

# Comparison of Consequence Estimation Model Using Machine Learning Technology

한양대학교  
박성현, 제무성\*



# 연구 배경 1

- ◆ 원자력계를 포함하여 전세계적으로 모든 산업 분야에 걸쳐 기계학습(Machine Learning)을 통한 다양한 시스템이 구축되고 있음
- ◆ 원자력계도 다양한 기계학습을 통한 연구를 시작하는 단계임
- ◆ 원자력연구원도 기계학습의 일환인 심층학습(deep learning)을 적용하여 다양한 연구를 수행 중
  - 중대사고 고속 예측
  - 선원항 고속 예측
- ◆ PSA의 장점은 다양한 사고 시나리오 도출을 통해 시스템의 리스크를 정량적으로 평가할 수 있음
- ◆ PSA 품질을 향상시키기 위해서는 주어진 시스템에서 발생 가능한 사고 시나리오를 가능한 한 모두 도출하여 정량화를 수행하는 것은 매우 힘들
  - 불확실성 분석을 수행할 시, 결말 분석의 수는 증폭
  - 다수기 분석을 수행할 시, 결말 분석의 수는 증폭
- ◆ 기존의 학습 데이터(방출 분율 등)를 통해, 새로운 분석의 결말(조기사망 등)을 예측할 수 있지 않을까?

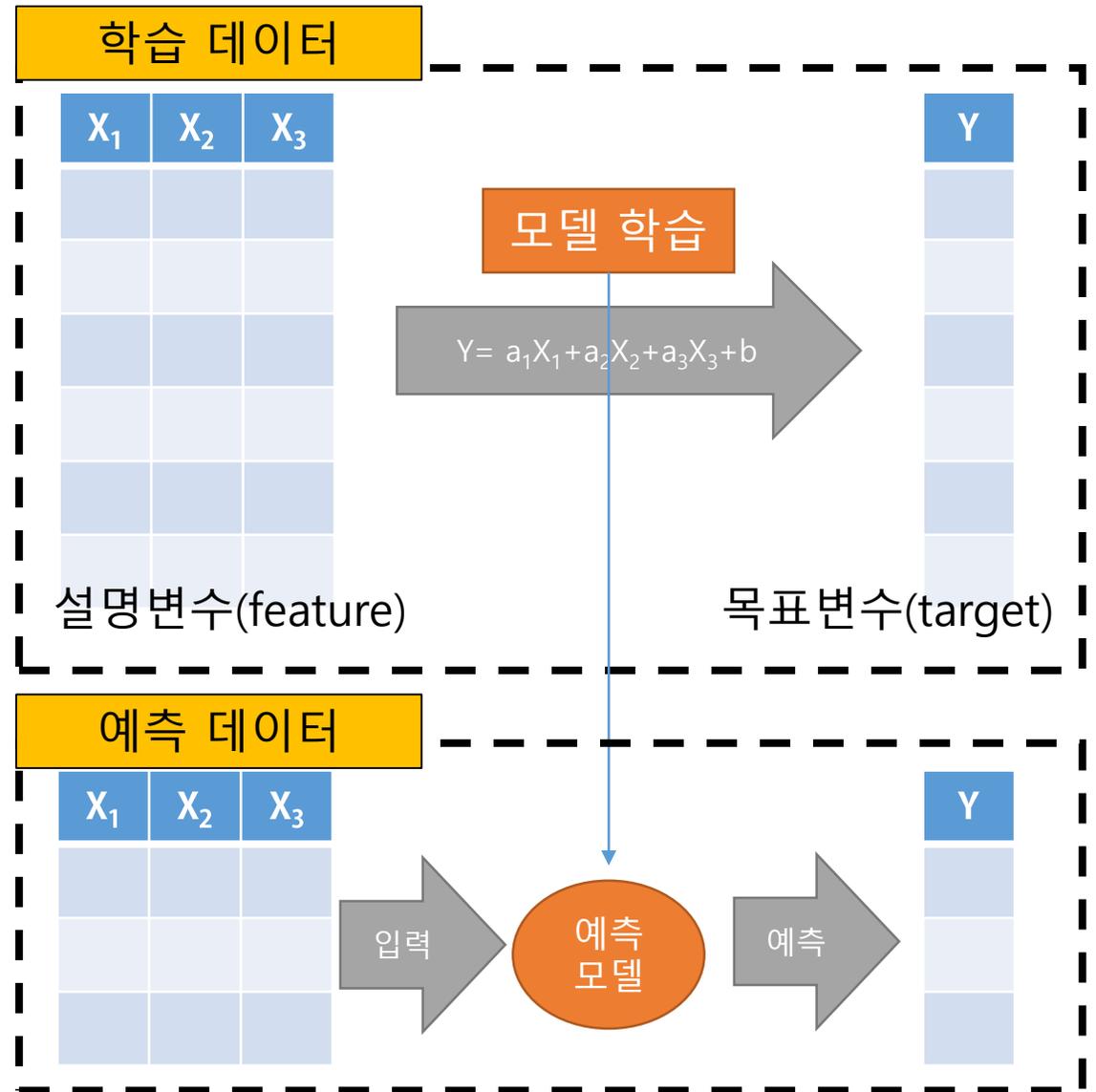
# 연구 배경 2

◆ 머신 러닝(Machine Learning)은 레이블(Label, 정답)이 있는 입력(Input) 데이터를 모델에 투입하면 머신러닝 모델이 입력 데이터와 정답 사이의 관계를 찾음

- 새로운 데이터를 입력했을 때 모델이 파악한 관계식을 적용하여 결과를 예측할 수 있음
- 정답이 주어지지 않은 경우 데이터 속에 숨어 있는 패턴이나 규칙을 알아내는 것도 가능함

## ◆ 지도 학습(Supervised Learning)

- 지도 학습은 학습과정에서 정답 Y가 주어짐. 즉 입력 데이터 X와 출력 데이터 Y를 모두 알고 있는 상태에서,  $Y=aX+b$ 와 같이 X와 Y 사이의 관계식을 알아내는 머신러닝 알고리즘을 말함
- 예측 목표가 되는 Y 변수를 목표 변수(target)라고 하고, 목표 변수를 예측하는데 사용되는 X 변수를 설명 변수(feature)라고 부름



# 연구 목표 및 수행 체계

## 1. 문제정의(Problem Definition)

- 데이터 분석의 목표: 방출 분율(release fraction)을 통해, 결말을 예측한다
- 분석의 전체 일정을 수립함

## 2. 데이터 확보 및 탐색

- MACCS를 통해, 9가지 핵종그룹의 방출 분율(0~1)에 따른 결말을 사전에 계산하여 데이터 확보(#10000개 데이터)
- 외부데이터를 포함하여, 결측값(missing data), 이상치(outlier), 데이터 분포, 상관관계 등을 파악함

