	36	N (N)	6	N	Y	0.3714	0.9473	2	0	
6	36	N (N)	6	N	Y	0.3185	1.0000	2	0	
7	36	N (N)	6	N	Y	0.3000	1.0000	2	0	
8	36	N (N)	6	N	Y	0.2666	0.9473	2	0	
9	36	N (N)	6	N	Y	0.2592	0.9736	2	0	
10	35	N (N)	6	N	Y	0.5777	0.7894	3	0	
11	35	N (N)	6	N	Y	0.5037	0.8421	3	0	
12	32	N (N)	6	N	Y	0.5285	0.8157	3	0	
13	32	N (N)	5	N	Y	0.5148	0.7894	3	0	
14	32	N (N)	5	N	Y	0.3412	1.0000	2	0	
15	30	Y (Y)	6	N	Y	0.6629	0.9473	3	0	
16	30	Y (Y)	6	N	Y	0.6333	1.0000	3	0	
17	30	N (N)	6	N	Y	0.5464	1.0000	3	0	
18	28	N (N)	6	N	Y	0.2407	0.8421	2	0	
19	27	Y (Y)	5	N	N	0.8000	0.9473	4	0	
20	25	N (N)	6	N	Y	0.3015	1.0000	2	0	
21	24	N (N)	6	N	Y	0.4370	1.0000	3	0	
22	24	N (N)	5	N	Y	0.4535	0.6578	3	0	
23	24	N (N)	3	N	Y	0.5079	1.0000	3	0	

meta\_data

학정번호-분반-실습 / 과목명 / 학점 / 담당교수 / 강의시간 /  
강의실 / 정원 / 참여인원 / 전공자 정원 / 2전공 포함 / 학년별 정원 /  
교환학생 가능여부 / Max Mileage / 마일리지 최소 · 최대 · 평균

mile\_data

순위 / 마일리지 / 전공자 · 복수전공자 여부 / 신청 과목수 /  
졸업신청 / 초수강여부 / 총이수학점 비율 / 직전학기이수학점  
비율 / 학년 / 수강여부(수강신청 결과)

본론



## Step 1) 크롤링

수강편람 조회

# 연세포탈 수강편람 크롤링

# 8개 학기 수강신청 데이터 크롤링 완료!

10	35	N (N)	6	N	Y	0.5777	0.7894	3	O
11	35	N (N)	6	N	Y	0.5037	0.8421	3	O
12	32	N (N)	6	N	Y	0.5285	0.8157	3	O
13	32	N (N)	5	N	Y	0.5148	0.7894	3	O
14	32	N (N)	5	N	Y	0.3412	1.0000	2	O
15	30	Y (Y)	6	N	Y	0.6629	0.9473	3	O
16	30	Y (Y)	6	N	Y	0.6333	1.0000	3	O
17	30	N (N)	6	N	Y	0.5464	1.0000	3	O
18	28	N (N)	6	N	Y	0.2407	0.8421	2	O
19	27	Y (Y)	5	N	N	0.8000	0.9473	4	O
20	25	N (N)	6	N	Y	0.3015	1.0000	2	O
21	24	N (N)	6	N	Y	0.4370	1.0000	3	O
22	24	N (N)	5	N	Y	0.4535	0.6578	3	O
23	24	N (N)	3	N	Y	0.5079	1.0000	3	O



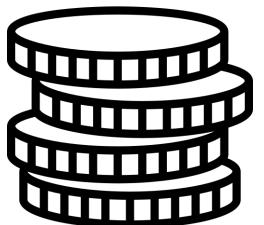
# mile\_data

순위 / 마일리지 / 전공자 · 복수전공자 여부 / 신청 과목수 /  
졸업신청 / 초수강여부 / 총이수학점 비율 / 직전학기이수학점  
비율 / 학년 / 수강여부(수강신청 결과)

# 목차



서론



목표

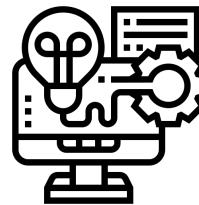
본론



크롤링



DB

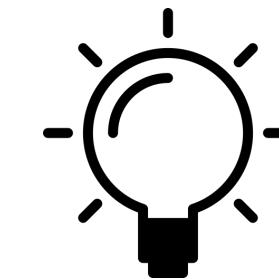


머신러닝



웹

결론



결론

# 본론

Step 2) DB

## DB 구축

크롤링한 **데이터를 적재하기** 위해 대표적인 관계형  
데이터베이스 관리 시스템(RDBMS)인 **MySQL DB**를 개설하였다.



- 수강신청 데이터와 같은 **구조화된 데이터**에 강점
- 파이썬 스크립트 등을 위한 **간편한 API** 제공

## 본론

Step 2) DB

## DB 데이터 전송

크롤링한 수강신청 데이터를 **pyMySQL** 모듈을 활용하여 **MySQL DB 서버**에 전송한다.



\***pyMySQL** : MySQL을 Python에서 사용할 수 있는 라이브러리  
Table Create, Insert query 등을 시행할 수 있다.

