

Dacon Analysis Study

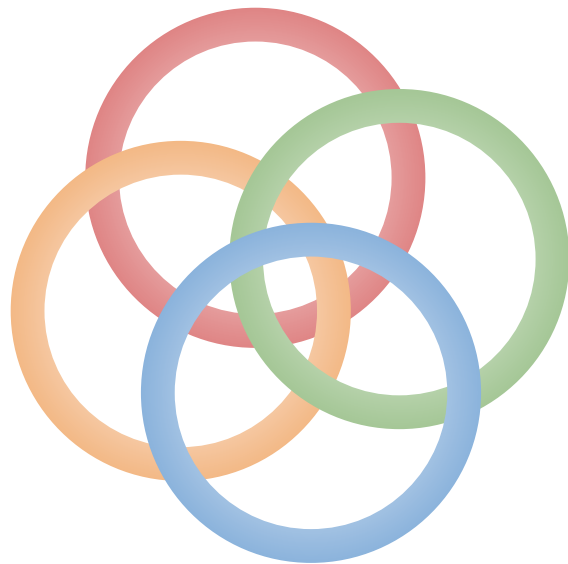
Chap 2) 반도체 박막 두께 분석

17 강신현

17 김건우

17 송원진

17 신도현



목차



1. 박막 소개 및 데이터 분석



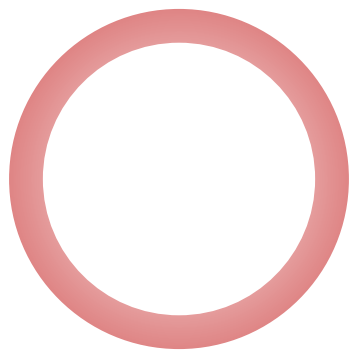
2. 데이터 전처리



3. 모델 구축 및 검증



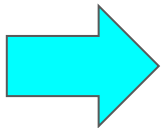
4. 성능 향상을 위한 방법



1. 박막 소개 및 데이터 분석

1. 박막 소개 및 데이터 분석

문제 정의

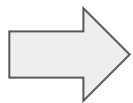


데이터 분석

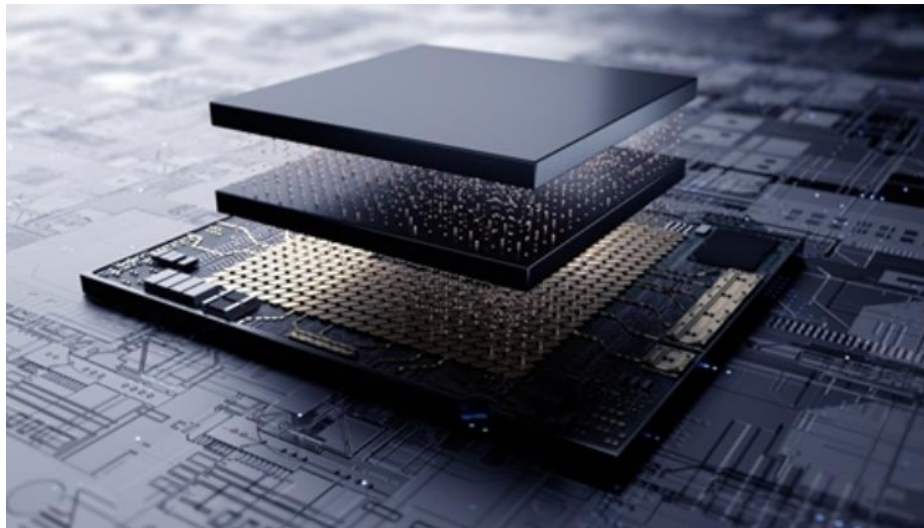
1.1 문제 정의

반도체를 수직으로 수십, 수백 층 적층하는
3차원 공정 연구가 활발하게 진행중입니다.

이때 공정 과정에서 박막의 결함으로 박막 두께
가 달라져 균일도 저하



반도체 성능에 문제가 발생



[출처] 네이버블로그 <https://blog.naver.com/midashan/222062900764>

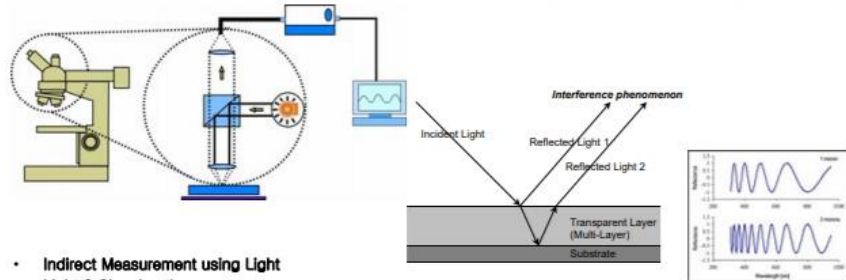
1.1. 문제 정의

박막의 두께를 측정하는 것이 중요하다.

널리 이용되는 방법 반사율 스펙트럼 분석

➡ 해당 분야의 전문 지식을 가진 전문가가 필요하며

고사양의 컴퓨터 자원이 필요하다.



- Indirect Measurement using Light

- Light & Signal path

Visible light => Thin Layers => **Reflection from Surface & interfaces** => Fiber Probe => Spectrophotometer => Wavelength Decomposition => CCD => A/D Conversion => PC => Software

- Thickness by Spectrum Fitting

Measured Spectrum shows sine shapes or valleys in Wavelength Domain depending on Film Thickness and N & K

Optimize Film Thickness and N & K by fitting Computed Spectrum to Measured one as varying Film Characteristics

13

[출처]

https://www.4science.net/goods/detail.do?glt_no=3169

이 한계를 극복하기 위해 머신러닝 및 딥러닝 기술을 이용해

박막의 반사율 스펙트럼 데이터를 분석해 소자 두께를 알아내는 것 목표

1.1.(1) 평가척도

파장에 따른 반사율 스펙트럼을 학습해 4개의 박막층 두께를 예측하는 문제

$$MAE(Mean Absolute Error) = \frac{1}{n} \sum_{i=1}^n |y_i - y'_i|$$

모델을 통해 예측한 4개 층의 두께와 실제 두께의 평균 절대 오차가
얼마나 작은지를 놓고 모델 성능을 판단

1.1.(1) 평가척도

	실제 값	예측 값
layer_4 : 이산화규소	ex) 0.113	ex) 0.133
layer_3 : 질화규소	ex) 0.651	ex) 0.649
layer_2 : 이산화규소	ex) 0.344	ex) 0.331
layer_1 : 질화규소	ex) 0.452	ex) 0.440

$$\text{MAE} = \frac{1}{4} \times (|0.452-0.440|+|0.344-0.331|+|0.651-0.649|+|0.113-0.133|) = 0.01175$$