## 3. 데이터 전처리

- 내외부 데이터 결합(#외부 데이터는 다른 선행연구 데이터 등이 있을 수 있음)
- 결측값, 이상치, 자료형 변환, 범주형 변수 인코딩, 속성 추가

## 4. 모델 학습

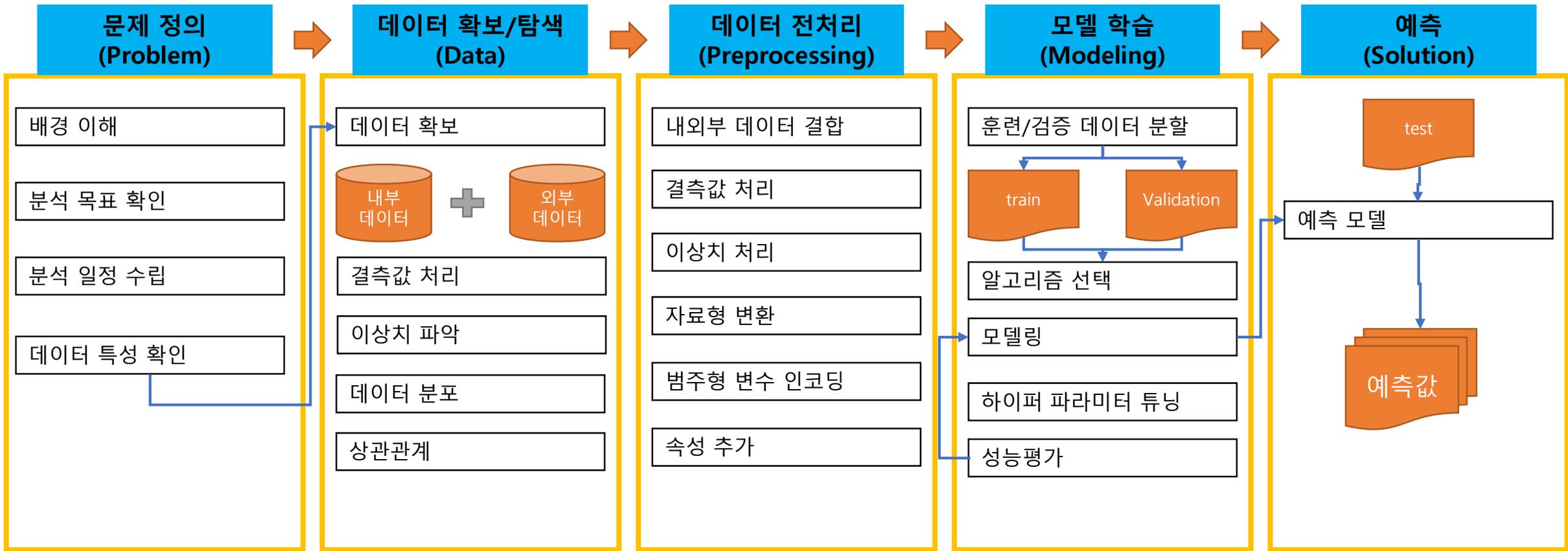
- 훈련/검증 데이터 분할(train/validation), 알고리즘 선택, 모델링, 튜닝, 성능평가

## 5. 예측

- 예측 모델을 통한 테스트, 예측값 산출

### #분석도구#

1. 학습 데이터 산출: MACCS ver. 4.0
2. Code Editor: Visual Studio Code ver. 1.57.1
3. Language: Python ver. 3.9.6
4. 주요 라이브러리:
  - Pandas:
  - Numpy:
  - Random:
  - Tensorflow:
  - Sklearn
  - matplotlib.pyplot
  - seaborn



# 1. 문제 정의(Problem Definition)

## 1. 배경 이해

- Level 3 PSA는 수많은 변수들의 영향으로 인해, 소외 주민들에 대한 건강영향을 평가하는 방법
- 다수기 Level 3 PSA를 수행하기 위해서는 분석의 수가 증폭됨
- 빠른 결말 예측을 하기 위해선 다양하고 수많은 학습데이터를 산출하여, 계산을 수행치 아니하고도 예측할 수 있는 기술이 필요함

## 2. 분석 목표 확인

- 9가지 핵종 그룹(Xe, Cs, ..., La)의 방출분율(0-1)에 따른 결말 예측 (기타 features 추가 예정)

## 3. 분석 일정 수립

## 4. 데이터 특성 확인

- 신고리 1호기 방출 분율(0-1)에 대한 Uniform 분포 가정(0-1)
- 10,000개의 LHS(Latin Hypercube Sampling) 수행 및 결말 분석 완료

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	Xe	Cs	Ba	I	Te	Ru	Mo	Ce	La	EF-2	EF-5	EF-10	LF-10	LF-16	LF-26
Model1.out	0.935813	0.271075	0.180771	0.296865	0.446958	0.734332	0.408556	0.776604	0.885355	51.3	356	942	2510	10200	115000
Model2.out	0.944064	0.475762	0.312053	0.182343	0.517058	0.012851	0.789794	0.142251	0.731276	46.1	293	564	2690	10200	94400
Model3.out	0.058822	0.996954	0.735196	0.620115	0.963037	0.951096	0.135571	0.635692	0.705072	50.5	347	851	2550	10200	108000
Model4.out	0.496441	0.770491	0.208338	0.98046	0.201525	0.952312	0.584485	0.652815	0.991367	51.6	358	957	2530	10300	118000
Model5.out	0.265422	0.352622	0.191012	0.061863	0.637979	0.183665	0.069416	0.322771	0.763212	47.3	310	659	2660	10300	100000
Model6.out	0.700237	0.81749	0.941062	0.021987	0.501194	0.512394	0.228431	0.531725	0.774355	49.6	338	795	2590	10300	107000
:															
Model9997.out	0.356604	0.981764	0.516944	0.979931	0.202416	0.621013	0.100477	0.481213	0.268502	45.7	286	508	2540	9140	76300
Model9998.out	0.033572	0.351718	0.526632	0.668728	0.748873	0.276102	0.193047	0.882266	0.616257	50.5	348	861	2490	9990	104000
Model9999.out	0.333517	0.013866	0.871355	0.579371	0.725287	0.466195	0.39162	0.416885	0.737274	49	331	742	2610	10300	104000
Model10000.out	0.141152	0.611811	0.931245	0.740499	0.192262	0.936708	0.018031	0.165533	0.85219	48.2	321	692	2700	10600	107000

# 2. 데이터 확보/탐색(Data)

## 1. 데이터 확보

➢ 엑셀 데이터 > PANDAS 자료형 데이터로 변환 (오른쪽 그림)

