DScover 1st Seminer

2021.09.06.

Index

1 **스몰 토크**
진행자 : 정원(운영채널)

2 **직무 소개**
진행자 : 유림(학습채널)

3 **파이썬 라이브러리**
part 1(pandas, numpy)
진행자 : 범규(학습채널)

4 **머신러닝의 전체적인 개요**
진행자 : 새미(학습채널)

5 **파이썬 라이브러리**
part 2(scikit-learn)
진행자 : 새미(학습채널)

6 **스터디 조별 회의**
진행시간 : 30분

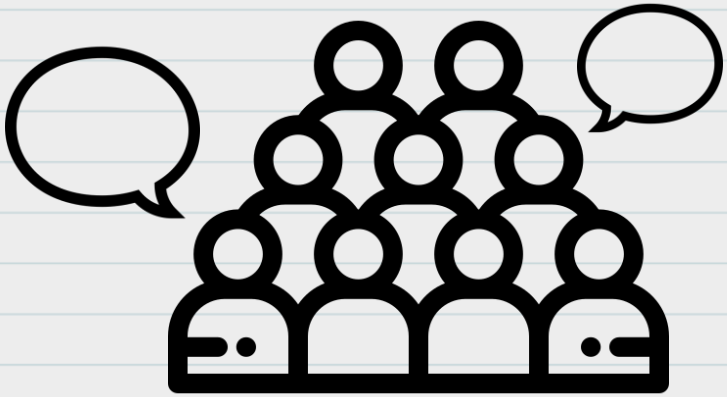
01

스몰 토크

진행자 : 정원, 동하(운영채널)

스몰토크가 무엇인가요?

학회원 2명을 무작위로 지목해서 간단한 질문을 통해
본격적인 세미나 전에 분위기를 환기시키는 활동



02

데이터 직무 소개

데이터를 활용하는 직무들의
종류 및 업무 소개

데이터 직무 소개

A회사, 데이터분석가

보편적인 데이터분석가

: 데이터를 분석하여 인사이트를
도출하는 역할

B회사, 데이터분석가

데이터분석가
+ 데이터 사이언티스트

: 데이터를 분석하여 인사이트 도출 및
개발을 통해 새로운 분석 모델 적용

회사에 따라서도 명칭 및 업무가 다름.
직무 간의 업무를 명확히 나누기 어렵.

데이터 분석가

데이터 분석 및 인사이트 도출

- 데이터 활용 기획
- 데이터 분석 및 시각화
- 회사 내 Key 비즈니스 메트릭스의 모니터링 및 보고

  * 메트릭스 : 업무 수행 결과를 보여주는 계량적 분석

- 전사적 KPI 설정 및 관리

  * KPI : 핵심성과지표

- 이슈 및 트렌드 분석
- 타 부서의 요청 데이터 처리

https://www.korea.kr/news/policyNewsView.do?newsId=148777363

https://www.mobiinside.co.kr/2016/10/21/bigdata-specialist/

Justin Grimes/Flickr.com

https://www.sasbigdata.com/

데이터 사이언티스트

모델 개발 및 적용

- 주로 머신러닝 모델, 딥러닝 모델 개발 및 적용
- 도메인의 목적에 맞는 모델 개발
- 데이터 모델을 커스터마이징하고 알고리즘 개발
- 예측 모델링 기법을 활용해 사용자 경험, 수익 창출 등을 최적화
- A/B 테스팅 및 테스트 모델의 품질 개선

데이터 엔지니어

개발자

- 데이터 수집 및 가공

- 각종 소스의 데이터를 분석하기 위한 데이터를 모으는 파이프 라인 개발

- 파이프 라인 및 데이터 레이크 관리

  * 데이터 레이크 : 가공되지 않은 상태로 저징되어 접근이 가능한 엄청난 양의 데이터

- 데이터 분석가에게 제공할 대시보드 개발

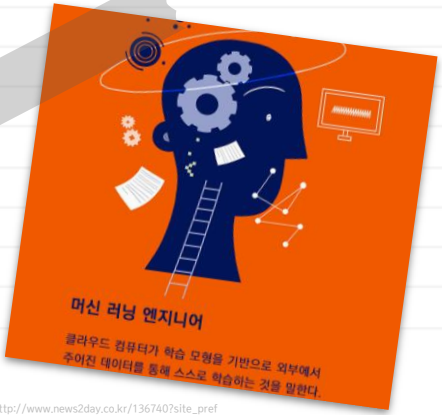
- 데이터 마트 및 통계 개발

  * 데이터 마트 : 데이터 웨어하우스에 있는 데이터를 사용자의 요구 항목에 따라 체계적으로 분석한 데이터베이스의 형태

https://soft.plusblog.co.kr/9

http://www.codingworldnews.com/news/articleView.html?idxno=3252

https://m.hellot.net/mobile/article.html?no=42605



http://www.news2day.co.kr/136740?site_preference=normal

머신러닝 엔지니어

ML, DL 기반의 개발자

- ML 모델 개발, 학습, 배포
- ML 모델 API서버 개발

  * API : 응용프로그램에서 특정한 기능을 사용하기 위해 필요한 데이터를 주고받게끔 만든 도구나 방법.

- ML 모델 학습 및 배포 자동화 파이프라인 구축
- 도메인의 목적에 맞는 모델 개발
- 데이터 사이언티스트에 비해 Production(실제 서비스)에 집중

중간 정리

데이터 수집/전처리, 데이터 베이스 구축/관리에 관심이 있다

-> **데이터 엔지니어**

API 개발, 플랫폼 구축, ML/DL 개발 및 개선에 관심이 있다

-> **머신러닝 엔지니어**

데이터 활용 기획, 데이터 대시보드 생성/관리에 관심이 있다

-> **데이터 분석가**

ML/DL 관련 R&D, 논문 연구와 통계 모델링에 관심이 있다

-> **데이터 사이언티스트**

“데이터를 활용하는 직무”

프로덕트 분석가

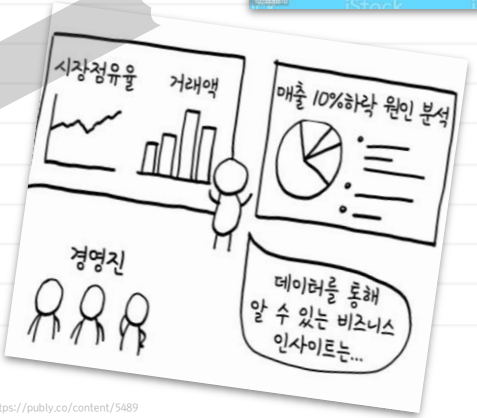
제품 데이터를 분석하는 분석가

- 앱 또는 웹 서비스에서 발생하는 유저 행동 로그 데이터를 분석
  예) 고객이 제품을 어떻게 사용하고 있을까?
  퍼널별로 얼마나 체류할까?(퍼널분석)