1.1.(2) 접근 방식

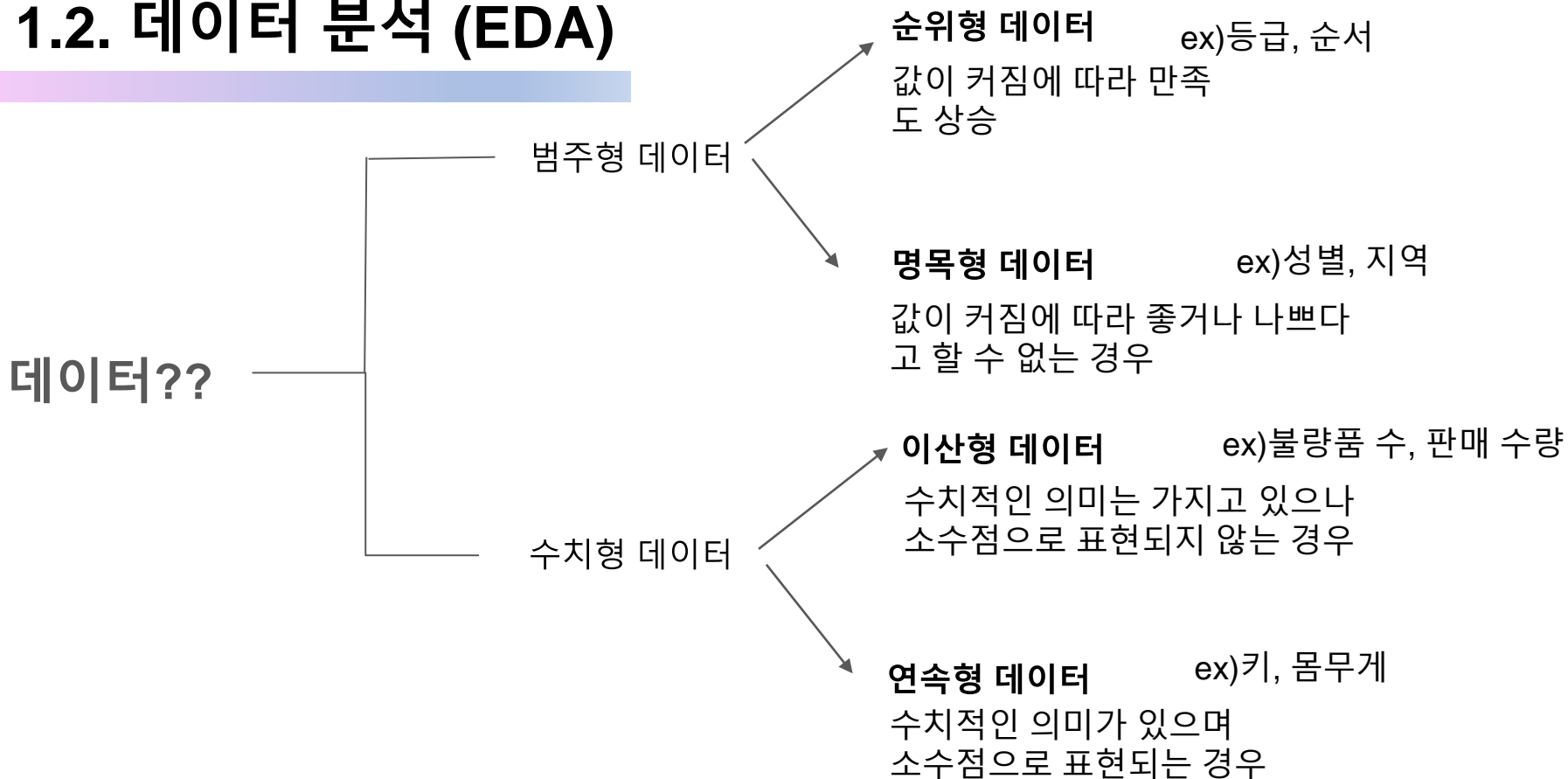
- 박막의 두께를 예측하기 위해 빛의 반사율을 사용해야 함

-> 반사율을 계산하기 위해서는 입사각 및 편광 계산, 굴절률 계산, 물질에 의한 반사율

A1	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
	layer_1	layer_2	layer_3	layer_4																	
1	10	10	10	10	0.254551	0.258823	0.254659	0.252085	0.247678	0.253614	0.246511	0.259407	0.260862	0.242524	0.25387	0.245156	0.245548	0.255501	0.228948	0.228632	0.225802
2	10	10	10	10	20	0.205062	0.225544	0.217758	0.202169	0.199633	0.20738	0.191318	0.195369	0.200536	0.197588	0.198726	0.191803	0.199625	0.206465	0.182836	0.193341
3	10	10	10	10	30	0.189196	0.165869	0.177655	0.156822	0.175094	0.177755	0.157582	0.158885	0.156911	0.166162	0.148831	0.14495	0.151362	0.14511	0.159201	0.139296
4	10	10	10	10	40	0.131003	0.120076	0.138975	0.117931	0.130566	0.131262	0.126962	0.134453	0.106717	0.127309	0.099958	0.112908	0.106853	0.108288	0.101393	0.094074
5	10	10	10	10	50	0.091033	0.086893	0.108125	0.080405	0.105917	0.077083	0.097895	0.086765	0.078676	0.075729	0.086023	0.070649	0.078957	0.072772	0.069867	0.080523
6	10	10	10	10	60	0.064314	0.082353	0.073697	0.059653	0.071695	0.049325	0.056011	0.049143	0.061499	0.042358	0.049665	0.044624	0.050232	0.044264	0.05385	0.032225
7	10	10	10	10	70	0.044233	0.05058	0.049386	0.054143	0.059369	0.038195	0.052949	0.048194	0.045218	0.050587	0.055258	0.027519	0.030759	0.049205	0.028949	0.042312
8	10	10	10	10	80	0.030395	0.049087	0.051467	0.038069	0.033029	0.030484	0.050531	0.042321	0.051049	0.058052	0.035979	0.060488	0.04667	0.045049	0.065948	0.069264
9	10	10	10	10	90	0.04644	0.033531	0.05273	0.065774	0.062442	0.050097	0.046529	0.067729	0.069011	0.078237	0.078124	0.071469	0.060929	0.075576	0.073837	0.083285
10	10	10	10	10	100	0.070731	0.05913	0.083172	0.07862	0.070029	0.087893	0.071019	0.101433	0.077797	0.108224	0.092005	0.111433	0.093301	0.105488	0.124976	0.123335
11	10	10	10	10	110	0.087597	0.099092	0.113232	0.096118	0.103946	0.117888	0.118977	0.123853	0.140365	0.148711	0.140647	0.152344	0.16136	0.158495	0.157463	0.173605
12	10	10	10	10	120	0.133915	0.123353	0.142951	0.154168	0.141391	0.16283	0.164898	0.180446	0.165659	0.168438	0.179093	0.183578	0.204693	0.210141	0.214412	0.207011
13	10	10	10	10	130	0.175389	0.163683	0.180982	0.19606	0.182814	0.197864	0.208551	0.202096	0.22589	0.211688	0.244174	0.236667	0.238435	0.238112	0.253951	0.267077