## 2. 결측값 처리

## 3. 이상치 파악

## 4. 데이터 분포

➢ 학습변수 (Xe, ..., La) 사이의 분포를 분석하여 모델학습에 적절한 변수임을 확인해야함

## 5. 상관 관계

	Xe	Cs	Ba	I	Te	Ru	Mo	...	La	EF-2
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	...	1.000000e+04	10000.000000
mean	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000	...	4.999999e-01	47.268210
std	0.288690	0.288689	0.288689	0.288690	0.288690	0.288690	0.288689	...	2.886895e-01	3.051966
min	0.000031	0.000055	0.000054	0.000015	0.000084	0.000095	0.000050	...	6.807466e-07	35.300000
25%	0.250016	0.250007	0.250033	0.250028	0.250061	0.250033	0.250061	...	2.500392e-01	45.300000
50%	0.500025	0.500014	0.500024	0.500010	0.499999	0.499999	0.499999	...	4.999902e-01	47.700000
75%	0.750007	0.749964	0.749984	0.749974	0.749979	0.749963	0.749974	...	7.499587e-01	49.600000
max	0.999954	0.999953	0.999959	0.999929	0.999968	0.999994	0.999911	...	9.999186e-01	53.300000

[8 rows x 15 columns]  
PS D:\PSH>

<데이터의 통계 값(mean, std, min ...) 확인>

	Xe	Cs	Ba	I	Te	Ru	Mo	Ce	La	EF-2	EF-5	EF-10	LF-10	LF-16	LF-26
Xe	1.000000	0.003246	0.003508	-0.002973	0.000089	0.010342	0.005153	-0.000359	-0.000462	0.006571	0.005187	0.003501	0.002207	0.003668	0.002182
Cs	0.003246	1.000000	0.001810	-0.000904	0.000619	-0.004093	-0.002667	0.002199	-0.005100	0.004859	0.005017	-0.000087	0.003036	0.001612	0.000732
Ba	0.003508	0.001810	1.000000	-0.004762	0.000237	-0.001624	-0.000702	0.000596	0.003825	0.066167	0.067779	-0.026879	0.019937	0.033175	0.028149
I	-0.002973	-0.000904	-0.004762	1.000000	-0.002392	-0.000223	0.003644	-0.001336	-0.001923	0.043256	0.039445	0.020957	-0.006244	0.006814	0.007067
Te	0.000089	0.000619	0.000237	-0.002392	1.000000	-0.001468	0.000349	-0.001799	0.004370	0.079151	0.083469	0.054697	-0.035505	0.005138	0.017447
Ru	0.010342	-0.004093	-0.001624	-0.000223	-0.001468	1.000000	0.004896	-0.001237	-0.006366	0.107912	0.106010	0.096773	0.097810	0.089948	0.104242
Mo	0.005153	-0.002667	-0.000702	0.003644	0.000349	0.004896	1.000000	-0.001217	0.004229	0.065412	0.064395	0.048926	-0.043961	-0.005885	0.012312
Ce	-0.000359	0.002199	0.000596	-0.001336	-0.001799	-0.001237	-0.001217	1.000000	-0.000640	0.679773	0.672603	0.670550	-0.196662	0.190415	0.356815
La	-0.000462	-0.005100	0.003825	-0.001923	0.004370	-0.006366	0.004229	-0.000640	1.000000	0.693012	0.683651	0.725833	0.647322	0.829791	0.898630
EF-2	0.006571	0.004859	0.066167	0.043256	0.079151	0.107912	0.065412	0.679773	0.693012	1.000000	0.997562	0.990377	0.417009	0.793359	0.916198
EF-5	0.005187	0.005017	0.067779	0.039445	0.083469	0.106010	0.064395	0.672603	0.683651	0.997562	1.000000	0.983084	0.432637	0.803277	0.914548
EF-10	0.003501	-0.000087	0.026879	0.020957	0.054697	0.096773	0.048926	0.670550	0.725833	0.990377	0.983084	1.000000	0.362451	0.762570	0.916731
LF-10	0.002207	0.003036	0.019937	-0.006244	-0.035505	0.097810	-0.043961	-0.196662	0.647322	0.417009	0.432637	0.362451	1.000000	0.872460	0.673160
LF-16	0.003668	0.001612	0.033175	0.006814	0.005138	0.089948	-0.005885	0.190415	0.829791	0.793359	0.803277	0.762570	0.872460	1.000000	0.940513
LF-26	0.002182	0.000732	0.028149	0.007067	0.017447	0.104242	0.012312	0.356815	0.898630	0.916198	0.914548	0.916731	0.673160	0.940513	1.000000

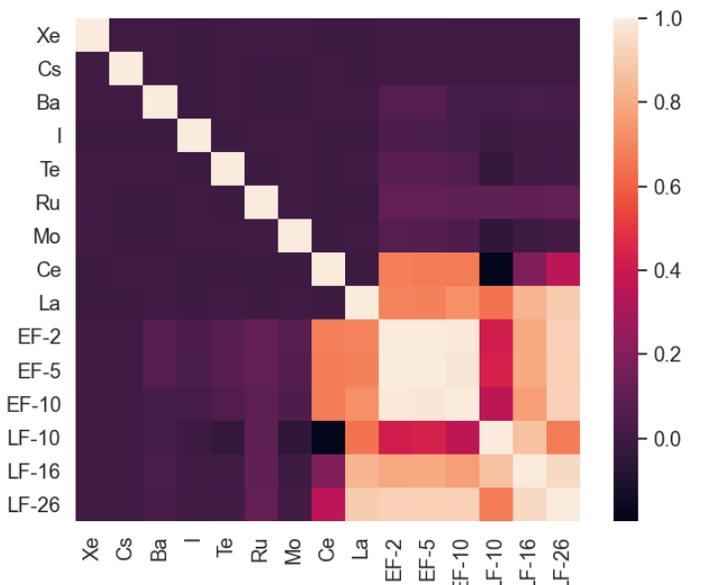
<변수 사이의 상관관계(correlation) 분석, 상관관계 Heat map>

```

1 import pandas as pd
2 import numpy as np
3 import random
4 import tensorflow as tr
5 from tensorflow.python.keras.engine.training import Model
6
7 print(tr.__version__)
8
9 #랜덤 시드 고정
10 SEED = 12
11 random.seed(SEED)
12 np.random.seed(SEED)
13 tr.random.set_seed(SEED)
14 print("시드 고정:", SEED)
15
16 #파일 불러오기
17 df = pd.read_excel('D:/PSH/For_Dissertation/Data(Total).xlsx')
18
19 from sklearn.model_selection import train_test_split
20 x_data = df.loc[:,['Xe','Cs','Ba','I','Te','Ru','Mo','Ce','La']]
21 y_data = df.loc[:, 'EF-2']
22
23 print(df.head(5))