- 제품 개선을 위한 유저 활동 데이터 분석

https://www.jobplanet.co.kr/contents/news-1582

https://publy.co/content/4526



https://publy.co/content/5489

비즈니스 분석가

비즈니스를 분석하는 분석가

- 비즈니스는 프로덕트와 연관된 경우가 많으나 “매출“, “비용“ 등 사업의 가치에 더 집중
- 비즈니스 KPI 모니터링
- 매출, 비용, 손익 등을 분석
- 빠른 의사 결정을 위한 데이터 분석
- 가설 검증 및 AB 테스트

퍼포먼스 마케터

데이터를 기반으로 마케팅

- 매체 광고 최적화 및 광고 집행 분석
- 마케팅 KPI 모니터링

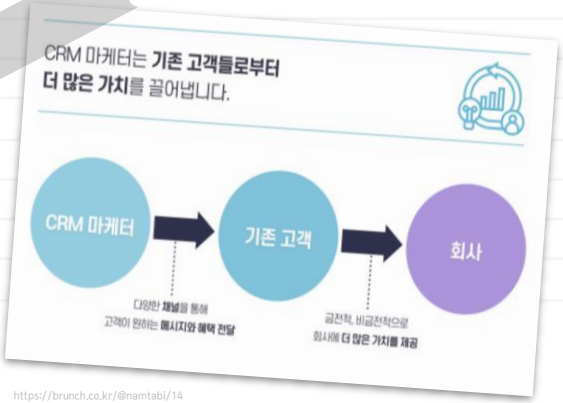
CRM 마케터

고객 관계 관리에 기초한 마케팅

- 고객과 커뮤니케이션 및 분석
- 유저 Life Cycle 관리

  예) 고객 등급별 분석, 쿠폰 마케팅, 고객 Segment 분석, AB 테스트 등

https://www.superookie.com/contents/5d26b6318b129f3d87383bd2



https://brunch.co.kr/@namtabi/14

03

파이썬 라이브러리

part 1(pandas, numpy)

진행자 : 범규(학습채널)

Pandas

Pandas란?

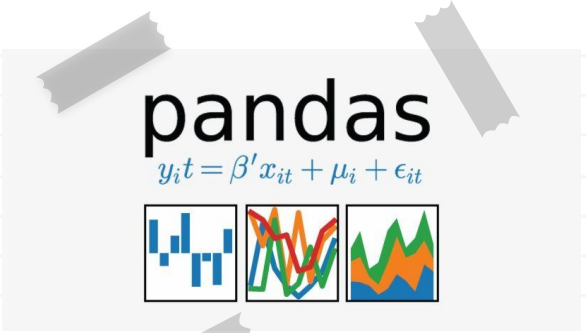
파이썬 기반의 데이터 분석과 시각화 라이브러리

Pandas의 특징

용이한 데이터 분석 함수 제공

대용량의 데이터를 빠르고 쉽게 분석 가능

고차원의 데이터를 변형, 제거, 추가에 용이



Series와 DataFrame

Series

1차원 배열의 값에 대응되는
인덱스가 부여되어 있는 구조

```
도시
서울      9904312
부산      3448737
인천      2890451
대구      2466052
Name: 인구, dtype: int64
```

DataFrame

시리즈 여러 개가 모여 행과 열로
이루어진 2차원 배열

	지역	2015	2010	2005	2000	2010-2015 증가율
서울	수도권	9904312	9631482	9762546	9853972	0.0283
부산	경상권	3448737	3393191	3512547	3655437	0.0163
인천	수도권	2890451	2632035	2517680	2466338	0.0982
대구	경상권	2466052	2431774	2456016	2473990	0.0141

Series와 DataFrmae 생성

DataFrame

```
values = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
index = ['one', 'two', 'three']
columns = ['A', 'B', 'C']
df = pd.DataFrame(values, index=index, columns=columns)
df
```

	A	B	C
one	1	2	3
two	4	5	6
three	7	8	9

Series

```
s = pd.Series([1, 4, 7],index=["one", "two", "three"])
s
```

```
one      1
two      4
three    7
dtype: int64
```

데이터 불러오기

CSV, txt, Excel, SQL등 다양한
데이터들을 읽고 불러오기 가능

데이터가 위치한 경로 확인

cd 경로 입력

ls로 현재 위치한 경로 확인

df = pd.read_csv("파일 이름")

```
cd C:₩Users₩82103₩Desktop₩data
```

```
df = pd.read_csv('SP player.csv')
```

```
df
```

	Name	Team	W	L	G	IP	ERA	WAR	playerid	IP/G
0	Griffin Canning	Angels	5	5	17	88.1	4.18	1.4	19867	5.182353
1	Tyler Skaggs	Angels	7	7	15	79.2	4.29	1.8	10190	5.280000
2	Andrew Heaney	Angels	4	6	18	95.1	4.91	1.2	15423	5.283333
3	Jaime Barria	Angels	2	7	13	60.1	5.52	-0.4	18356	4.623077
4	Dillon Peters	Angels	3	4	12	55.2	5.82	-0.4	18790	4.600000
...	...	...	...	...	...	...	...	...	...	...
165	James Paxton	Yankees	15	6	29	150.2	3.82	3.5	11828	5.179310
166	Domingo German	Yankees	16	4	24	134.2	4.28	1.5	17149	5.591667
167	Masahiro Tanaka	Yankees	11	8	31	179.0	4.47	3.2	15764	5.774194
168	CC Sabathia	Yankees	5	8	22	106.1	4.99	0.4	404	4.822727
169	J.A. Happ	Yankees	12	8	30	156.1	5.01	1.1	7410	5.203333

170 rows × 10 columns

인덱싱

df[n:m] ← 설정 범위 인덱싱

df.loc['행이름']

df.iloc[인덱스 번호]

#Boolean indexing
[ ] 연산자 내에 조건 입력
→ 조건에 맞는 데이터 인덱싱

```
df.iloc[52]
```

```
Name          Hyun-Jin Ryu
Team               Dodgers
W                        14
L                         5
G                        29
IP                    182.2
ERA                    2.32
WAR                     4.8
playerid              14444
Name: 52, dtype: object
```

```
df[10:13]
```