## 본론

Step 2) DB

## DB 데이터 전송

# 8개 학기 수강신청 데이터 DB 전송 완료!

python



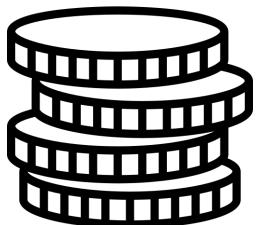
MySQL®

\***pyMySQL** : MySQL을 Python에서 사용할 수 있는 라이브러리  
Table Create, Insert query 등을 시행할 수 있다.

# 목차



서론



목표

본론



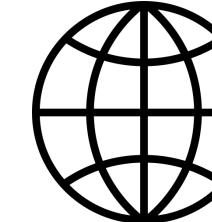
크롤링



DB

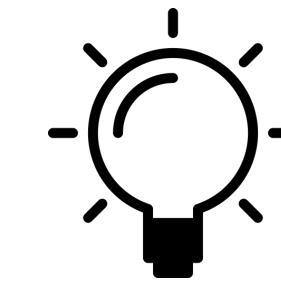


머신러닝



웹

결론



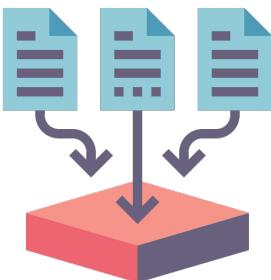
결론

# 본론

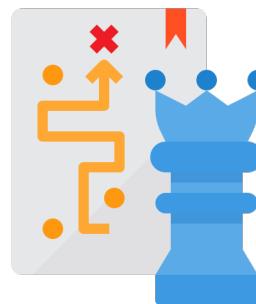
## Step 3) 머신러닝

### 머신러닝 과정

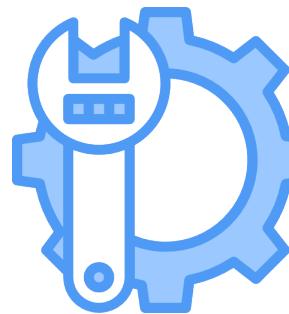
수강신청 예측 머신러닝은 총 **5단계**의 과정로 구성되어 있다.



① 데이터 수집



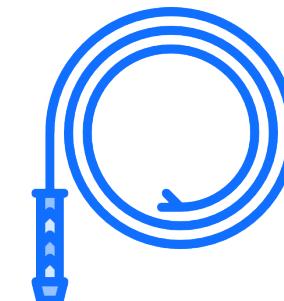
② 데이터 전처리



③ FE



④ 모델 학습



⑤ 최적 성능 도출



# 본론

## Step 3) 머신러닝

### 1. 데이터 수집(크롤링)

앞선 크롤링으로 가공할 데이터를 준비 완료하였다.

**meta\_data**

	hyhg	course_code	instructor	quota	participants	major_quota	second_major	grad
0	20161	OMA1001-01-00	김종철,홍기원	63	40	0	0	
1	20161	OMA1002-01-00	우환식	63	98	0	0	
2	20161	YCA1101-10-00	조재국	53	43	0	0	
3	20161	YCA1101-11-00	박정세	24	48	0	0	
4	20161	YCA1101-12-00	민경식	50	14	50	0	
...	...	...	...	...	...	...	...	
26581	20192	KLI1002-01-00	김성숙	80	45	0	0	
26582	20192	KLI1003-01-00	김성숙	50	32	0	0	
26583	20192	KLI1004-01-00	김성숙	40	24	0	0	
26584	20192	KLI1005-01-00	김성숙	15	9	0	0	
26585	20192	KLI1006-01-00	김성숙	15	5	0	0	

26,586 rows x 13 columns

**Mile\_data**

	enrolled	hyhg	mileage	major	double	enrolled_courses	graduation	first_enroll	c
0	O	20161	36	0	0	6	0	1	
1	O	20161	36	0	0	6	0	1	
2	O	20161	36	0	0	6	0	1	
3	O	20161	36	0	0	5	0	1	
4	O	20161	36	0	0	5	0	1	
...	...	...	...	...	...	...	...	...	...
985316	O	20192	1	0	0	5	0	1	
985317	O	20192	1	0	0	5	0	1	
985318	O	20192	1	0	0	4	0	1	
985319	O	20192	1	0	0	4	0	1	
985320	O	20192	1	0	0	3	0	1	

985,321 rows x 15 columns

# 본론

Step 3) 머신러닝

## 2. 전처리

1) 필요한 column만 남긴다.

**meta\_data**

### (1) 남긴 columns

학정번호 / 과목명 / 담당교수 / 정원 / 참여인원 / 전공자  
정원 / 학년별 정원 / Max Mileage / 마일리지 최대

### (2) 삭제한 columns

학점 / 강의시간 / 강의실 / 2전공 포함 / 교환학생 가능여부 / 마일리지 최소 · 평균

# 본론

Step 3) 머신러닝

## 2. 전처리

1) 필요한 column만 남긴다.

**mile\_data**

### (1) 남긴 columns

수강성공여부(수강신청 결과) / 마일리지 / 전공자 여부 / 복수전공자 여부 /  
신청 과목수 / 졸업신청 / 학년 / 총이수학점 비율 / 직전학기이수학점 비율

### (2) 추가한 columns

+ 연도 · 학기 / 학정번호 / 분반 / 실습분반

### (3) 삭제한 columns

순위 / ~~초수강여부~~

# 본론

\*FE : Feature Engineering

## Step 3) 머신러닝

### 3. FE

#### 1) 학정번호 완성

학정번호 + 분반 + 실습 분반 3개의 columns를  
연결시켜 실제와 같은 학정번호를 완성하였다.