```

<Excel to PANDAS 자료 변환>



# 3. 데이터 전처리(Preprocessing)

1. 내외부 데이터 결합
2. 결측값 처리
3. 이상치 처리
4. 자료형 변환
5. 범주형 변수 인코딩
6. 속성 추가

# 4. 모델 학습(Modeling)

## 1. 훈련/검증 데이터 분할

- 전체 데이터 중, 20~30%의 데이터를 검증에 쓰이는 것이 적절함
- 검증 데이터는 랜덤으로 설정함(shuffle=True)

```
from sklearn.model_selection import train_test_split
x_data = df.loc[:,['Xe', 'Cs', 'Ba', 'I', 'Te', 'Ru', 'Mo', 'Ce', 'La']]
y_data = df.loc[:, 'EF-2']
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.2, shuffle=True, random_state=12)
```

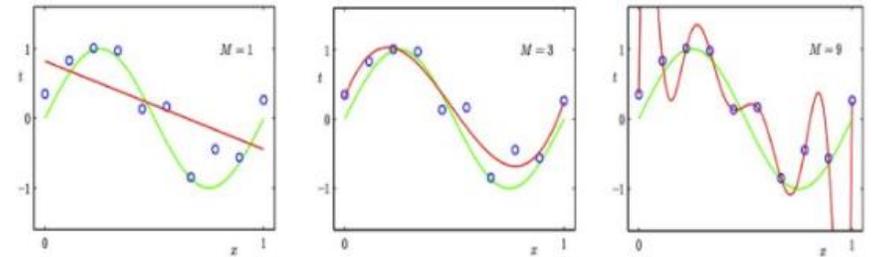
## 2. 알고리즘 선택 (기계학습)

- ① 베이스라인 모델: 선형 회귀(Linear Regression) 모델
- ② 다항식 모델: 2차 다항식 ~ 15차 다항식 모델
- ③ 비 선형 회귀 모델(의사결정나무, 랜덤포레스트, XGBoost)
- ④ 과대 적합(overfitting), 과소 적합(underfitting) 방지
  - ✓ 과대 적합: 모델이 학습에 사용한 데이터와 비슷한 데이터는 잘 예측하지만, 경험해보지 못한 새로운 특성을 갖는 데이터에 대해서는 예측력이 떨어짐 (너무 잘 맞아서 문제)
  - ✓ 과소 적합: 훈련 데이터의 특성을 파악하기 충분하지 않을 정도로 모델의 구성이 단순하거나 데이터 개수가 부족할 때 발생함 (너무 안 맞아서 문제)



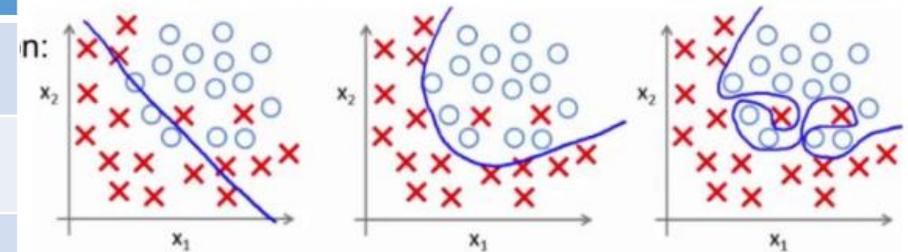
<학습 데이터 구분>

Regression:



predictor too inflexible: cannot capture pattern

predictor too flexible: fits noise in the data



Copyright © 2014 Victor Lavrenko

사진 출처: <https://www.youtube.com/watch?v=dBLZg-RqoLg>

구분	수식	설명
MAE(Mean Absolute Error)	$\frac{1}{n} \sum_{i=1}^n  Y_i - \hat{Y}_i $	실제값( $Y_i$ )과 예측값( $\hat{Y}_i$ )의 차이, 즉 잔차의 절대값을 평균한 값
MSE(Mean Squared Error)	$\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$	실제값( $Y_i$ )과 예측값( $\hat{Y}_i$ )의 차이, 즉 잔차의 제곱을 평균한 값
RMSE(Root Mean Squared Error)	$\sqrt{\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2}$	MSE의 제곱근

# 4. 모델 학습(Modeling)

## 1. 선형 회귀모델(Linear Regression Model) 및 L2/L1 규제

- 차수가 증가함에 훈련 MSE는 지속적으로 감소하나 계산시간은 크게 증가함 > 4차에서 6차방정식이 예측하는 차원에서 적절한 회귀 모델임
- TEST MSE는 6차 방정식이후로 크게 증가하여 전체적으로 과대 적합(overfitting)을 일으킴 > 부적절
- 과대적합/과소적합을 줄이기 위한 방법은 L2 규제(Ridge 규제)/L1규제(LASSO: Least Absolute Shrinkage and Selection Operator)가 있음

선형회귀	Train MSE	Test MSE	비고
1차 방정식	0.2604525	0.267284	overfitting
2차 방정식	0.0117238	0.010514	underfitting
3차 방정식	0.0045385	0.004155	underfitting
4차 방정식	0.0023242	0.002421	overfitting
5차 방정식	0.0012774	0.002569	overfitting
6차 방정식	0.0004664	0.006982	overfitting
7차 방정식	0	0.056855	overfitting
8차 방정식	0	0.053074	overfitting
9차 방정식	0	0.056577	overfitting
10차 방정식	0	0.065499	overfitting
11차 방정식	0	0.080129	overfitting
12차 방정식	0	0.10198	overfitting
13차 방정식	0	0.133509	overfitting
14차 방정식	0	0.178073	overfitting
15차 방정식	0	0.239788	overfitting

```
#6차 방정식
pf = PolynomialFeatures(degree=6)
x_train_poly = pf.fit_transform(x_train)

lr = LinearRegression()
lr.fit(x_train_poly, y_train)

#테스트 데이터에 대한 예측 및 평가
y_train_pred = lr.predict(x_train_poly)
train_mse = mean_squared_error(y_train, y_train_pred)
print("Train MSE:%.8f" % train_mse)

x_test_poly = pf.fit_transform(x_test)
y_test_pred = lr.predict(x_test_poly)
test_mse = mean_squared_error(y_test, y_test_pred)
print("Test MSE:%.8f" % test_mse)
```