	Name	Team	W	L	G	IP	ERA	WAR	playerid
10	Justin Verlander	Astros	21	6	34	223.0	2.58	6.4	8700
11	Zack Greinke	Astros	18	5	33	208.2	2.93	5.4	1943
12	Wade Miley	Astros	14	6	33	167.1	3.98	2.0	8779

```
df.loc[df['W']>18,:] #18승 이상 기록한 선수
```

	Name	Team	W	L	G	IP	ERA	WAR	playerid
9	Gerrit Cole	Astros	20	5	33	212.1	2.50	7.4	13125
10	Justin Verlander	Astros	21	6	34	223.0	2.58	6.4	8700
126	Eduardo Rodriguez	Red Sox	19	6	34	203.1	3.81	3.7	13164

인덱싱 함수(loc / iloc)

loc

라벨 값 기반의 2차원 인덱싱

df.loc[행 이름] 또는
df.loc[행 이름, 열 이름]

불린 인덱싱 가능

	A	B	C	D
a	10	11	12	13
b	14	15	16	17
c	18	19	20	21

```
df1.loc["b":"c","B":"C"]
```

	B	C
b	15	16
c	19	20

iloc

정수 기반의 2차원 인덱싱

df.iloc[행 인덱싱 값] 또는
df.iloc[행 인덱싱 값, 열 인덱싱 값]

불린 인덱싱 불가

```
df1.iloc[1:3,1:3]
```

	B	C
b	15	16
c	19	20

데이터 정렬

Index 기준 : df.sort_index(axis=0)

특정 열 값 기준 :
df.sort_values(by="특정열")

정렬 기준은 작은 수에서 큰 수로,
큰 수에서 작은 수로 정렬하려면
ascending=False 인수를 지정

```
df.sort_values(by='W')
```

	Name	Team	W	L	G	IP	ERA	WAR	playerid
92	Gabriel Ynoa	Orioles	0	9	13	64.1	6.02	-0.6	12938
130	Nathan Eovaldi	Red Sox	1	1	12	54.1	6.13	-0.6	9132
93	David Hess	Orioles	1	10	14	67.2	7.18	-0.6	16130
71	Felix Hernandez	Mariners	1	8	15	71.2	6.40	-0.1	4772
139	Chi Chi Gonzalez	Rockies	1	5	12	58.0	5.43	0.2	14663
...	...	...	...	...	...	...	...	...	...
85	Stephen Strasburg	Nationals	18	6	33	209.0	3.32	5.7	10131
11	Zack Greinke	Astros	18	5	33	208.2	2.93	5.4	1943

```
df.sort_values(by='W',ascending=False)
```

	Name	Team	W	L	G	IP	ERA	WAR	playerid
10	Justin Verlander	Astros	21	6	34	223.0	2.58	6.4	8700
9	Gerrit Cole	Astros	20	5	33	212.1	2.50	7.4	13125
126	Eduardo Rodriguez	Red Sox	19	6	34	203.1	3.81	3.7	13164
85	Stephen Strasburg	Nationals	18	6	33	209.0	3.32	5.7	10131
11	Zack Greinke	Astros	18	5	33	208.2	2.93	5.4	1943
...	...	...	...	...	...	...	...	...	...
154	Jordan Zimmermann	Tigers	1	13	23	112.0	6.91	1.3	4505
130	Nathan Eovaldi	Red Sox	1	1	12	54.1	6.13	-0.6	9132

데이터 연산

데이터 분석 통계 함수

sum, mean, min, max, var, corr 등

```
df['ERA'].mean()
```

```
4.539235294117644
```

```
df['W'].max()
```

```
21
```

```
df.['특정열'].corr(df["비교하고 싶은 열"])
```

연산으로 새로운 열 생성

```
df['IP/G'] = df['IP'] / df['G']
```

	Name	Team	W	L	G	IP	ERA	WAR	playerid	IP/G
0	Griffin Canning	Angels	5	5	17	88.1	4.18	1.4	19867	5.182353
1	Tyler Skaggs	Angels	7	7	15	79.2	4.29	1.8	10190	5.280000
2	Andrew Heaney	Angels	4	6	18	95.1	4.91	1.2	15423	5.283333
3	Jaime Barria	Angels	2	7	13	60.1	5.52	-0.4	18356	4.623077
4	Dillon Peters	Angels	3	4	12	55.2	5.82	-0.4	18790	4.600000
...	...	...	...	...	...	...	...	...	...	...
165	James Paxton	Yankees	15	6	29	150.2	3.82	3.5	11828	5.179310
166	Domingo German	Yankees	16	4	24	134.2	4.28	1.5	17149	5.591667
167	Masahiro Tanaka	Yankees	11	8	31	179.0	4.47	3.2	15764	5.774194
168	CC Sabathia	Yankees	5	8	22	106.1	4.99	0.4	404	4.822727
169	J.A. Happ	Yankees	12	8	30	156.1	5.01	1.1	7410	5.203333

Groupby 함수

같은 값의 데이터를 하나로 묶어
통계 및 집계 결과 도출하는 함수

df.groupby(그룹핑 대상)

연산 가능한 전체 column 계산 :
df.groupby(그룹핑 대상).연산함수()

특정 column만 계산 :
df[‘특정 열’].groupby(그룹핑 대상).연산함수()

```
df.groupby(df['Team']).mean()
```

Team	W	L	G	IP	ERA	WAR	playerid
Angels	3.500000	5.625000	14.125000	69.625000	5.697500	0.287500	15062.375000
Astros	13.833333	7.000000	29.166667	170.283333	3.686667	3.850000	8239.666667
Athletics	11.666667	6.500000	27.833333	153.900000	3.881667	2.266667	9950.833333
Blue Jays	3.666667	6.333333	18.000000	87.733333	5.243333	0.800000	13269.666667

```
df['WAR'].groupby(df['Team']).mean()
```

```
Team
Angels        0.287500
Astros        3.850000
Athletics     2.266667
Blue Jays     0.800000
```

Numpy

Numpy란?