**hakno**

---

OMA1001

기존의 학정번호



**course\_code**

---

OMA1001-  
01-00

새로운 학정번호

# 본론

Step 3) 머신러닝

## 3. FE

### 2) 학년 합치기

(1) 4학년 이상은 모두 4학년으로 통합

4,5,6학년 → 4학년

(2) 1학년 이하는 모두 1학년으로 통합

0,1학년 → 1학년

\*학년별 정원이 0/0/0/0인 강의는 **NaN**으로 바꾸었다.



예를 들어, 학년별 정원이 0/13/13/13인 강의와 0/0/0/0인  
강의는 그 성격이 다르다고 판단하였기 때문이다.

# 본론

\*Dummy 변수 : 범주형 변수를 "연속형 변수"화 시킨 것

## Step 3) 머신러닝

### 3. FE

#### 3) Dummy 변환

##### (1) 범주형(Categorical) 변수 → 0, 1로 변환

전공자 여부 / 복수전공자 여부 / 졸업신청 여부 / 초수강 여부 / 수강성공 여부

$$X, O \rightarrow 0, 1$$

##### (2) 학년 → Dummy화

grade\_1 / grade\_2 / grade\_3 / grade\_4 columns 생성

# 본론

Step 3) 머신러닝

## 3. FE

### 4) Merge

중복값 제거 후,

“학정번호” & “연도학기” 기준으로 Merge

# 본론

## Step 3) 머신러닝

### 3. FE

#### 최종 데이터셋

enrolled	hyhg	course_code	instructor	quota	max_mileage	mileage	major_quota	second_major	grade_quota_1	grade_quota_2	grade_quota_3	grade_quota_4	grade_quota_5	grade_quota_6	grade_quota_7	grade_quota_8	grade_quota_9	grade_quota_10	grade_quota_11	grade_quota_12	graduation	major	double	first_enroll	credits_rate	previous_credits_rate	grade_1	grade_2	grade_3	grade_4	
0	1	20161	OMA1001	김종철,홍기원	63.0	36.0	36	NaN	0.0	0.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	6	0	0	0	1	0.7925	0.9473	0	0	0	1
1	1	20161	OMA1001	김종철,홍기원	63.0	36.0	36	NaN	0.0	0.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	6	0	0	0	1	0.7857	0.7142	0	0	0	1
2	1	20161	OMA1001	김종철,홍기원	63.0	36.0	36	NaN	0.0	0.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	6	0	0	0	1	0.7428	0.8947	0	0	0	1
3	1	20161	OMA1001	김종철,홍기원	63.0	36.0	36	NaN	0.0	0.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	5	0	0	0	1	0.7642	0.8421	0	0	0	1
4	1	20161	OMA1001	김종철,홍기원	63.0	36.0	36	NaN	0.0	0.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	5	0	0	0	1	0.4337	0.4210	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...		
985498	1	20192	KLI1006	김성숙	15.0	36.0	1	NaN	0.0	NaN	NaN	5	0	0	0	1	0.7500	0.9375	0	0	0	1									
985499	1	20192	KLI1006	김성숙	15.0	36.0	1	NaN	0.0	NaN	NaN	5	0	0	0	1	0.2500	0.9375	0	1	0	0									
985500	1	20192	KLI1006	김성숙	15.0	36.0	1	NaN	0.0	NaN	NaN	4	0	0	0	1	0.7500	0.9375	0	0	0	1									
985501	1	20192	KLI1006	김성숙	15.0	36.0	1	NaN	0.0	NaN	NaN	4	0	0	0	1	0.7500	0.8750	0	0	0	1									
985502	1	20192	KLI1006	김성숙	15.0	36.0	1	NaN	0.0	NaN	NaN	3	0	0	0	1	0.2500	0.9375	0	1	0	0									

```
final_df['enrolled'].value_counts()
```

```
O    744699
X    240804
Name: enrolled, dtype: int64
```

985503 rows × 26 columns

-행 : 약 100만 개  
-열(Feature 수) : 26개

-수강신청 성공/실패 비율 : 약 3:1  
∴ 수강신청 성공 확률 : 약 75%

# 본론

Step 3) 머신러닝

## 4. 모델 학습

모델링 전, 2가지 갈래의 **고려사항** 발생

“ 예측 시 알 수 없는 정보들이 있다. ”



**목표** : 실제 수강신청 시점에서의 수강생의 정보를 입력 받아 예측 결과를 반환한다.



**문제** : 과거 데이터는 수강신청이 끝난 후 알 수 있는 정보인 “참여인원”과 “최대 마일리지”를 포함하고 있지만, 이는 **신청 당시에는 알 수 없는 정보**다.

# 본론

Step 3) 머신러닝

## 4. 모델 학습

따라서,

- A : 모든 columns 포함
- B : 예측 시 알 수 없는 columns 제거

데이터셋을 A / B 두 가지로 분류하여  
A / B 두 개의 모델링 동시에 진행

# 본론

Step 3) 머신러닝

## 4. 모델 학습

학습할 모델

일반적 · 경험적으로 가장 우수한 세 가지 **Classifier**를 선택하였다.