14	10	10	10	10	140	0.194168	0.211107	0.205971	0.21044	0.235742	0.24823	0.238678	0.261129	0.244792	0.254573	0.280458	0.274813	0.283874	0.272785	0.281389	0.285044
15	10	10	10	10	150	0.240196	0.240986	0.24413	0.254554	0.259723	0.280683	0.286207	0.288268	0.279551	0.299499	0.290515	0.294334	0.309887	0.311967	0.327861	0.343051
16	10	10	10	10	160	0.283009	0.279989	0.285432	0.279384	0.293673	0.315918	0.319731	0.323484	0.307325	0.31256	0.325948	0.335758	0.35358	0.34259	0.344257	0.357095
17	10	10	10	10	170	0.312391	0.301437	0.319575	0.313917	0.326146	0.343689	0.331584	0.347792	0.34504	0.33913	0.35114	0.369649	0.377573	0.377534	0.360711	0.369682
18	10	10	10	10	180	0.340762	0.33335	0.336038	0.336424	0.343459	0.345301	0.358118	0.366525	0.363641	0.373494	0.358716	0.374501	0.386374	0.36933	0.372188	0.390103
19	10	10	10	10	190	0.353118	0.368153	0.361643	0.349262	0.37368	0.365197	0.357716	0.376008	0.367576	0.375573	0.374669	0.386433	0.381937	0.373761	0.378535	0.388312
20	10	10	10	10	200	0.379767	0.37941	0.370593	0.376895	0.369809	0.37161	0.368238	0.382476	0.370052	0.382255	0.379343	0.394322	0.382597	0.398675	0.377329	0.399642
21	10	10	10	10	210	0.374522	0.374845	0.389308	0.375322	0.389984	0.381685	0.377107	0.375776	0.392926	0.382596	0.372674	0.380255	0.371368	0.370084	0.365452	0.380888
22	10	10	10	10																	

- layer_1 ~ layer_4 열 : 각 박막의 두께
- 헤더(0~255) 행 : 파장 역수와 스펙트럼값 등으로 구성

1.2. 데이터 분석 (EDA)



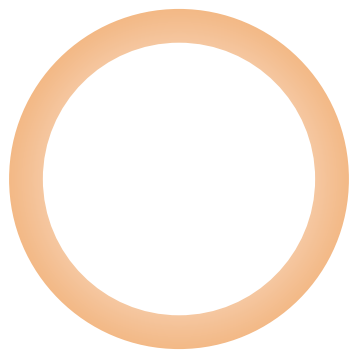
1.2. 데이터 분석 (EDA)

분석할 데이터의 구조, 변수의 타입 및 크기 등을 탐색하는 것

데이터에서 의미 있는 패턴과 현상을 쉽게 알아낼 수 있음

그래프 및 통계적인 방법으로 데이터를 직관적으로 바라볼 수 있음

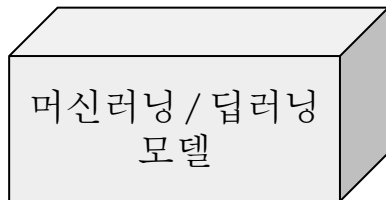
결측치(NaN)나 이상치(Outlier) 같이 문제 있는 부분을 발견해
미리 대처 가능



2. 데이터 전처리

Data Preprocessing

Bad Data
(ex. 낮은 품질, 노이즈 많음)



잘못된 결과를 도출

데이터 전처리의 이유 : 위와 같은 문제를 사전 방지

올리는 과정

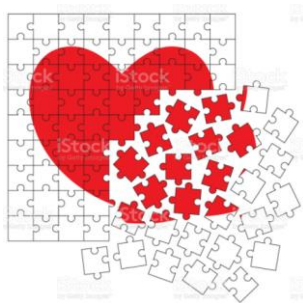
→ 데이터의 품질을

how to ?

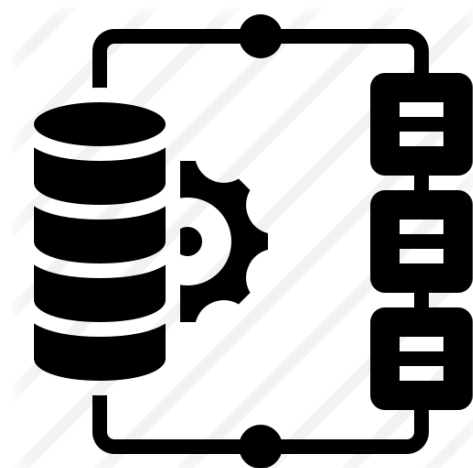


$$\begin{bmatrix} x_0 & x_0 & \dots & x_0 \\ x_1 & x_1 & \dots & x_1 \\ | & | & \dots & | \\ x^1 & x^2 & \dots & x^m \\ | & | & \dots & | \\ x_n & x_n & \dots & x_n \end{bmatrix}$$

Data Vectorization



Data Cleaning

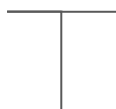


Data Integration / Reduction

결측치(불완전 data) 처리

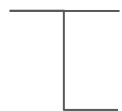
TMI Data → pandas 라이브러리 함수 이용

결측치
행)