수치 해석용 배열 라이브러리

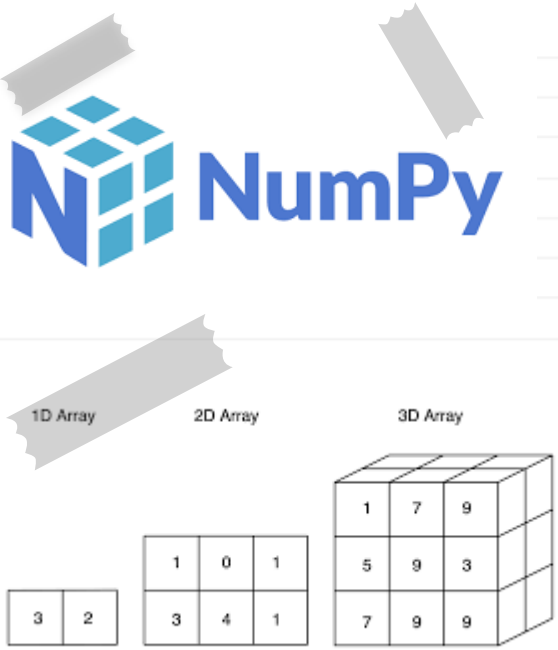
벡터, 행렬, 선형대수 관련 수치 연산 가능

Numpy의 특징

다차원 배열을 다룰 수 있음

간단한 코드로도 복잡한 수식 연산 가능

파이썬 반복문이나 list에 비해 속도가 빠름



배열 생성 (ndarray)

1차원(rank=1) 배열 생성

```
a = np.array([1,2,3,4])
a
```

```
array([1, 2, 3, 4])
```

2차원(rank=2) 배열 생성

```
b = np.array([[1,2,3,4],[10,20,30,40]])
b
```

```
array([[ 1,  2,  3,  4],
       [10, 20, 30, 40]])
```

배열 연산

이 외에도 고차원 배열 생성

같은 rank에서 연산 가능

```
d = np.array([[1,2],[10,20]])
s = np.array([[10,20],[100,200]])
```

```
d+s
```

```
array([[ 11,  22],
       [110, 220]])
```

```
d-s
```

```
array([[ -9, -18],
       [-90, -180]])
```

```
d/s
```

```
array([[0.1, 0.1],
       [0.1, 0.1]])
```

```
d*s
```

```
array([[  10,   40],
       [1000, 4000]])
```

```
np.dot(d,s)
```

```
array([[ 210,  420],
       [2100, 4200]])
```

Nparray와 python list의 차이점

nparray

같은 인덱스에 위치한 값끼리 계산

```
nparray1 = np.array([1,2,3])
nparray2 = np.array([4,5,6])
nparray1 + nparray2
```

```
array([5, 7, 9])
```

Python list

앞의 리스트에 뒤에 리스트가 붙어 한 개의 리스트로 됨

```
pylist1 = [1,2,3]
pylist2 = [4,5,6]
pylist1 + pylist2
```

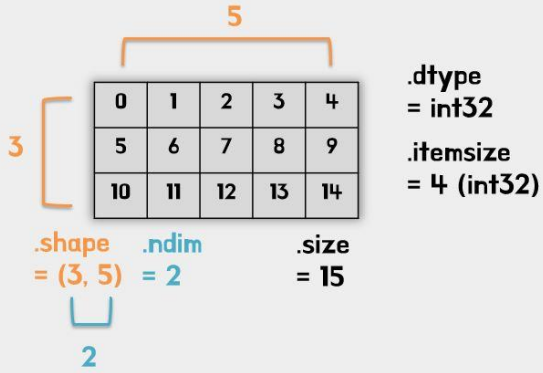
```
[1, 2, 3, 4, 5, 6]
```

Ndarray 관련 함수들

labaputer.github.io

Numpy.ndarray

.shape / . ndim / .dtype / .itemsize / .size



.shape : 배열의 각 축의 크기

.ndim : 배열의 차원

.dtype : 각 요소의 데이터 타입

.itemsize : 각 요소 타입의 bytes 크기

.size : 전체 요소의 개수

인덱싱

slicing

```
a = np.array([[0, 1, 2], [3, 4, 5]])
a
```

```
array([[0, 1, 2],
       [3, 4, 5]])
```

결과로 도출되는 배열은 항상
원본 배열의 부분 배열

```
a[0, :]   # 첫번째 행 전체
```

```
array([0, 1, 2])
```

```
a[:, 1]   # 두번째 열 전체
```

```
array([1, 4])
```

```
a[1, 1:]  # 두번째 행의 두번째 열부터 끝열까지
```

```
array([4, 5])
```

```
a[:2, :2] # 두번째 행까지, 두번째 열까지
```

```
array([[0, 1],
       [3, 4]])
```

배열 인덱싱 (Fancy Indexing)

불린 인덱싱

인덱스 배열의 원소가 True, False 두 값으로만 구성

특정 조건에 맞는 원소를 찾아낼 때 주로 사용

```
a = np.array([[0, 1, 2], [3, 4, 5]])
a
```

```
array([[0, 1, 2],
       [3, 4, 5]])
```

```
a % 2 ==0 #각 원소가 조건에 부합하는지
```

```
array([[ True, False,  True],
       [False,  True, False]])
```

```
a[a % 2 == 0] #조건에 부합하는 원소들
```

```
array([0, 2, 4])
```

배열 인덱싱 (Fancy Indexing)

정수 배열 인덱싱

인덱스 원소 각각이 원래 배열의
원소 하나를 가리켜야 함

```
a = np.array([11, 22, 33, 44, 55, 66, 77, 88, 99])
idx = np.array([0, 2, 4, 6, 8])
a[idx]
```

```
array([11, 33, 55, 77, 99])
```

그렇지 않으면 오류

```
idx[a] #a의 원소가 idx의 원소를 가르키지 못함
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-101-4e2735ea5736> in <module>
----> 1 idx[a]

IndexError: index 11 is out of bounds for axis 0 with size 5
```

배열 인덱싱 (Fancy Indexing)

정수 배열 인덱싱

고차원에서도 정수 배열 인덱싱 가능

```
a = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
a #다차원에서의 정수 배열 인덱싱
```

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

```
idx=np.array([2,0,1])
```

```
a[[idx],:]
```

```
array([[[ 9, 10, 11, 12],
        [ 1,  2,  3,  4],
        [ 5,  6,  7,  8]]])
```

```
a[[idx],[1,2,3]]
```

```
array([[10,  3,  8]])
```

```
a[[idx],[idx]]
```

```
array([[11,  1,  6]])
```

04

머신러닝의 전체적인 개요

진행자 : 새미(학습채널)