# 4. 모델 학습(Modeling)

## 2. 비선형 회귀모델(Linear Regression Model)

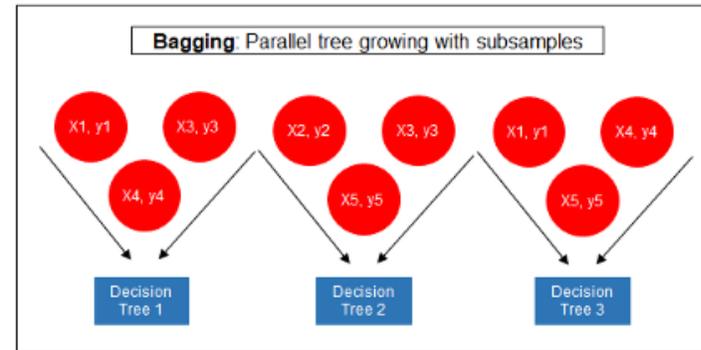
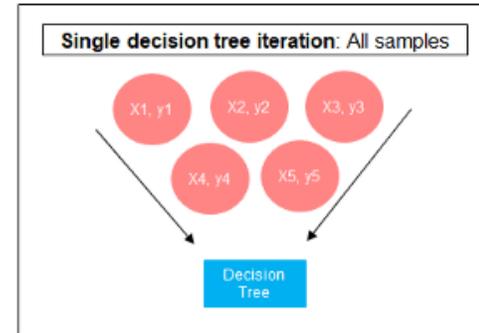
- 의사결정나무(Single Decision Tree)
  - 트리(tree) 알고리즘을 사용함
  - 트리의 각 분기점(node)에는 데이터셋의 피처(설명변수)를 하나씩 위치함
- 랜덤 포레스트(Multiple Decision Tree)
- XGBoost(Sequential Tree)

비선형회귀	Train MSE	Test MSE	비고
의사결정나무	0.13135125	0.29466080	overfitting
랜덤포레스트	0.06334497	0.12851057	overfitting
XGBoost	0.00051215	0.05887720	overfitting

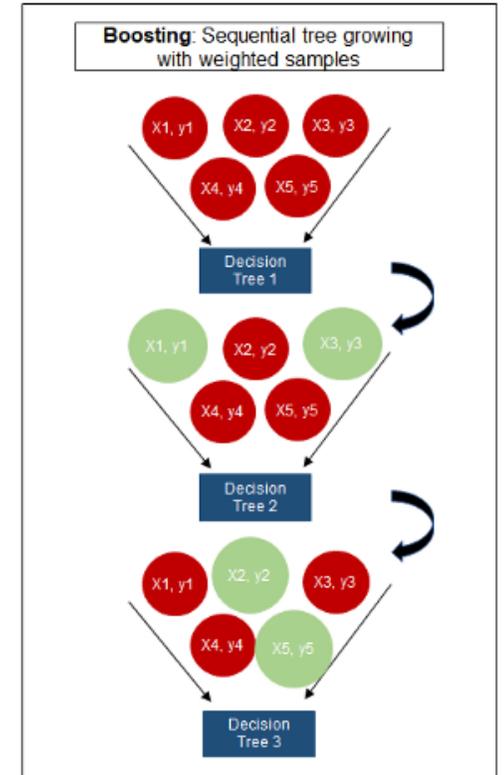
```
#의사결정나무
from sklearn.tree import DecisionTreeRegressor
dtr = DecisionTreeRegressor(max_depth=100, random_state=100)
dtr.fit(x_train, y_train)

y_train_pred = dtr.predict(x_train)
train_mse = mean_squared_error(y_train, y_train_pred)
print("Train MSE:%.8f" % train_mse)

y_test_pred = dtr.predict(x_test)
test_mse = mean_squared_error(y_test, y_test_pred)
print("Test MSE:%.8f" % test_mse)
```



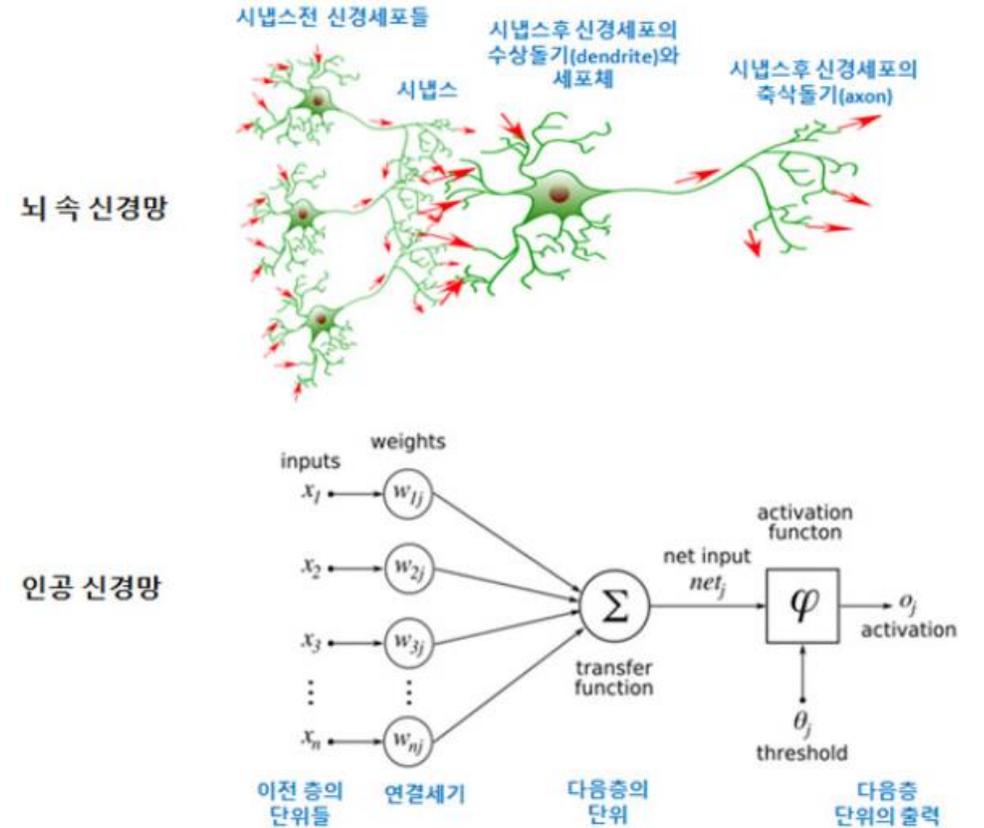
출처: Julia Nikulski.