제거 (단일 값 or 해당 전체

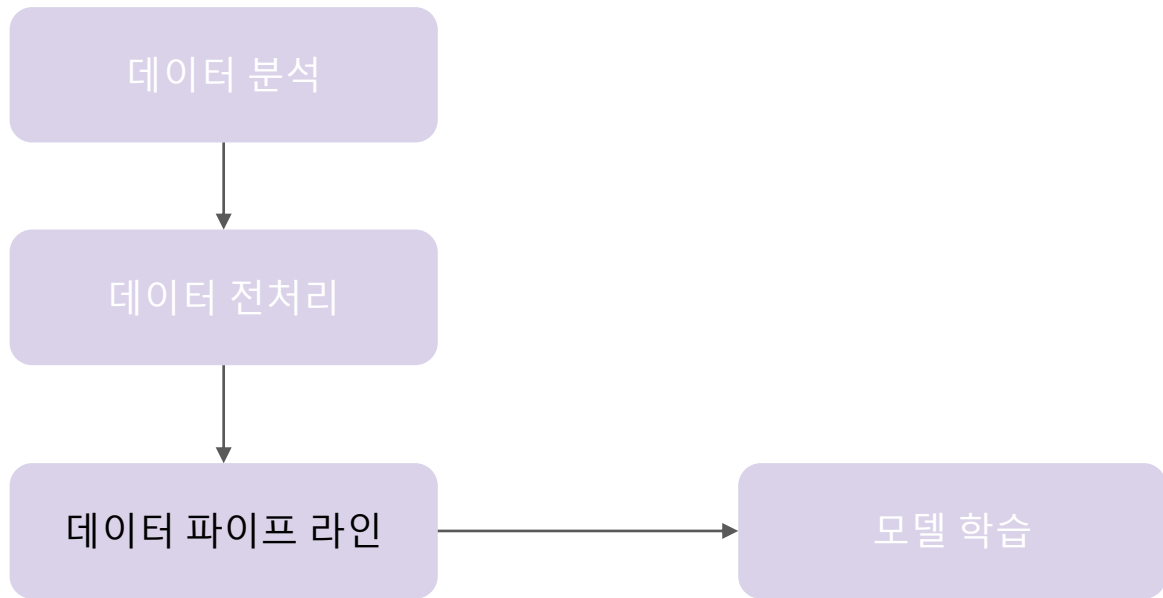
평균(mean)



채우기

중앙값(median)

데이터 파이프라인(Data Pipeline)



데이터를 한 장소에서 다른 장소로 차례대로 전달하는, 데이터로 구성된 일련의 시스템

in ML(Machine Learning)...

검증 데이터 셋 사용

(모델이 설정한 하이퍼파라미터에 따라 데이터의 크기, 예측값, 입력 등 얼마나 잘 학습했는지 추정하고 모델 속성을 추정하기 위해)

- 학습 데이터 셋과 독립적으로 구성
- 모델의 하이퍼파라미터를 최적화 하기 위해 사용

종류

K-Fold
(k겹 교차 검증)

Stratified K-Fold

ShuffleSplit



Original Set

Training

Testing

Training

Validation

Testing

데이터를 섞어 다시 저장 시 인덱스 재정렬을 위한 작업을 진행합니다.

```
layers = [['layer_1', 'layer_2', 'layer_3', 'layer_4'], \
          [str(i) for i in np.arange(0,226) tolist()]]
```

```
layers = list(chain(*layers))
```

tolist() : 같은 level끼리 있는 data끼리 묶어준다



```
import pandas as pd
df = pd.DataFrame({'value': [1,2,3], 'test': ['a', 'b', 'c']})
df.values.tolist()
```

```
[[1, 'a'], [2, 'b'], [3, 'c']]
```

데이터를 섞어 다시 저장 시 인덱스 재정렬을 위한 작업을 진행합니다.

```
layers = [['layer_1', 'layer_2', 'layer_3', 'layer_4'], \
          [str(i) for i in np.arange(0,226).tolist()]]
```

```
layers = list(chain(*layers))
```

from itertools import chain

간단히 말하면 리스트(lists/tuples/iterables) 를 연결하는 것

```
letters = ['a', 'b', 'c', 'd', 'e', 'f']
```

```
booleans = [1, 0, 1, 0, 0, 1]
```

```
decimals = [0.1, 0.7, 0.4, 0.4, 0.5]
```

```
print list(itertools.chain(letters, booleans, decimals))
```

결과 : ['a', 'b', 'c', 'd', 'e', 'f', 1, 0, 1, 0, 0, 1, 0.1, 0.7, 0.4, 0.4, 0.5]

2.3.3 커스텀 데이터 클래스 ?



미니 배치 학습



: 일정량씩 데이터를 배치로 묶어주는 과정이 필요하다

지도학습: 학습데이터 + 정답 데이터를 묶어서 구성해주는 과정이 필요!