머신러닝의 전체적인 개요

1. 정의
2. 지도 학습
    - 2-1. 회귀 분석
        - 2-1-1. 단일 변수
            - 2-1-1-1. 가설 함수
            - 2-1-1-2. 손실 함수
            - 2-1-1-3. 학습 알고리즘
        - 2-1-2. 다변수
            - 2-1-2-1. 가설 함수
            - 2-1-2-2. 손실 함수
            - 2-1-2-3. 학습 알고리즘
            - 2-1-2-3. 특성 표준화
            - 2-1-2-5. 과적합/과소적합
            - 2-1-2-6. 최적화
            - 2-1-2-7. 학습곡선
    - 2-2. 분류 분석
        - 2-2-1. 이진 분류
            - 2-2-1-1. 가설 함수
            - 2-2-1-2. 손실 함수
            - 2-2-1-3. 학습 알고리즘
            - 2-2-1-4. 과적합/과소적합
            - 2-2-1-5. 평가
        - 2-2-2. 다중 분류
            - 2-2-2-1. 분류 방법
3. 비지도 학습
    - 3-1. clustering
    - 3-2. non-clustering

1. 머신러닝 (Machine Learning) 이란 ?

Arthur Samuel

컴퓨터가 명시적 (explicit) 프로그램 없이도 스스로 학습할 수 있는 능력을 연구하는 학문 분야

QUIZ !! 다음 중 머신러닝에 해당하는 것을 모두 고르시오.

1. 집의 여러 특성을 바탕으로 집 값을 예측하는 프로그램
2. 학생을 점수에 따라 상위 20%는 A, 20~50%는 B, 나머지는 C로 분류하는 프로그램
3. 메일을 내용을 바탕으로 스팸메일인지 아닌지 분류하는 프로그램
4. 체스 게임을 수행하는 프로그램

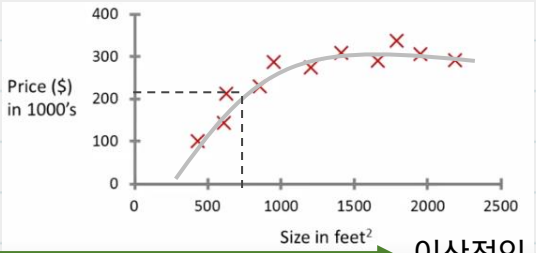
2. 지도 학습 (Supervised Learning)

지도 학습 : 학습하는 데이터에 정답이 포함되어 있는 경우

연속적인 값 : **회귀 모델**

예시 1) 집 가격 예측 프로그램

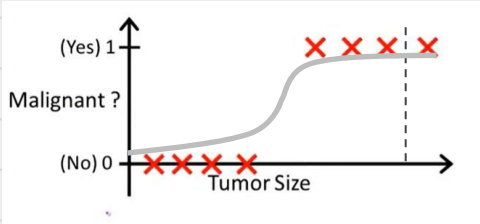
Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…



이산적인 값 : **분류 모델**

예시 2) 환자 종양의 양성/악성 판단 프로그램

Tumor Size in mm (x)	Malignant in (y)
10	0 (x)
50	1 (o)
…	….



2. 지도 학습 (Supervised Learning)

QUIZ !! 각 모델이 회귀 모델인지 분류 모델인지 고르시오

1. ABC마트에서 하루에 초콜릿이 몇 개 팔릴지 예측하는 모델
2. 소프트웨어 고객들의 계정이 해킹 당했는지 예측하는 모델

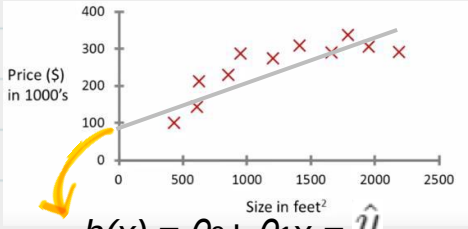
2-1. 회귀 분석 (Regression Analysis)

2-1-1. 단일 변수

2-1-1-1. 가설 함수 (Hypothesis Function)

지도 학습에서 타깃을 가장 잘 설명하는 함수

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...



$h(x) = \theta_0 + \theta_1 x = \hat{y}$

2-1-1-2. 손실 함수 (Loss Function)

알고리즘/모델의 성능을 평가하는 함수

$$(\hat{y}_i - y_i) \rightarrow (\hat{y}_i - y_i)^2 \rightarrow \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 \rightarrow \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 \rightarrow \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

손실 함수

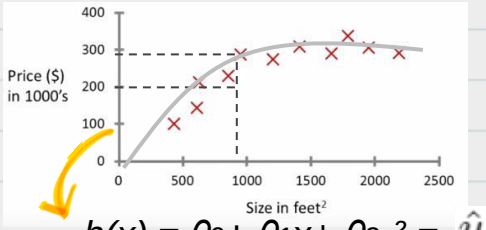
2-1. 회귀 분석 (Regression Analysis)

2-1-1. 단일 변수

2-1-1-1. 가설 함수 (Hypothesis Function)

지도 학습에서 타깃을 가장 잘 설명하는 함수

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...



$h(x) = \theta_0 + \theta_1 x + \theta_2 x^2 = \hat{y}$

2-1-1-2. 손실 함수 (Loss Function)

알고리즘/모델의 성능을 평가하는 함수

$$(\hat{y}_i - y_i) \rightarrow (\hat{y}_i - y_i)^2 \rightarrow \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 \rightarrow \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 \rightarrow \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

손실 함수

2-1. 회귀 분석

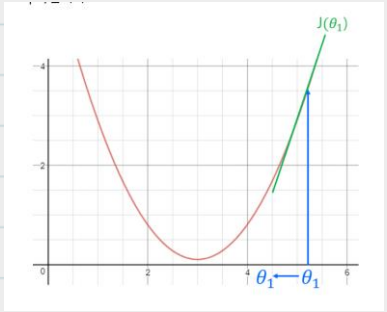
2-1-1. 단일 변수

2-1-1-3. 학습 알고리즘 (Learning Algorithm) - 경사하강법

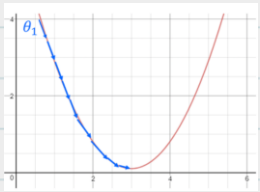
프로그램이 인간의 학습 방식을 모방할 수 있도록 하는 지시

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$
$$for\ j = 0\ and\ j = 1)$$