(1) XGBoost

(2) LightGBM

(3) Random Forest

# 본론

Step 3) 머신러닝

## 4. 모델 학습

학습할 모델

일반적 · 경험적으로 가장 우수한 세 가지 **Classifier**를 선택하였다.

(1) XGBoost

(2) LightGBM

(3) Random Forest

→ NaN값 존재로 학습 실패

# 본론

Step 3) 머신러닝

## 4. 모델 학습

### (1) XGBoost

A 데이터셋  
1<sup>st</sup> Model

```
x = df_a[list(df_a)[1:]]
y = df_a['enrolled']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

학습 진행하기

```
# 시간 설정
import time
start = time.time()

## Xgboost Classifier
from xgboost import XGBClassifier, plot_importance

model = XGBClassifier()
model.fit(X_train, y_train)
```

\*A데이터셋 : 모든 columns 포함

기본 설정

train set : 0.75 / test set : 0.25

결과

소요 시간

56.448241233825684

→ 약 56초 소요

성능

0.8571064093815509

→ 약 86%의 성능  
(F1 score)

## 본론

\*Grid : 격자무늬



## Step 3) 머신러닝

## 4. 모델 학습

## (1) XGBoost



## GridSearchCV

여러 경우의 **hyperparameter**가 있을 때, 하나하나 수동으로 테스트하기 **불편**하므로, 테스트할 parameter만 설정하면 **자동으로 조합**하여 결과를 내주는 method다.



## CV(Cross Validation)

**교차 검증**, 데이터를 **여러 구간**으로 나누어 **샘플링**을 하여 예측/분류 모델의 **과적합**을 방지하고, 보다 일반적인 모집단에 적용 가능한지 여부를 확인하기 위한 샘플링 방법  
ex) **CV = 4**라면, 각기 다른 샘플링을 통해 **4번의 검증**을 거친다.

# 본론

✓ CV가 4라면, 각 모델마다 4번의 CV를 시행하는 것이  
므로 이 경우에는 총  $4 \times 6 = 24$ 개의 모델을 검증한다.

## Step 3) 머신러닝

### 4. 모델 학습

#### (1) XGBoost

##### 1<sup>st</sup> GridSearchCV

목표 parameters  
"gamma" / "max\_depth"

```
xgb_params_1 = {
    'max_depth' : [3,5,7],
    'gamma' : [0, 0.1],
}
```

최적의 parameters  
"gamma" : 0 / "max\_depth" : 7

각 3개, 2개의 parameter 설정  
→  $3 \times 2 = 6$ 개의 조합

#### 성능 소요 시간

```
Best Score : 0.9456649395238692
Best Params : {'gamma': 0, 'max_depth': 7}
얼마나 걸렸뇨? : 1992.2222027778625
```

#### 결과

약 30분 소요

약 95%의 성능  
(F1 score)

# 본론

\*앞서 구한 최적의 gamma parameter를 적용

## Step 3) 머신러닝

### 4. 모델 학습

#### (1) XGBoost

#### 2<sup>nd</sup> GridSearchCV

##### 목표 parameters

"max\_depth" / "subsample"

```
xgb_params_2 = {
    'max_depth' : [7,8,9],
    'subsample' : [0.5, 0.7],
}
```

##### 최적의 parameters

"max\_depth" : 9 / "subsample" : 0.7

각 3개, 2개의 parameter 설정  
 $\rightarrow 3 \times 2 = 6$ 개의 조합

##### 성능 소요 시간

```
Best Score : 0.9558779238476712
Best Params : {'max_depth': 9, 'subsample': 0.7}
얼마나 걸렸뇨? : 4031.8685393333435
```

##### 결과

약 70분 소요

약 96%의 성능  
(F1 score)

# 본론

## Step 3) 머신러닝

### 4. 모델 학습

#### (1) XGBoost

##### 2<sup>nd</sup> GridSearchCV

```
xgb_1 = XGBClassifier(  
    learning_rate = 0.1,  
    n_estimators=100,  
    tree_method = 'gpu_hist')
```

**"gpu\_hist"** : XGBoost는 **GPU 환경**에서 "tree\_method" parameter를 **"gpu\_hist"**로 설정하여 **GPU로 적합**시킬 수 있다.  
(GPU를 지원해주는 **Google Colab** 덕분)

성능 소요 시간  
 Best Score : 0.9557605047327252  
 Best Params : {'max\_depth': 9, 'subsample': 0.7}  
 얼마나 걸렸뇨? : 64.44010829925537

#### 결과

→ **약 1분 소요**

→ **약 96%의 성능**  
(F1 score)

목표 parameters와 성능도 같게 나왔지만, 연산 속도가 **60배** 이상 **증가**하였다.

# 본론

\*앞서 구한 최적의 parameters를 적용

## Step 3) 머신러닝

### 4. 모델 학습

#### (1) XGBoost

#### 3<sup>rd</sup> GridSearchCV

##### 목표 parameters

"learning\_rate" /  
"n\_estimator"

→ XGBoost의 핵심적인 parameters 2개

```
xgb_params_3 = {
    'learning_rate' : [0.01, 0.1, 0.2],
    'n_estimators' : [100, 150, 200]}
```

→  $3 \times 3 = 9$ 개의 조합

##### 최적의 parameters

"learning\_rate" : 0.2 /  
"n\_estimators" : 200

##### 성능 소요 시간