Pytorch



`from torch.utils.data import Dataset`

`__init__()` : 필요한 변수, 함수를 선언

`__len__()` : 데이터의 전체 길이(총 sample 개수)를 return

`__getitem__()`: 데이터셋에서 해당 idx의 sample을 return

class **PandasDatatset**(Dataset) : 임의 클래스 생성

: CSV 파일을 입력받은 후 numpy로 변환된 학습 데이터 -> index로 가져오는 기능

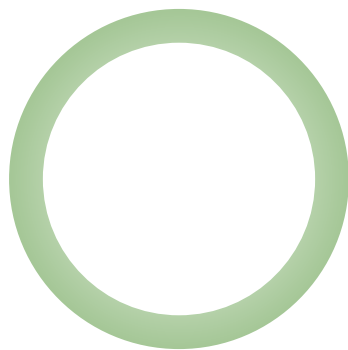
CSV 파일 입력받기



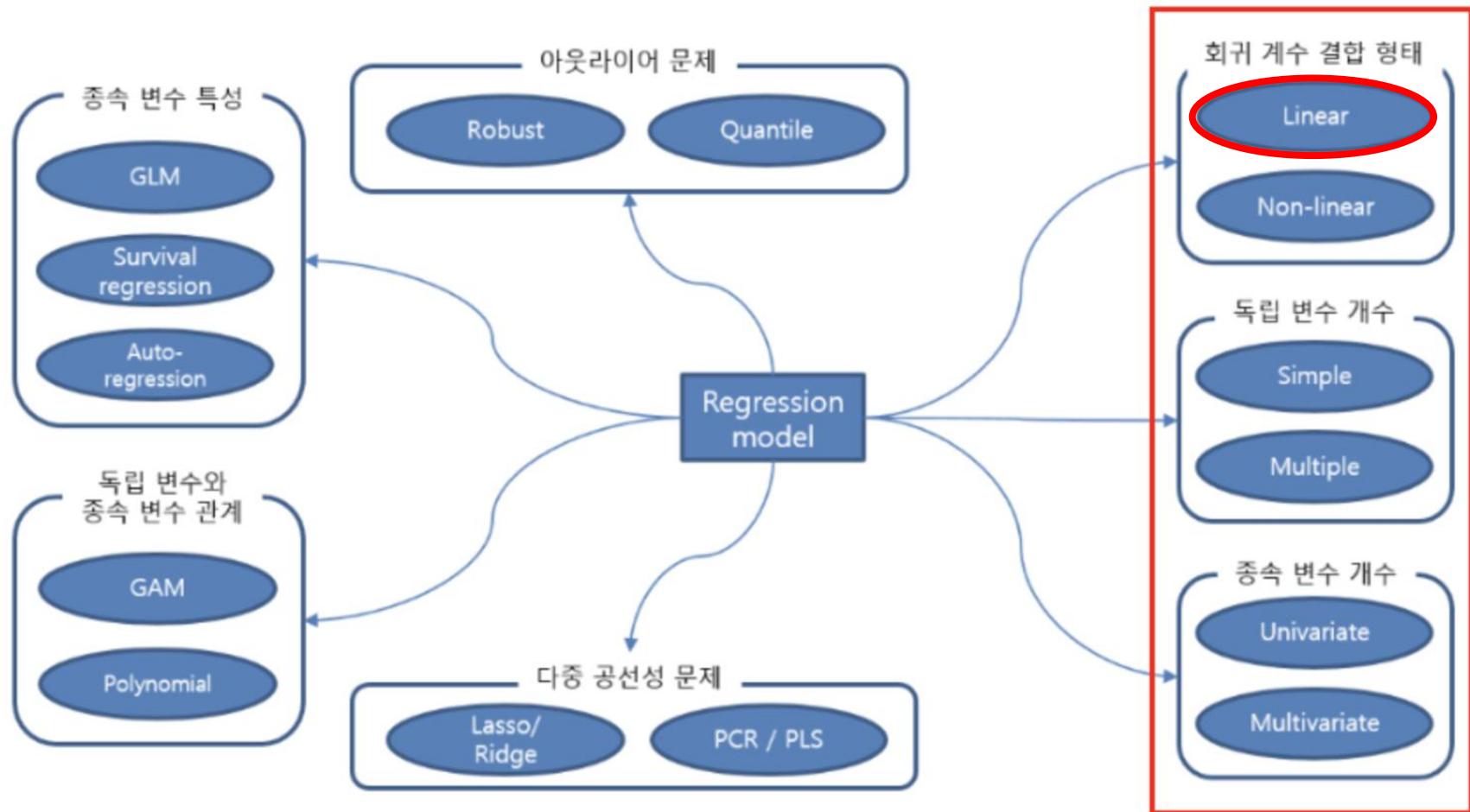
numpy로 변환된
학습데이터



index로 가져오기



3. 모델 구축 및 검증



모델 탐색

독립 변수 개수 ?

: 반사율 스펙트럼 -> 다중(Multiple)

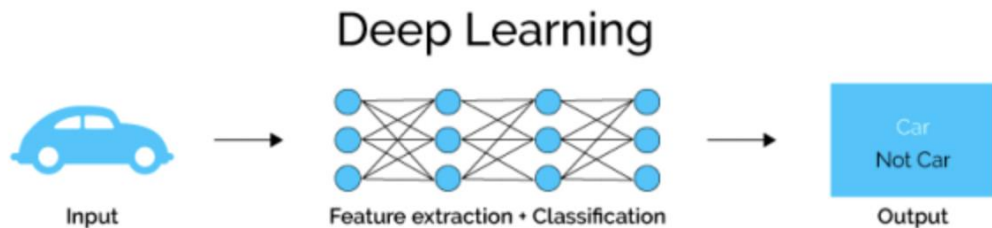
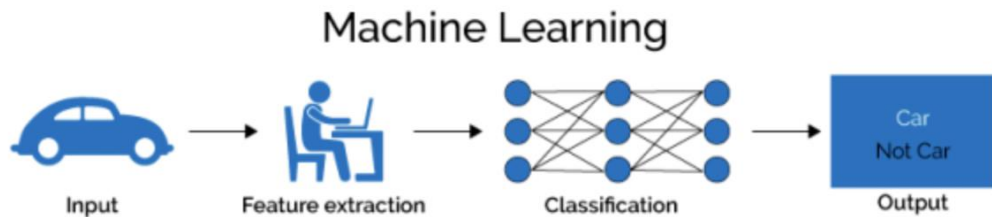
종속 변수 개수 ?

: 박막 두께 -> 다변량
(Multivariate)

즉, 다변량 다중선형회귀



머신러닝 모델 vs 신경망 모델 (in 회귀)



머신러닝 (XGBoost)

- 여러 프로세스

딥러닝 (MLP-다층 퍼셉트론)

- 한 번의 프로세스
- 데이터가 많을수록 우수한 성능
- 중요도에 따른 특징 관리

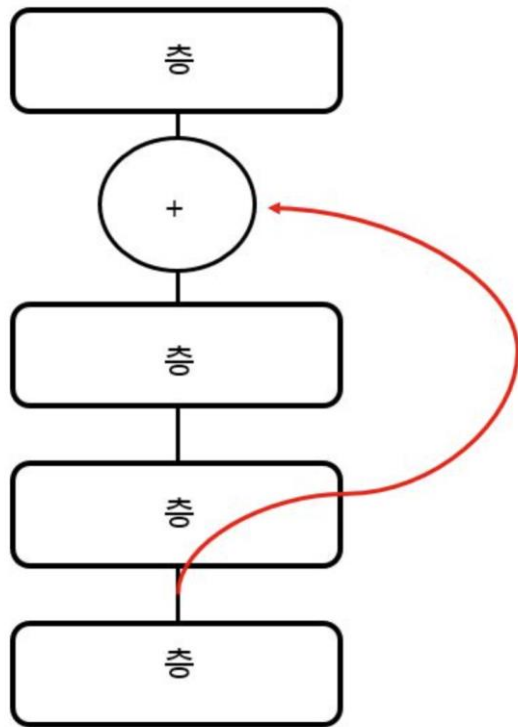
검증1 - 잔차 연결

하위 층의 출력을 상위 층의 입력으로 사용.