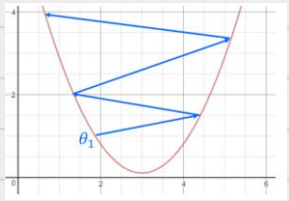
학습률
(Learning Rate)



학습률이 낮은 경우



학습률이 높은 경우



2-1. 회귀 분석

2-1-2. 다변수

2-1-2-1. 가설 함수 (Hypothesis Function)

Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

$$= \begin{bmatrix} \theta_0 & \theta_1 & \dots & \theta_n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = \theta^T x$$

* 편의상 $x_0^{(i)} = 1 \text{ for } (i \in 1, \dots, m)$ 로 정의

2-1-2-2. 손실 함수 (Loss Function)

단일 변수 손실 함수와 동일

$$\frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

2-1. 회귀 분석

2-1-2. 다변수

2-1-2-3. 학습 알고리즘 (Learning Algorithm) - 경사하강법

$$
\begin{aligned}
&\text{repeat until convergence: } \{ \\
&\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)} \\
&\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)} \\
&\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)} \\
&\dots \\
&\}
\end{aligned}
$$

경사 하강법 외 배치 경사 하강법, 확률적 경사 하강법, 미니배치 경사 하강법 등이 있음.

2-1-2-4. 특성 표준화 (Feature Scaling)

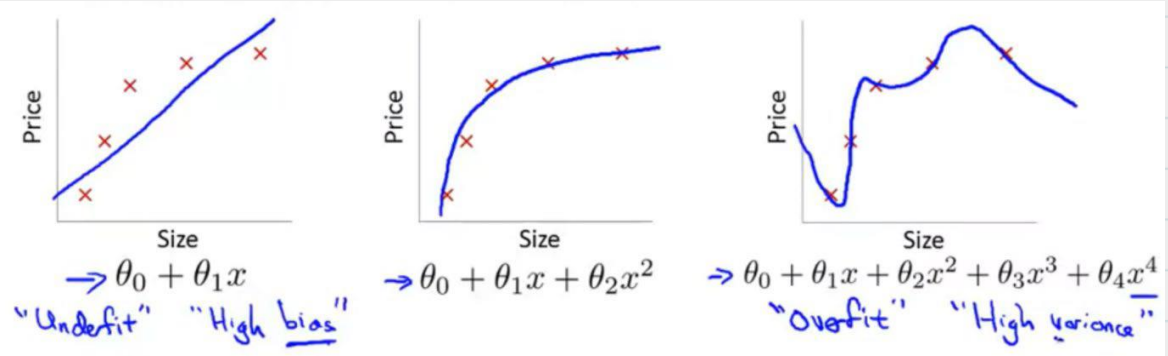
특성을 여러 개 사용할 경우 표준화가 꼭 필요함

$$
x_i := \frac{x_i - \mu_i}{s_i}
$$

2-1. 회귀 분석

2-1-2. 다변수

2-1-2-5. 과적합/과소 적합 (Overfitting/Underfitting)



① 선형 함수 (1차 함수)

과소 적합 (underfitting) /
높은 편향 (high bias)
너무 간단함 함수가 만들어질 때 발생

② 2차 함수

실제 값(데이터)와 잘 들어맞음

③ 4차 함수

과적합 (overfitting) /
높은 분산 (high variance)
Training data에 한해서만 데이터와 잘 맞음
정확한 집 값 예측 불가능
과도하게 복잡한 함수가 생성될 때 발생

2-1. 회귀 분석

2-1-2. 다변수

2-1-2-5. 과적합/과소 적합 (Overfitting/Underfitting)

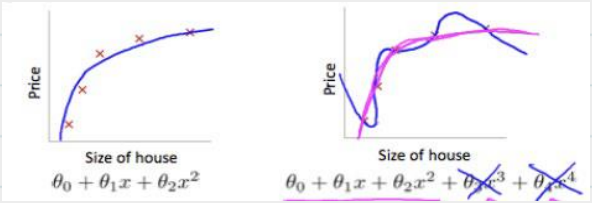
과적합 (Overfitting)

가설 함수가 훈련한 데이터에만 지나치게 잘 맞아서 새로운 데이터에 대한 일반화(예측)가 어려움

해결 방법 :

① 특성 선택 (Feature selection)

② 모든 특성을 유지하되, 각 θ 값을 줄임 -> 정규화 (Regularization)



기존 손실 함수

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

정규화가 적용된 손실 함수

$$min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000 \cdot \theta_3^2 + 1000 \cdot \theta_4^2$$

2-1. 회귀 분석

2-1-2. 다변수

2-1-2-5. 과적합/과소 적합 (Overfitting/Underfitting)

$$min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2$$

일반적으로 θ_0(i=0)의 포함 여부가 결과에 큰 영향을 미치지 않으므로
θ_0을 제외한 모든 θ에 패널티를 줌

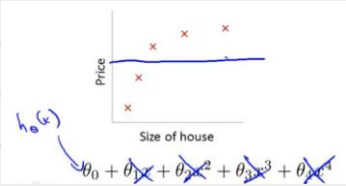
Regularization Parameter λ 의 역할 :

패널티의 크기 결정

Regularization Parameter λ가 무작정 크기만 하면 좋을까 ?

-> No !!! 모든 θ가 0에 가까워지면 평평한 가설함수가

만들어져 과소적합 발생



2-1. 회귀 분석

2-1-2. 다변수

2-1-2-6. 최적화 (Optimization)

비용 함수를 최소화하기 위해 하이퍼 파라미터 (α, λ 등)를 조정하는 과정

어떤 λ가 가장 적합할까 ?