```
Best Score : 0.973796413861263
Best Params : {'learning_rate': 0.2, 'n_estimators': 200}
얼마나 걸렸뇨? : 265.2431664466858
```

##### 결과

약 4분 소요

약 97%의 성능  
(F1 score)

거듭 최적의 parameters를 구하면서 반복하니 성능이 올라가는 것을 확인할 수 있다.

# 본론

\*앞서 구한 최적의 parameters를 적용

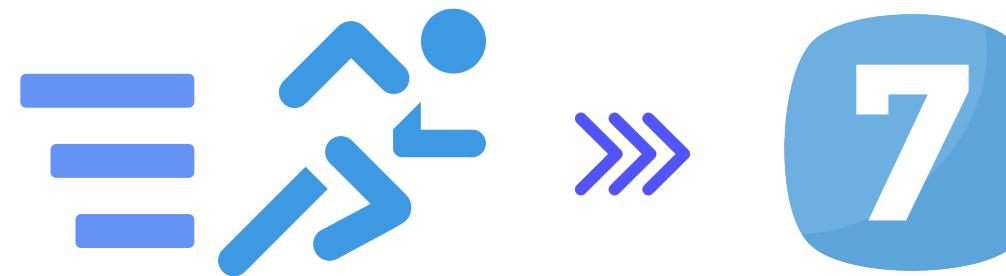
## Step 3) 머신러닝

### 4. 모델 학습

#### (1) XGBoost

~ 7<sup>th</sup> GridSearchCV

위와 같은 과정대로 총 7번의 GridSearchCV를 진행하였다.



3번째까지의 GridSearchCV로 parameters를 전체적으로 가능해보았고,  
이후 7번째까지는 주요 parameter인 max\_depth와 n\_estimators를 세밀하게 조정하였다.

# 본론

\*앞서 구한 최적의 parameters를 적용

## Step 3) 머신러닝

### 4. 모델 학습

#### (1) XGBoost

~ 7<sup>th</sup> GridSearchCV

그 결과,

base_score=0.5	learning_rate=0.2	n_jobs=1	scale_pos_weight=1
booster='gbtree'	max_delta_step=0,	nthread=None	seed=None
colsample_bylevel=1	max_depth=20	objective='binary:logistic'	silent=None
colsample_bynode=1	min_child_weight=1	random_state=0	subsample=0.9
colsample_bytree=0.9	missing=None	reg_alpha=0	tree_method='gpu_hist'
gamma=0	n_estimators=300	reg_lambda=1	verbosity=1

우리 데이터셋에 적합한 최적의 parameters를 위와 같이 구할 수 있었다.

# 본론

Step 3) 머신러닝

## 4. 모델 학습

구한 최적의 parameters로 다시 순차적으로

A : 모든 columns 포함

B : 예측 시 알 수 없는 columns 제거

A / B 두 개의 모델링을 이어서 진행하였다.

# 본론

Step 3) 머신러닝

## 4. 모델 학습

### (1) XGBoost

A 데이터셋  
2<sup>nd</sup> Model

```
x = df_a[list(df_a)[1:]]
y = df_a['enrolled']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=42)
```

학습 진행하기

```
# 시간 설정
import time
start = time.time()

## Xgboost Classifier
from xgboost import XGBClassifier, plot_importance

model = XGBClassifier()
model.fit(x_train, y_train)
```

\*A데이터셋 : 모든 columns 포함

기본 설정

train set : 0.75 / test set : 0.25

결과

소요 시간

113.95030045509338

→ 약 110초 소요

성능

0.978935892282205

→ 약 97.9%의 성능  
(F1 score)

# 본론

Step 3) 머신러닝

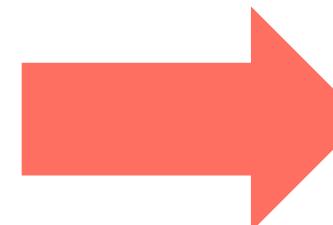
## 4. 모델 학습

### (1) XGBoost

\*A데이터셋 : 모든 columns 포함

A 데이터셋  
1<sup>st</sup> Model

약 86%의 성능  
(F1 score)



A 데이터셋  
2<sup>nd</sup> Model

약 97.9%의 성능  
(F1 score)

# 본론

Step 3) 머신러닝

## 4. 모델 학습

### (1) XGBoost

#### B 데이터셋

\*B데이터셋 : 예측 시 알 수 없는 columns 제거

```
x = df_a[list(df_a)[1:]]
y = df_a['enrolled']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=42)
```

학습 진행하기

```
# 시간 설정
import time
start = time.time()

## Xgboost Classifier
from xgboost import XGBClassifier, plot_importance

model = XGBClassifier()
model.fit(x_train, y_train)
```

#### 결과

##### 소요 시간

125.77204418182373

→ 약 125초 소요

##### 성능

0.9706325633141977

→ 약 97.1%의 성능  
(F1 score)

# 본론

Step 3) 머신러닝

## 4. 모델 학습

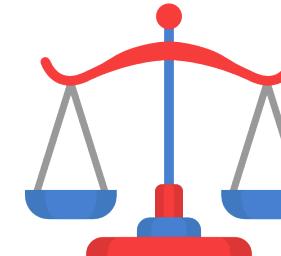
### (1) XGBoost

A 데이터셋  
Model

약 **97.9%**의 성능  
(F1 score)

B 데이터셋  
Model

약 **97.1%**의 성능  
(F1 score)



B는 A에 비해 정보가 적지만,  
좋은 성능을 기록했다.

# 본론

## Step 3) 머신러닝

### 4. 모델 학습

#### (2) LightGBM

##### A 데이터셋 Model

\*A데이터셋 : 모든 columns 포함