하위 층의 출력이 상위 층의 활성화 출력에 연결되는 것이 아니고 더해진다. 크기가 다르면 선형 변환을 사용하여

하위 층의 활성화 출력을 목표 크기로 변환해야한다.

일반적으로 10개층 이상을 가진 모델에 잔차연결을 추가하면 도움이된다



검증1 - 잔차 연결

problem1 - 표현 병목

히든 레이어의 히든 유닛 수가 너무 적어지면 앞선 레이어의 출력을 다 받아낼 만큼의 충분한 용량을 갖기 힘들.

이런 경우 기울기(gradient)가 잘 전달되어도 용량 자체가 작아서 다음 레이어에 전달할 수 있는 정보가 줄어들게 되어 표현 병목이라는 현상이 발생.

즉 잔차 연결을 통해 본래 데이터에 더 가까운 값을 더해줌으로써 손실을 만회해준다.

problem2 - 그래디언트 소실 문제

심층 신경망을 훈련하는 데 사용되는 핵심 알고리즘인 역전파는 출력 손실에서 얻은 피드백 신호를 하위 층에 전파한다.

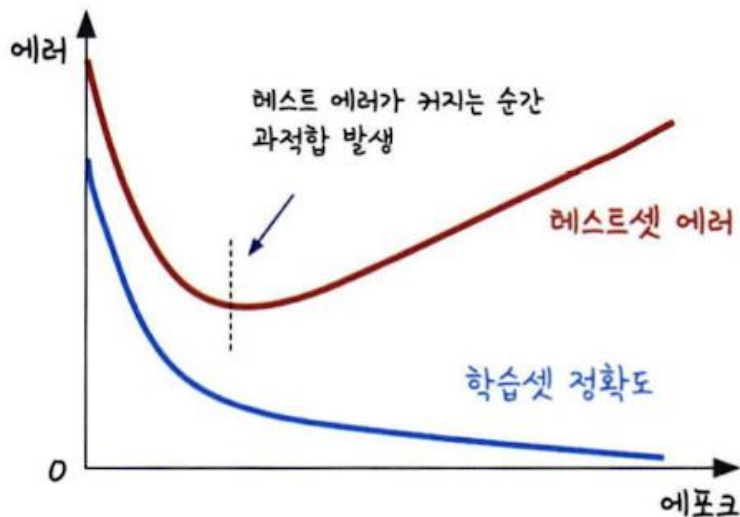
피드백 신호가 깊이 쌓인 층을 통과하여 전파되면 신호가 아주 작아지거나 완전히 사라질 수도 있다. 이렇게 되면 네트워크가 훈련되지 않는데 이런 것을 그래디언트 소실 문제라고 한다.

이 문제는 심층 신경망과 순환 신경망에서 모두 나타나며 양쪽 모두 피드백 신호가 일련의 긴 연산을 통과하여 전

검증2 - 과적합 발생 여부 확

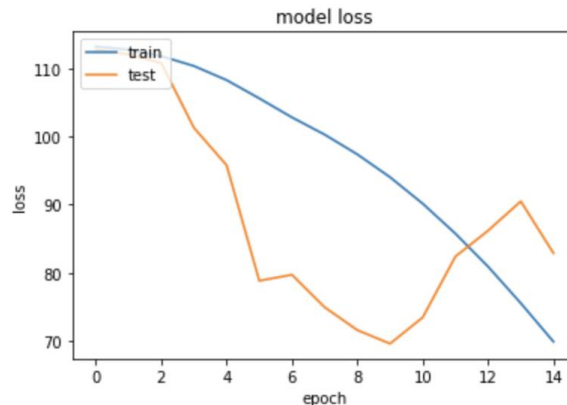
인

(지난 겨울 스터디 내용)



```
import matplotlib.pyplot as plt

plt.plot(results.history['loss'])
plt.plot(results.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



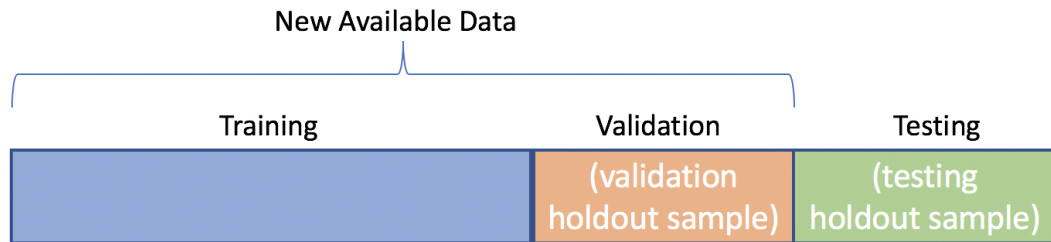
검증3 - 검증데이터 활용

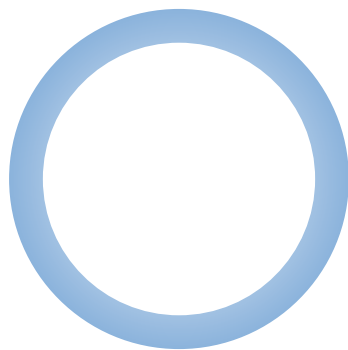
```
model.fit(x_train,y_train,epochs=20,batch_size=1024,  
validation_split=0.2,callbacks=[early_stop])
```

별도로 존재하는 검증 데이터를 주는 것이 아니라
X_train과 y_train에서 일정
비율을 분리하여
이를 검증 데이터로 사용.