Choosing the regularization parameter λ

Model: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

1. Try $\lambda = 0$
2. Try $\lambda = 0.01$
3. Try $\lambda = 0.02$
4. Try $\lambda = 0.04$
5. Try $\lambda = 0.08$

⋮

12. Try $\lambda = 10$

2-1. 회귀 분석

2-1-2. 다변수

2-1-2-6. 최적화 (Optimization)

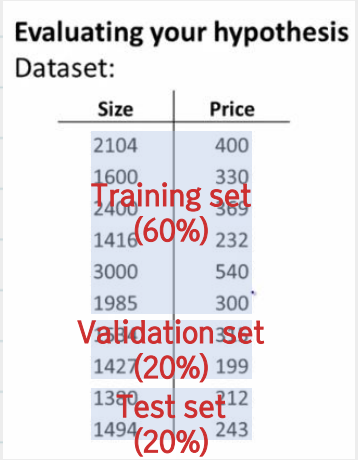
① Training(60%) / Validation(20%) / Test(20%) Split

② 특정 λ값을 대입하여 **Training set error**을 최소화하는 Θ 계산

③ ②에서 학습한 Θ를 Jcv에 대입해 **Validation set error** 계산

④ 여러 람다 값에 대하여 ②~③ 과정 반복 후, Validation set error가 가장 작은 모델 Select

⑤ Select된 모델의 일반적인 오차를 구하기 위해 ②~④에서 학습한 파라미터를 바탕으로 **test set error** 계산

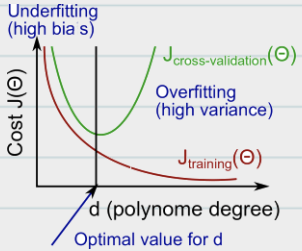


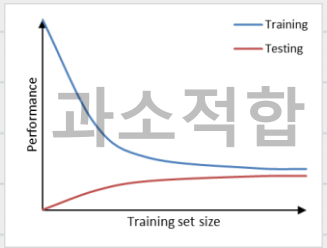
2-1. 회귀 분석

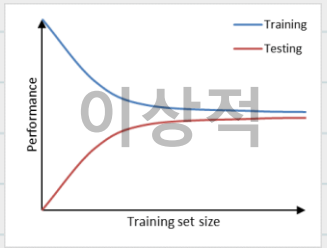
2-1-2. 다변수

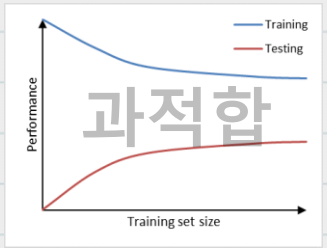
2-1-2-7. 학습 곡선 (Learning Curve)

작업 수행에 따른 학습 성능 변화를 도식화한 곡선







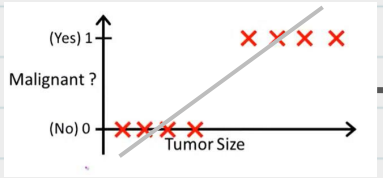


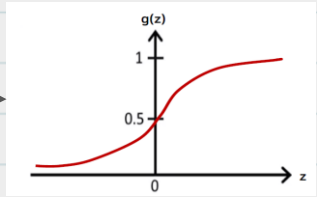
2-2. 분류 분석

2-2-1. 이진 분류

2-2-1-1. 가설 함수 (Hypothesis Function)

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$





기존 선형 회귀 방정식을 가설 함수로 사용하면 분류에 왜곡이 생기고, 값의 범위가 0과 1 사이를 벗어남
따라서, 로지스틱 회기 함수가 필요함

hθ(x)	$\theta^T x$: 기존회귀방정식
g(z)	$\frac{1}{1+e^{-z}}$: 로지스틱 함수, sigmoid 함수
g(hθ(x))	$\frac{1}{1+e^{-\theta^T x}}$: 로지스틱 함수에 회귀식 대입 → 가설함수 h(x)

2-2. 분류 분석

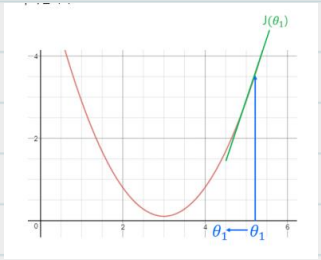
2-2-1. 이진 분류

2-2-1-2 . 손실 함수 (Loss Function)

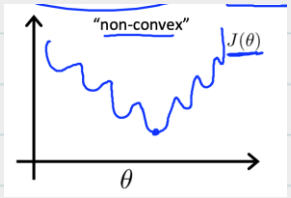
기존 회귀 분류는 로지스틱 함수(sigmoid)가 없어 손실 함수가 무조건 볼록 함수였지만,

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

손실 함수



분류 분석은 로지스틱 함수(sigmoid)로 인해 손실 함수가 볼록 함수가 아님



$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

새로운 손실 함수

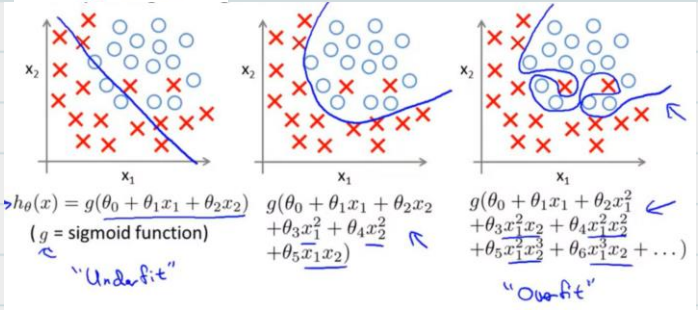
2-2. 분류 분석

2-2-1. 이진 분류

2-2-1-3. 학습 알고리즘 (Learning Algorithm)

회귀 분석과 동일

2-2-1-4. 과적합/과소 적합 (Overfitting/Underfitting)



회귀 분석과 마찬가지로 특성의 개수를 줄이거나, 손실 함수에 정규화 변수 λ를 추가하므로 과적합 완화

$$min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2$$

2-2. 분류 분석

2-2-1. 이진 분류

2-2-1-5. 평가

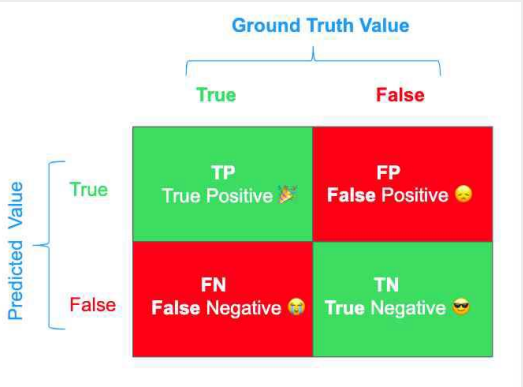
정확도 (Accuracy) : 전체 데이터 중 맞힌 것의 개수 비율

오차 행렬 (Confusion Matrix)

$$\text{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\text{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$



예시) 종양의 양성/악성을 판단하는 분류 분석 (y=1 : 악성 / y=0 : 양성)

2-2. 분류 분석

2-2-1. 이진 분류

2-2-1-5. 평가

이진 분류에서 두 집단의 크기가 동일하지 않은 경우가 많음 (차이가 클 경우 Skewed Class라고 칭함)