# 4. 모델 학습(Modeling)

## 3. 인공 신경망

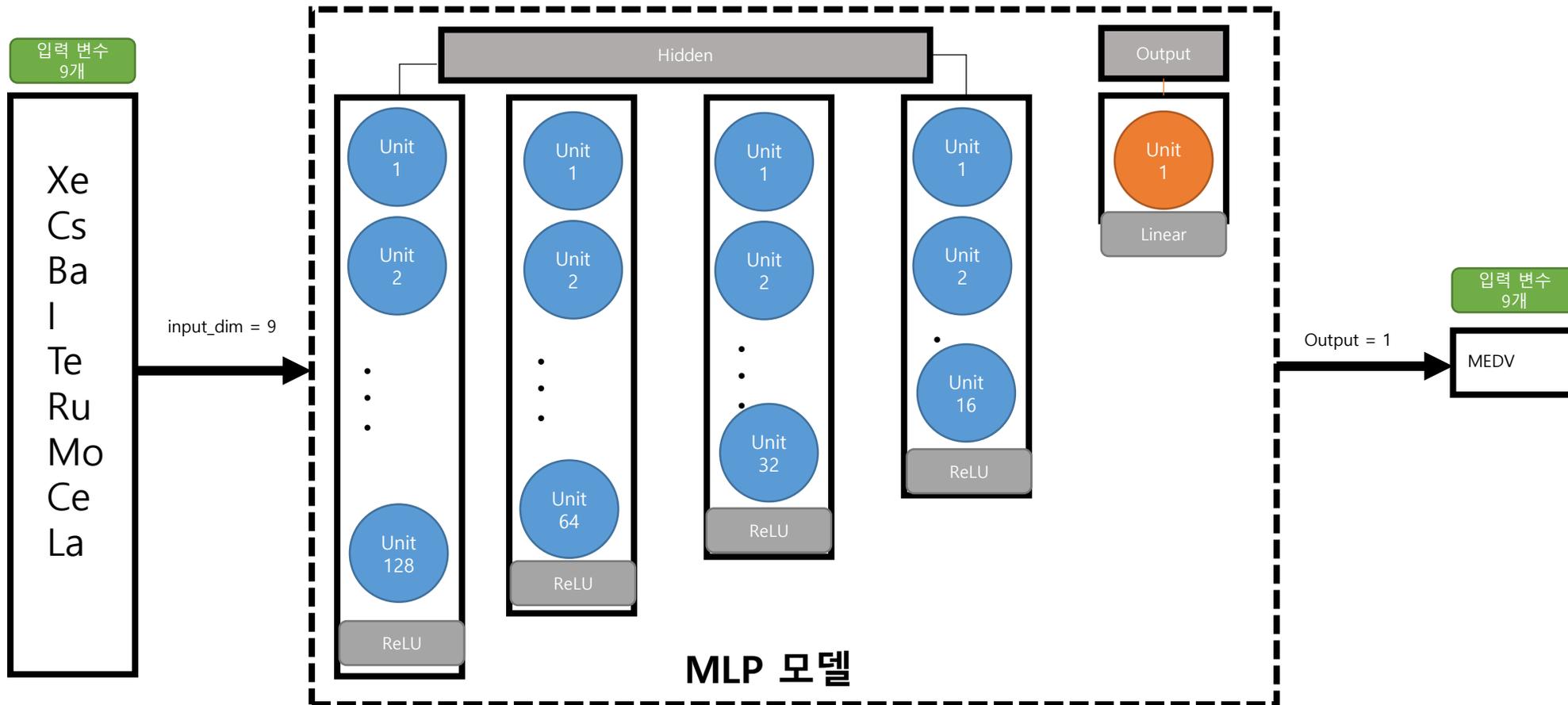
- 인공 신경망은 뇌 신경계 정보 처리 구조를 모방하여 만든 컴퓨터 계산 알고리즘
- 뇌 신경은 수많은 신경세포(뉴런)들이 연결되어 정보를 처리하고 전달함
- 신경세포는 신호를 입력 받아 다음 신경세포에 출력 형태로 연결한다는 점에서 입력과 출력을 갖는 함수와 비슷함
- (오른쪽 그림에서)  $w_i$ 는 함수식의 가중치(함수의 계수)를 말하고, 딥러닝은 입력( $x_i$ )과 출력( $y$ )의 관계를 잘 설명할 수 있는 복잡한 함수식의 가중치를 찾는 과정을 말함
- **활성화 함수(activation function)**
  - 인공 신경망이 복잡한 문제를 잘 설명하는데 활성화 함수를 이용한 비선형 변환이 중요한 역할을 함
  - 실제로 어떤 활성화 함수를 적용하냐에 따라 딥러닝의 예측력이 크게 달라짐
- **손실함수(loss function)**
  - 인공 신경망이 출력하는 값과 실제 정답과의 차이를 말함.
  - 손실함수가 적을수록 좋음
- **옵티마이저(최적화)**
  - 손실함수를 최소화하는 방향으로 가중치를 갱신하는 알고리즘
  - ex) 경사하강법 등



# 4. 모델 학습(Modeling)

## 3. 다중 신경망 아키텍처

- 인공 신경망은 뇌 신경계 정보 처리 구조를 모방하여 만든 컴퓨터 계산 알고리즘
- 모델의 레이어 개수와 각 레이어의 유닛 수를 변경하여 최적의 모델 아키텍처를 찾는 것이 필요함(향후 연구)



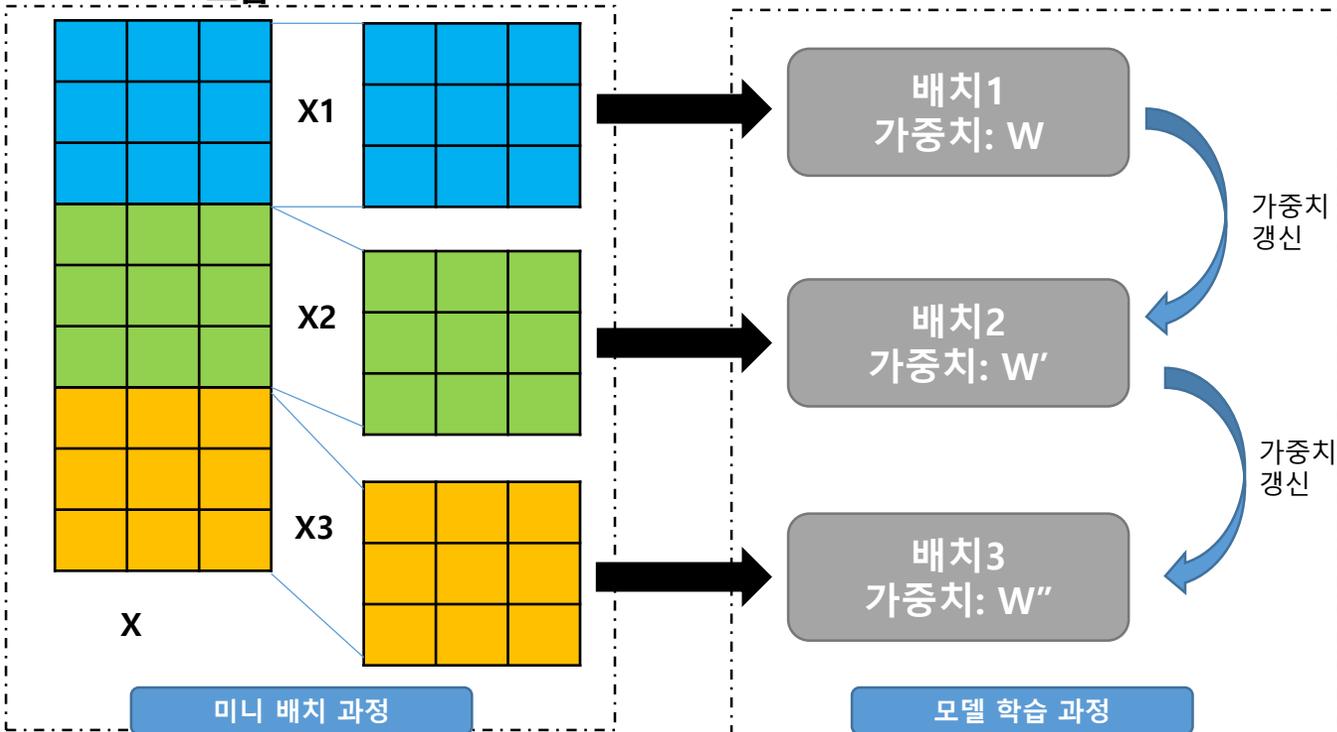
# 4. 모델 학습(Modeling)