```
- val_loss: 76.1072 - val_mae: 76.1072  
val_loss: 67.5421 - val_mae: 67.5421
```

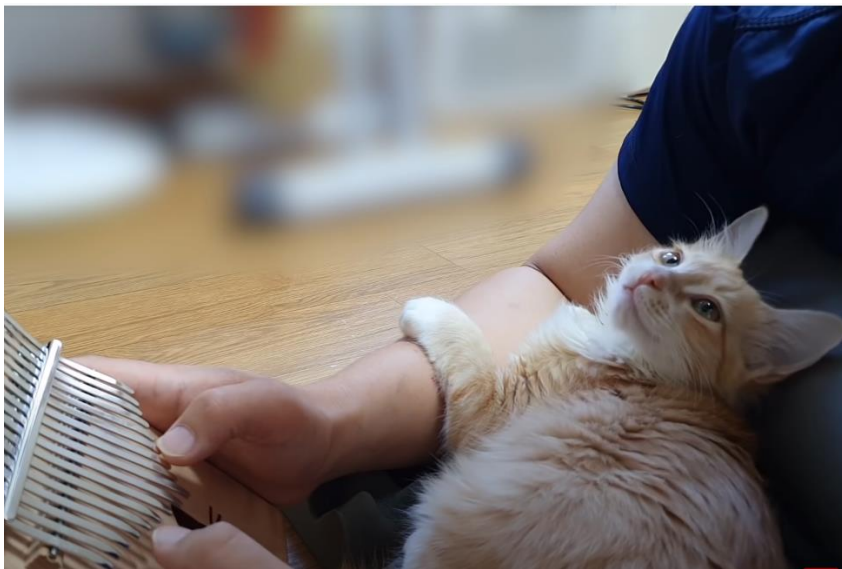
훈련 자체에는 반영되지 않고
훈련 과정을 지켜보기 위한
용도로 사용





4. 성능향상을 위한 방법

4.1 성능 향상을 위한 방법



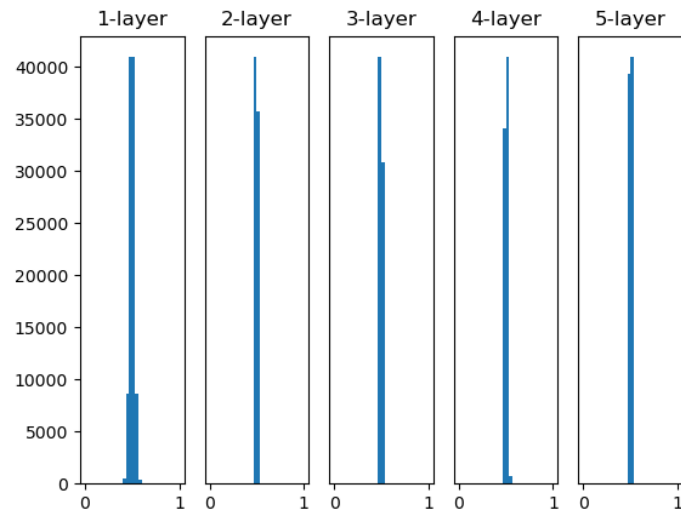
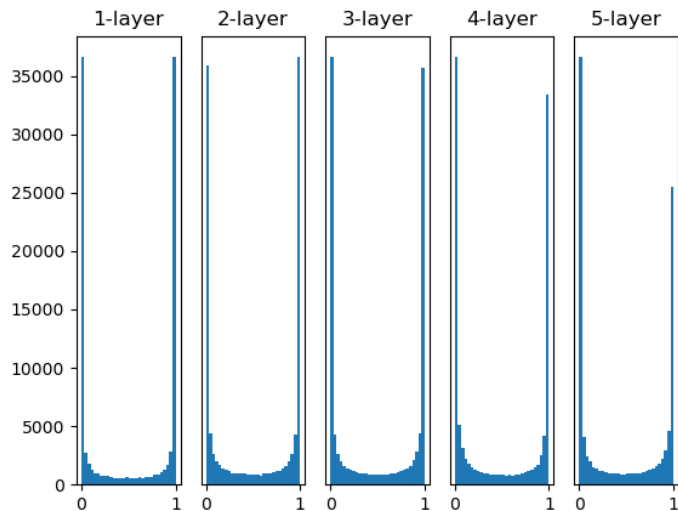
정규화 기법 적용하기

추가 실험 하기

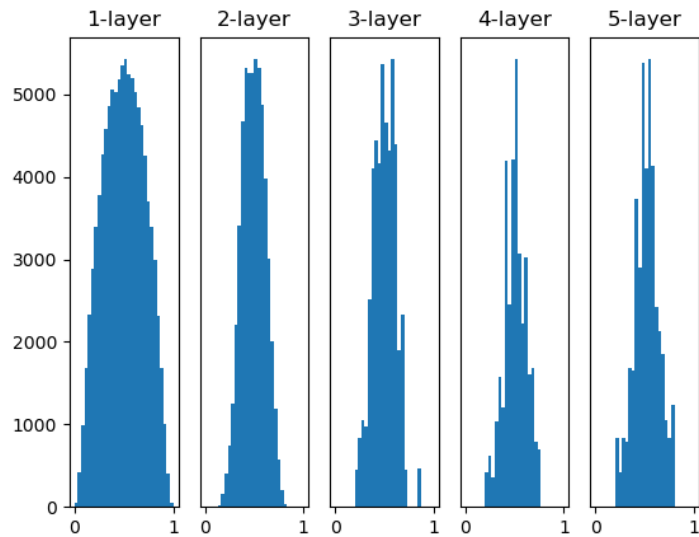
앙상블

4.1.1 정규화 기법 적용 - 초깃값과 기울기 소실문제

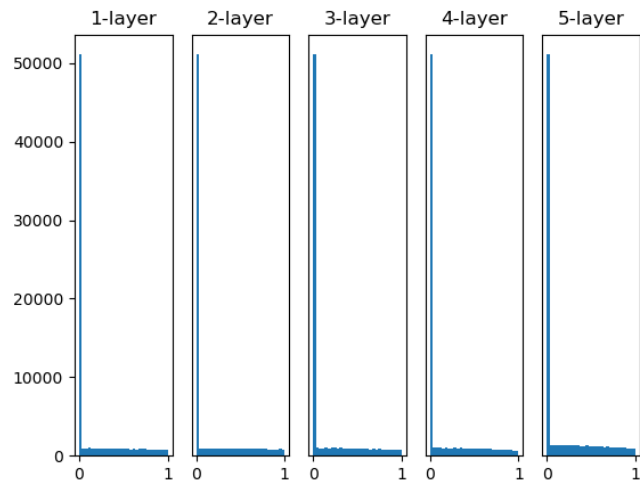
초깃값..?



4.1.1 정규화 기법 적용 - 초깃값과 기울기 소실문제(해결1)



<Xavier 초깃값>



<HE 초깃값>

4.1.1 정규화 기법 적용- 초깃값과 기울기 소실문제(해결2)

굳이 초깃값을 정하지 않고 각 레이어에서 활성화를
적당히 퍼뜨리도록 강제한것이 '배치정규화'!



배치 정규화를 사용한 신경망의 예

$$\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^m x_i$$
$$\sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$$
$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

4.1.1 정규화 기법 적용



배치 정규화 사용시 장점