따라서, 정확도가 정확한 성능을 예측하지 못함

예시) 종양의 양성/악성을 판단하는 분류 분석 (y=1 : 악성 / y=0 : 양성)

전체 데이터의 0.5%만 악성 종양 (Skewed Class)

모델 학습 결과 정확도 0.99 -> 항상 양성이라고 판단하는 것이 더 정확도가 높음

y=1인 데이터가 비교적 작을 때, 이 케이스를 잘 예측하는 것이 중요 -> **Precision & Recall**

Precision : 악성이라고 판단한 종양 중 실제 악성의 비율

Recall : 실제 악성 종양 중 악성이라고 판단한 비율

2-2. 분류 분석

2-2-1. 이진 분류

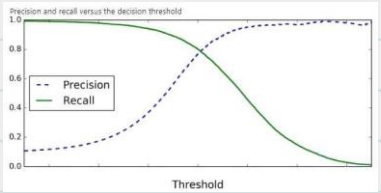
2-2-1-5. 평가

악성이라고 판단한 종양이 무조건 악성이어야 하는 경우

-> Higher Precision, Lower Recall

악성 종양을 하나도 놓치고 싶지 않은 경우

-> Higher Recall, Lower Precision



두 가지 평가 지표를 병합한 지표 : F1 Score

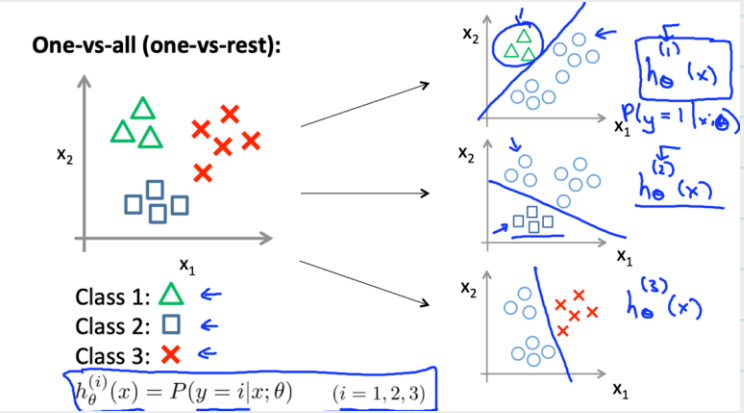
$$F_1 \text{ Score: } 2\frac{PR}{P+R}$$

	Precision(P)	Recall (R)	Average	F_1 Score
Algorithm 1	0.5	0.4	0.45	0.444
Algorithm 2	0.7	0.1	0.4	0.175
Algorithm 3	0.02	1.0	0.51	0.0392

2-2. 분류 분석

2-2-2. 다중 분류

2-2-2-1. 분류 방법



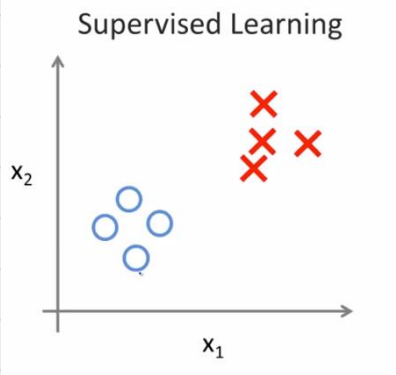
종류는 세 가지지만, 이진 분류 문제로 바꿔서 문제 해결

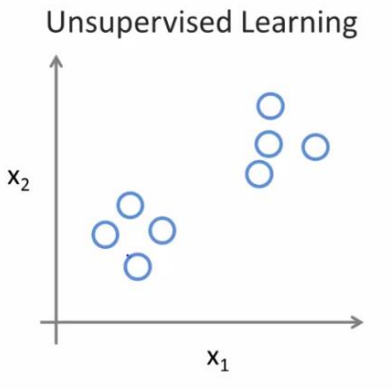
각 개체는 세 값 중 최대 값이 나온 클래스로 분류

3. 비지도 학습 (Unsupervised Learning)

비지도학습 : 학습하는 데이터에 정답이 포함되어 있지 않은 경우

-> 우리가 이 데이터로 무엇을 할지, 데이터가 무엇인지 알 수 없다!





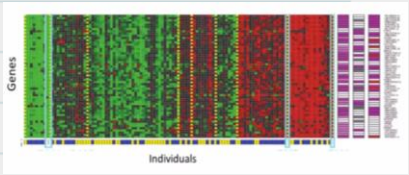
3-1. 군집화 (Clustering)

예시1) 구글 뉴스

구글에 있는 수많은 뉴스 기사 중 비슷한 소재의 기사들끼리 묶기

예시 2) 유전자 배열

환자들의 유전자 배열 데이터 중 유사한 유전자 배열의 환자들끼리 묶기



예시 3) 소셜 네트워크 분석

어떤 친구에게 이메일을 많이 보내는지, 페이스북 친구 등의 데이터를 통해

친구를 친한 친구 그룹, 알 거 같은 친구 그룹 형성

예시4) 시장 세분화

고객 데이터를 바탕으로 시장 세분화

3-2. Non-Clustering

예시1) 칵테일 파티에서 여러 명이 동시에 대화해서 의사소통 어려움

-> 여러 위치에 있는 마이크로 전체 음성 녹음

-> 각 마이크의 녹음된 음성을 분석해 개인들의 음성 추출

05

파이썬 라이브러리

part 2(scikit-learn)

진행자 : 새미(학습채널)

Scikit-learn 이란 ?

예측 데이터 분석을 위한 단순하고 효율적인 python 라이브러리

분류 (Classification), 회귀 (Regression), 군집화 (Clustering), 차원 축소 (Dimensionality Reduction), 모델 선택 (Model Selection), Preprocessing (전처리) 모두 가능하여 머신러닝 공부에 빠질 수 없는 도구 !

모두 scikit-learn 실습을 위해 네이버 카페 '세미나 자료실' 게시판에서 오늘 세미나 자료를 다운 받아주세요 !


실습 중 ☺

06

스터디 조별 회의

진행시간 : 30분

스터디 조별 회의

1. 밥 먹을 날짜 정하기
2. 스터디 주제에 대해 얼만큼 알고 있는지 얘기하기
3. 스터디를 어떻게 진행할 지 토의하기
4. 오늘 스터디조 게시판에 회의 내용 올리기

※ 스터디 인증은 일요일(9/12) 11시 59분까지.
조원 모두 인증한 사진 + 배운 내용 요약해서 카페에 게시.

수고하셨습니다 ☺