```
x = df_a[list(df_a)[1:]]
y = df_a['enrolled']
```

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
```

```
lgbm_params = {"max_depth": 20, "learning_rate" : 0.2, "num_leaves": 100,
 "n_estimators": 300}
```

```
start = time.time() # 시간 재보기

lgb_train = lgb.Dataset(X_train, y_train)
lgb_eval = lgb.Dataset(X_test, y_test, reference=lgb_train)

model = lgb.train(lgbm_params, lgb_train, 2500, lgb_eval, verbose_eval=10)
print("얼마나 걸렸뇨? :", time.time()-start) # 현재시간 - 시작시간 = 실행시간
```

##### 기본 설정

train set : 0.75 / test set : 0.25

##### 결과

##### 소요 시간

21.557641744613647

→ 약 21초 소요

##### 성능

0.9863066935829619

→ 약 98.6%의 성능  
(roc\_auc)

# 본론

## Step 3) 머신러닝

### 4. 모델 학습

#### (2) LightGBM

##### B 데이터셋 Model

```
x = df_b[list(df_b)[1:]]
y = df_b['enrolled']
```

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
```

```
lgbm_params = {"max_depth": 20, "learning_rate" : 0.2, "num_leaves": 100,
 "n_estimators": 300}
```

```
start = time.time() # 시간 재보기

lgb_train = lgb.Dataset(X_train, y_train)
lgb_eval = lgb.Dataset(X_test, y_test, reference=lgb_train)

model = lgb.train(lgbm_params, lgb_train, 2500, lgb_eval, verbose_eval=10)
print("얼마나 걸렸뇨? :", time.time()-start) # 현재시각 - 시작시간 = 실행시간
```

\*B데이터셋 : 예측 시 알 수 없는 columns 제거

##### 기본 설정

train set : 0.75 / test set : 0.25

##### 결과

###### 소요 시간

20.561105728149414

→ 약 21초 소요

###### 성능

0.9752616902926499

→ 약 97.5%의 성능  
(roc\_auc)

# 본론

Step 3) 머신러닝

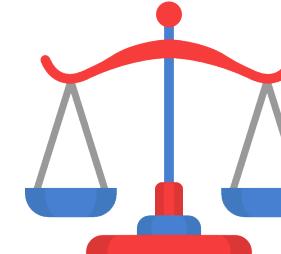
## 4. 모델 학습

### (2) LightGBM

A 데이터셋  
Model

약 **98.5%**의 성능  
(F1 score)

LightGBM 역시,



B는 A에 비해 정보가 적지만,  
좋은 성능을 기록했다.

B 데이터셋  
Model

약 **97.5%**의 성능  
(F1 score)

## 본론

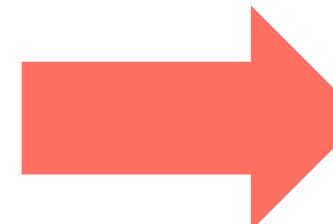
Step 3) 머신러닝

## 4. 모델 학습

## XGBoost vs LightGBM

A / B  
Model평균 약 120초 소요약 97.5%의 성능  
(F1 score)A / B  
Model평균 약 20초 소요약 98.1%의 성능  
(roc\_auc)

LightGBM이 XGBoost에 비해

연산 속도가 빠르고,  
좋은 성능을 기록했다.

# 본론

Step 3) 머신러닝

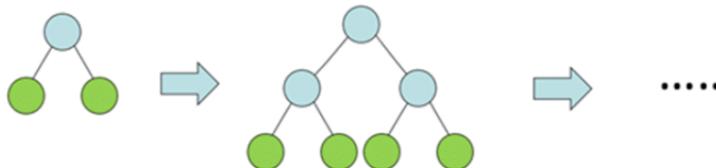
## 4. 모델 학습

### XGBoost vs LightGBM

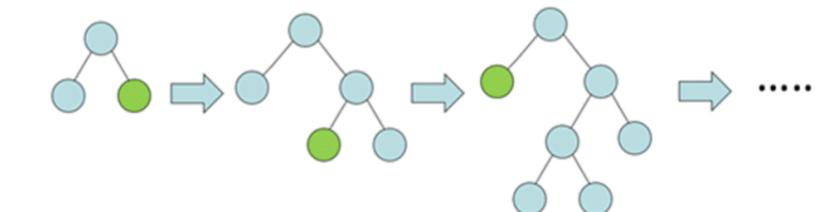
왜 LightGBM이 XGBoost에 비해  
연산 속도가 빠르고 성능이 더 좋은가?

LightGBM이 ~~Level-wise~~ 방식이 아니라 **Leaf-wise** 방식이기 때문이다.

**Level-wise**



**Leaf-wise**



따라서 연산 속도가 빠르고, 메모리 사용량이 적다.

# 본론

Step 3) 머신러닝

## 4. 모델 학습

### 클래스 생성

#### CRpredict

클래스 CRpredict의 주요 기능은 다음과 같다.

사용자가 자신의 수강신청 정보를 입력했을 시

(1) 합 / 불 여부를 반환한다.