1. 가중치 초기값의 의존도 감소
2. 기울기 손실 문제 해결
3. 과적합 방지
4. 학습속도 개선

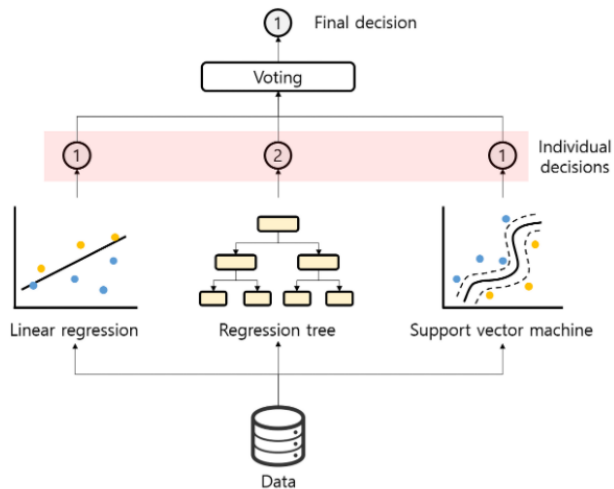
4.1.2 추가 실험하기



옵티마이저 및 스케줄러 조정

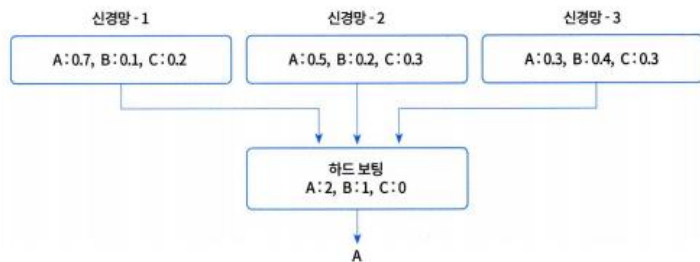
하이퍼파라미터(배치 크기, 은닉층 노드 개수, 레이어 구성) 조정

4.1.3 앙상블

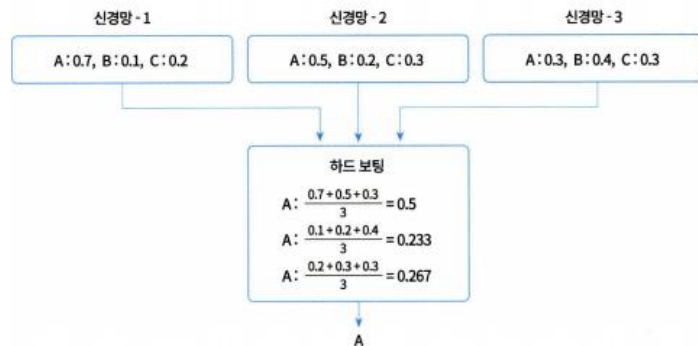


앙상블 학습(**Ensemble** Learning)이란 여러 개의 분류기를 생성하고
그 예측을 결합함으로써 보다 정확한 예측을 도출하는 기법.

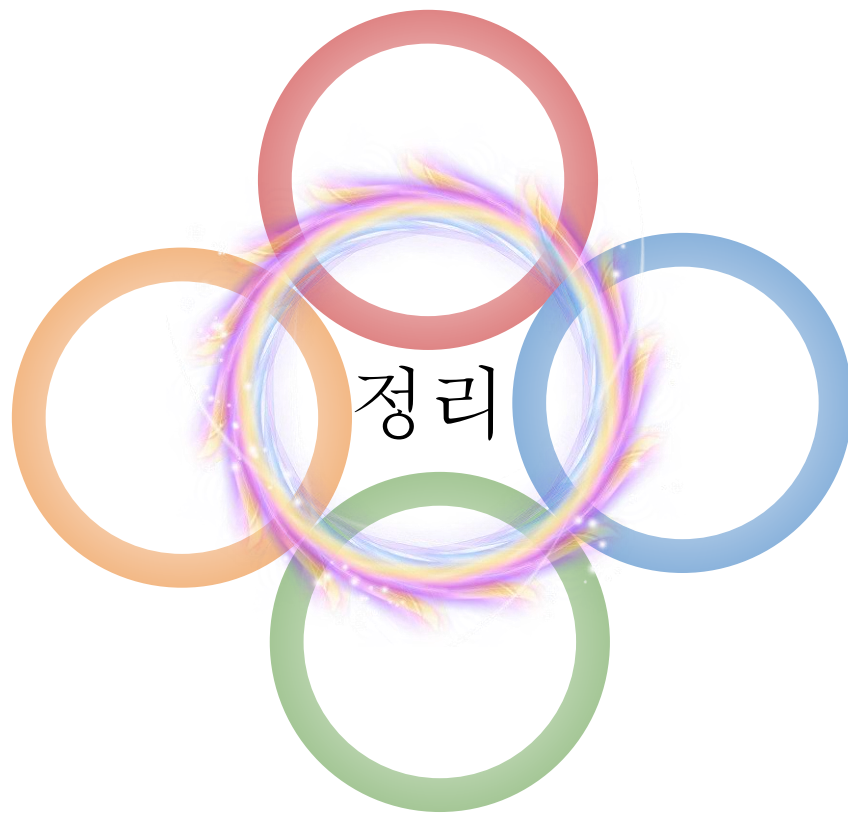
4.1.3 앙상블



하드 보팅(hard voting)



소프트 보팅(soft voting)



review)



- 1) 데이터 분석 및 해석
- 2) 데이터의 성질 파악 (데이터의 어떤 특성을 보이는지 확인)
- 3) 알고리즘 적용(어떤 모델을 사용할지)

실패 분석

- 1) XGBoost 머신러닝 모델로 구축해보려고 했는데 처음 다뤄보는거라 걸린 오류
- 2) 각 프로그램 간 문법호환 문제
- 3) 81만개의 데이터를 다루는데 있어 colab의 한계

- 1) 데이콘(Daicon) 사이트 내 '코드 공유', '토론' 게시글을 살펴보면서

다른 사람은 어떻게 풀었나, 어떤 이슈가 있었나 살펴 볼 것

- 1) 베이스라인 코드만 짜기 -> 파라미터 값을 변형해보며 코드를 응용 / 다양한 실험
- 2) colab에 국한된 코드 작성 말고 pytorch, keras, kaggle 등 연계의 필요성