## - 다중 신경망 구현

- TensorFlow의 Keras 모듈 사용
- 파라미터수
- $12161 = 128 * (9+1) + 64 * (128 + 1) + 32 * (64 + 1) + 16 * (32 + 1) + 1 * (16+1)$

## - 미니 배치

- 모델을 훈련시킬 때 샘플 데이터를 한 개씩 입력해서 가중치를 갱신하려면 학습 시간이 오래 걸리는 문제가 있음
- 미니 배치 학습은 여러 개의 작은 배치 단위로 나누고 배치에 들어 있는 샘플 데이터를 묶어서 모델에 입력하는 것
- 배치 단위로 경사하강법을 적용하고 손실함수를 최소화하는 방향으로 가중치를 업데이트함



```
31
32 #심층 신경망
33 from tensorflow.keras import Sequential
34 from tensorflow.keras.layers import Dense
35 from sklearn import model_selection
36
37 def build_model(num_input=1):
38     model = Sequential()
39     model.add(Dense(128, activation='relu', input_dim=num_input))
40     model.add(Dense(64, activation='relu'))
41     model.add(Dense(32, activation='relu'))
42     model.add(Dense(16, activation='relu'))
43     model.add(Dense(1, activation='linear'))
44     model.compile(optimizer='adam', loss='mse', metrics=['mae'])
45     return model
46
47 model = build_model(num_input=9)
48 print(type(model))
49 print(model.summary())
50
51
```

문제 출력 **터미널** 디버그 콘솔

```
<class 'tensorflow.python.keras.engine.sequential.Sequential'>
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	1280
dense_1 (Dense)	(None, 64)	8256
dense_2 (Dense)	(None, 32)	2080
dense_3 (Dense)	(None, 16)	528
dense_4 (Dense)	(None, 1)	17

Total params: 12,161  
Trainable params: 12,161  
Non-trainable params: 0

# 4. 모델 학습(Modeling)

## - 미니 배치 학습

- 전체 입력 데이터를 32개씩 구분하여 미니 배치 구성함
- $10000/32 = 312.5$  (313개 배치 생성)
- 에포크(epoch) : 100 (학습 횟수)
- >> 가중치 업데이트 횟수 =  $313 * 100 = 31,300$ (회)

- 실행 결과를 보면, 에포크가 진행됨에 따라 손실 함수의 출력값이 감소하고 있으므로 학습이 잘 진행되는 것을 확인함
- 마지막 100번째 에포크가 끝났을 때 훈련데이터에 대한 손실함수(loss)는 0.0084 이고 metrics 옵션에 추가한 보조 지표인 MAE는 0.0734임
- 학습된 모델에 대한 평가(evaluate) 결과 손실함수(loss)는 0.0033 이고 MAE는 0.0457

딥러닝	Train MSE	Test MSE	비고
MLP 모델	0.0084	0.0033	underfitting

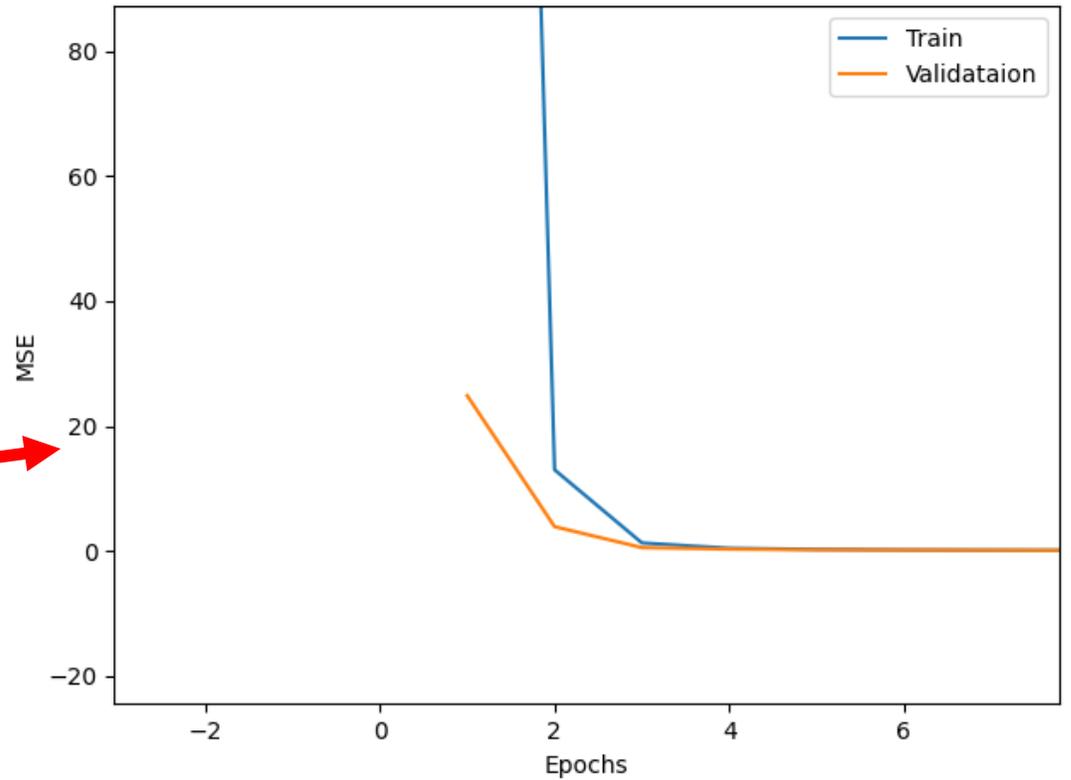
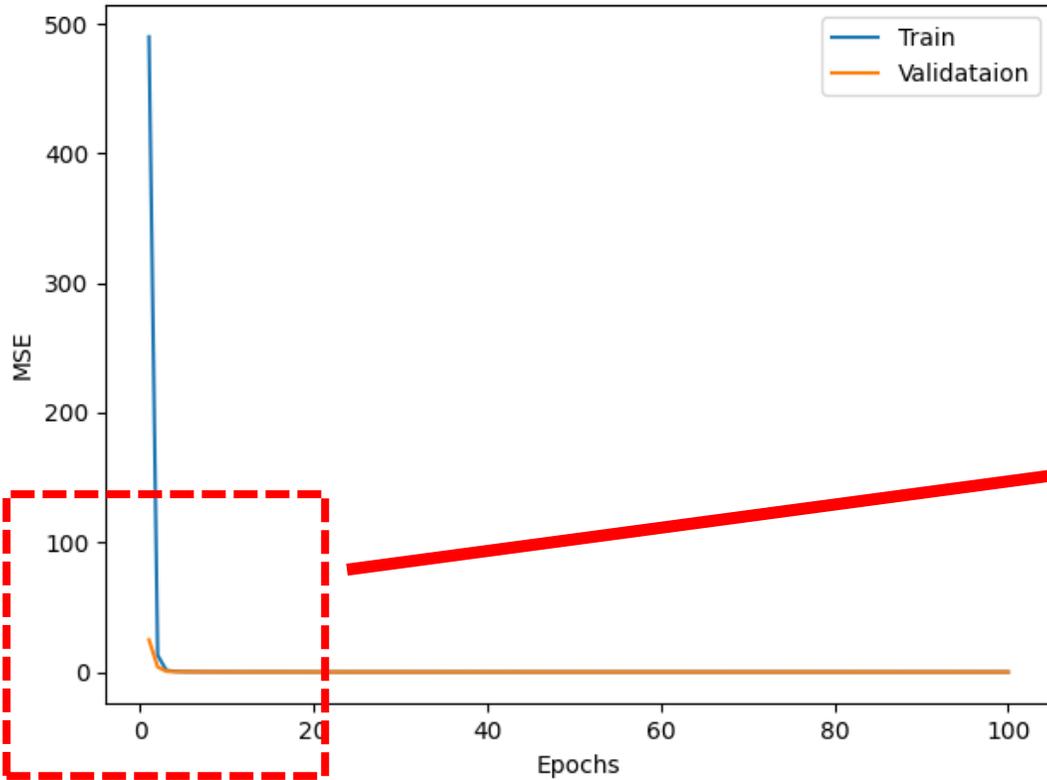
```
model = build_model(num_input=9)
print(type(model))
print(model.summary())
print(model.fit(x_train, y_train, epochs=100, batch_size=32, verbose=2))