```
predict_1st.prediction()
```

'가능! '

(2) 최적의 마일리지를 반환한다.

```
predict_1st.optimum_mileage
```

# 본론

Step 3) 머신러닝

## 4. 모델 학습

### 최종 모델

XGBoost를 사용하였다.



우리의 목표 모델은 학정번호와 교수님 데이터를 뽑아서 얻을 수 있다.

LightGBM에서는 1만 개의 row가 없으면 과적합 위험이 큰데,  
우리 데이터셋에서는 1만 개 이상의 데이터를 가진 교수님들이 적다.

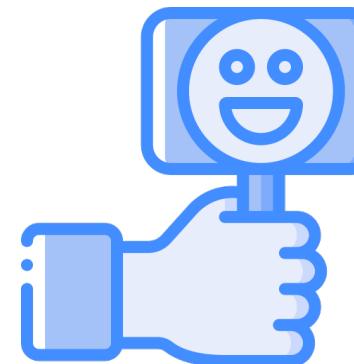
따라서 XGBoost를 최종 모델로 사용하기로 했다.

# 본론

Step 3) 머신러닝

## 4. 모델 학습

### 성능 측정

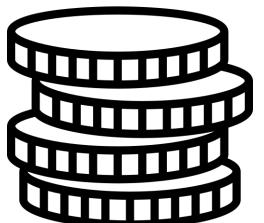


무작위로 100개의 학정번호를 추출하여 모델링한 결과  
평균 96.3~96.8%의 높은 성능을 보였다.

# 목차



서론

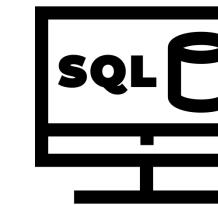


목표

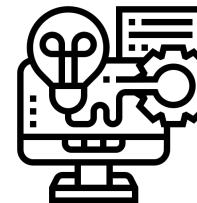
본론



크롤링



DB

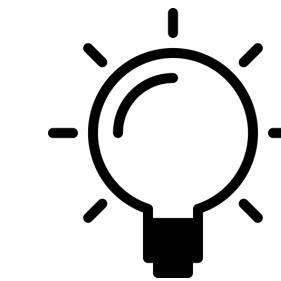


머신러닝



웹

결론



결론

# 본론

Step 4)

웹

## 웹 구현

### 목표

1. 수강자 정보 입력시 예측 결과 출력
2. 연세포탈 수강편람 연동하여 조회

# 본론

Step 4)

웹

## 웹 구현

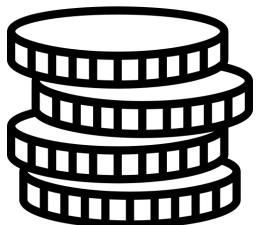
파이썬 웹 프레임워크인 **Flask**를 이용하여 데이터가 담긴 **MySQL**이랑  
**사용자의 입력 정보**를 **연결**해서 보여줄 수 있는 **웹**을 구현하였다.



# 목차



서론



목표

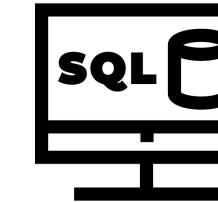
본론



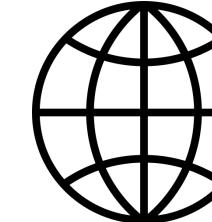
크롤링



머신러닝



DB



웹

결론



결론

# 결론



## 마일리지 수강신청 예측 웹페이지

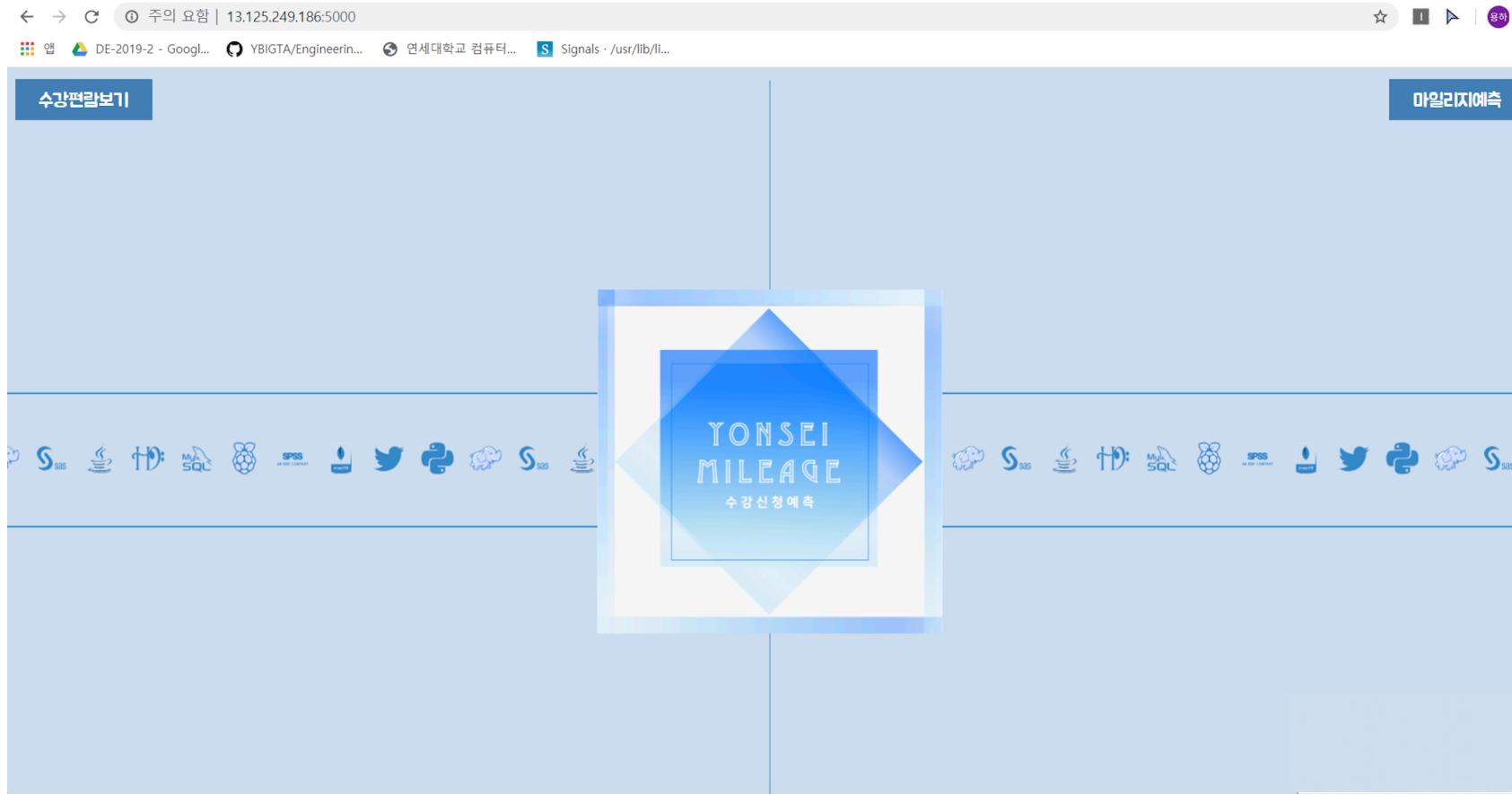
웹페이지는 크게 **5가지** 페이지로 **구분**된다.

- (1) 메인 페이지
- (2) 편람 조회 입력 페이지
- (3) 편람 조회 결과 페이지
- (4) 예측 정보 입력 페이지
- (5) 예측 결과 페이지

## 결론

## 메인 페이지

## 마일리지 수강신청 예측 웹페이지



## 결론



## 마일리지 수강신청 예측 웹페이지

편람 조회 입력 페이지

The screenshot shows a web browser window with the URL `13.125.249.186:5000/search`. The page title is "마일리지 수강신청 예측 웹페이지". On the left, there is a button labeled "편람 조회 입력 페이지". The main content area contains a search form with the following fields:

- 학기 :
- 학정번호 :
- 과목명 :
- 교수 :

At the bottom of the form are two buttons: "검색" (Search) and "뒤로 가기" (Back). In the top right corner of the page, there is a "HOME" link.

# 결론



## 결론



## 마일리지 수강신청 예측 웹페이지

예측 정보 입력 페이지

정보를 입력해주세요

HOME

학정번호:	<input type="text" value="00: CS3102"/>	전공자:	<input type="radio"/> YES <input type="radio"/> NO
담당교수:	<input type="text" value="예: 이경우"/>	복수전공자:	<input type="radio"/> YES <input type="radio"/> NO
정원:	<input type="text" value="예: 90"/>	졸업신청:	<input type="radio"/> YES <input type="radio"/> NO
최대 마일리지:	<input type="text" value="00: 36"/>	초수강여부:	<input type="radio"/> YES <input type="radio"/> NO
전공자 정원:	<input type="text" value="예: 70 (있으면 -1)"/>	학년:	<input type="text" value="1학년"/>
복수전공자 포함:	<input type="radio"/> YES <input type="radio"/> NO	신청과목수:	<input type="text" value="1과목"/>
1학년 정원:	<input type="text" value="예: 0(있으면 -1)"/>	총이수학점/졸업이수학점:	<input type="text" value="예: 0.7701"/>
2학년 정원:	<input type="text" value="예: 30(있으면 -1)"/>	직전학기이수학점/학기당수강학점:	<input type="text" value="예: 0.8311"/>
3학년 정원:	<input type="text" value="예: 30(있으면 -1)"/>	마일리지:	<input type="text" value="00: 28"/>
4학년 정원:	<input type="text" value="예: 30(있으면 -1)"/>	<input type="button" value="확인"/>	

# 결론

예측 결과 페이지

마일리지 수강신청 예측 웹페이지

시연



시연을 통해 예측 결과 페이지까지 확인해보도록 하겠다.

# 결론



## 활용 방안

### 앞으로의 수강신청

2020학년도 1학기 수강신청부터 마일리지 수강신청 예측 웹페이지를 이용하여, 합리적인 수강신청 의사결정에 도움을 받을 수 있다.



# 감사합니다

## 수강신청 결과 예측과 웹 구현

웹을 통한 마일리지 수강 신청 가이드라인 제공



안주영 | 이용하 | 정현우 | 최민태 | 한승희