#평가
print(model.evaluate(x_test, y_test))
```

문제	출력	터미널	디버그 콘솔
Epoch 93/100	250/250 - 1s - loss: 0.0119 - mae: 0.0873		
Epoch 94/100	250/250 - 1s - loss: 0.0129 - mae: 0.0908		
Epoch 95/100	250/250 - 1s - loss: 0.0070 - mae: 0.0654		
Epoch 96/100	250/250 - 1s - loss: 0.0092 - mae: 0.0751		
Epoch 97/100	250/250 - 1s - loss: 0.0093 - mae: 0.0753		
Epoch 98/100	250/250 - 1s - loss: 0.0085 - mae: 0.0727		
Epoch 99/100	250/250 - 1s - loss: 0.0101 - mae: 0.0786		
Epoch 100/100	250/250 - 1s - loss: 0.0084 - mae: 0.0734		
<tensorflow.python.keras.callbacks.History object at 0x0000026BAC5C9280>			
63/63 [=====] - 0s 3ms/step - loss: 0.0033 - mae: 0.0458			
[0.0033425448928028345, 0.045758627355098724]			
PS D:\PSH>			

# 4. 모델 학습(Modeling)

- 훈련 손실과 검증 손실을 그래프 검토
  - 모델은 3에포크(3번째 학습)까지 매우 빠른 속도로 학습이 되고, 이후 점차 완만하게 학습 속도가 낮아지면서 그래프가 평평해지는 추이를 보임



# 5. 결과 요약

- ✓ 9가지 핵종 그룹의 방출분율(피쳐)과 조기사망자수(레이블)에 대하여 기계학습을 수행함
- ✓ 선형회귀, 비선형회귀, 다중인공신경망을 적용해보았고, 결과는 다음과 같음

선형회귀	Train MSE	Test MSE	비고
1차 방정식	0.2604525	0.267284	overfitting
2차 방정식	0.0117238	0.010514	underfitting
3차 방정식	0.0045385	0.004155	underfitting
4차 방정식	0.0023242	0.002421	overfitting
5차 방정식	0.0012774	0.002569	overfitting
6차 방정식	0.0004664	0.006982	overfitting
7차 방정식	0	0.056855	overfitting
8차 방정식	0	0.053074	overfitting
9차 방정식	0	0.056577	overfitting
10차 방정식	0	0.065499	overfitting
11차 방정식	0	0.080129	overfitting
12차 방정식	0	0.10198	overfitting
13차 방정식	0	0.133509	overfitting
14차 방정식	0	0.178073	overfitting
15차 방정식	0	0.239788	overfitting

비선형회귀	Train MSE	Test MSE	비고
의사결정나무	<b>0.13135125</b>	<b>0.29466080</b>	overfitting
랜덤포레스트	<b>0.06334497</b>	<b>0.12851057</b>	overfitting
XGBoost	<b>0.00051215</b>	<b>0.05887720</b>	overfitting

딥러닝	Train MSE	Test MSE	비고
MLP 모델	<b>0.0084</b>	<b>0.0033</b>	underfitting

## 5. 결론 및 향후 연구

- ✓ Level 3 PSA의 입력과 결말에 대하여 기계학습의 적용 가능 여부를 판단하기 위해 본 연구를 수행함
- ✓ 수많은 알고리즘에 모두 적용이 가능하며, 학습데이터가 더 방대해지고, 변수가 증가하여도 적용이 충분히 가능함(방출 높이, 비상 대응, 확산 모델 등등)
- ✓ 향후, 선원항에 대한 머신러닝 기술과 접목하여 소외결말 관련 연구에 대해서 딥러닝 기술을 포함한 다양한 머신러닝 기술이 적용 가능함을 본 연구를 통해 입증함

### ✓ 향후 연구 계획

- ✓ 다양한 입력변수에 대하여 학습데이터 생산(최소 10만개 + 외부 데이터)
- ✓ 파이썬을 활용한 데이터 베이스 구축(PANDAS)
- ✓ 다양한 결말에 대한 기계학습 수행
- ✓ 알고리즘 비교 분석을 통한 최적 알고리즘 선정
- ✓ MACCS 입력 값을 통한 결말 산출 프로그램 개발(독자적으로 할 예정)