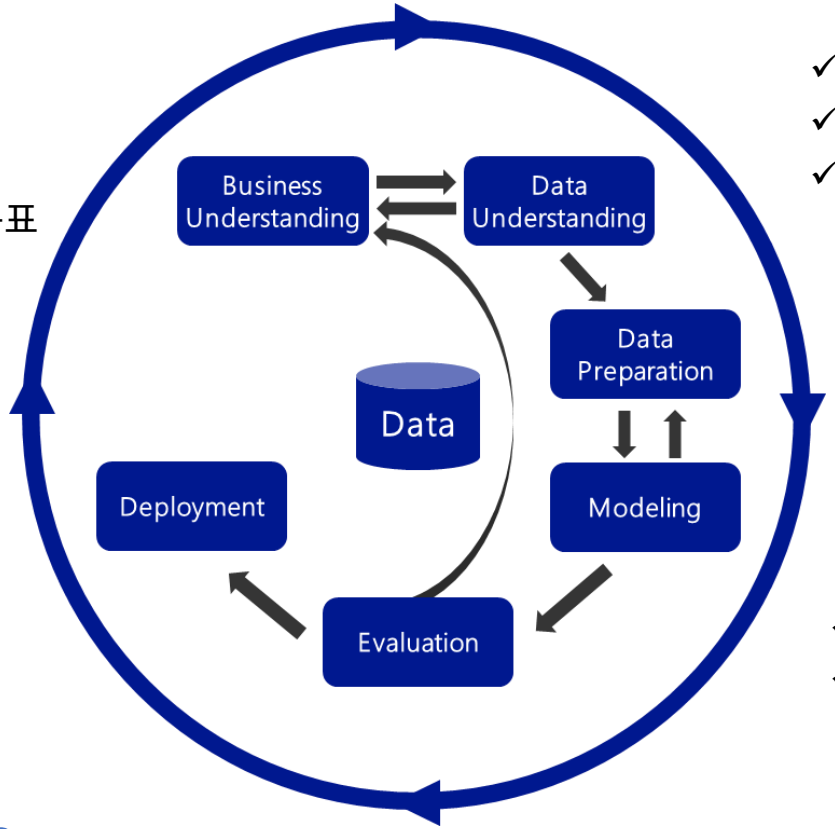
1일차 정리

전체 Process(CRISP-DM)

무엇이 문제인가?

✓ 비즈니스 **문제정의**
✓ 데이터분석 방향, 목표
✓ 초기 **가설 수립**

$$x \rightarrow y$$



✓ 원본식별
✓ **분석을 위한 구조 만들기**
✓ **데이터분석 EDA & CDA**

✓ 모델링을 위한
데이터 구조 만들기
- 모든 셀은 **값**이 있어야 한다.
- 모든 값은 **숫자** 여야 한다.
- (필요 시) 숫자의 **범위**가 일치

✓ 모델을 만들고
✓ 검증하기

✓ 모델 관리
✓ AI 서비스 구축

문제가 해결 되었는가?

✓ 기술적 관점 평가
✓ 비즈니스 관점 평가

알고리즘 한판 정리

	선형회귀	로지스틱회귀	KNN	SVM	Decision Tree	Random Forest	Gradient Boost (GBM, XGB, LGBM)
개념	✓오차를 최소화 하는 직선, 평면	✓오차를 최소화 하는 직선, 평면 ✓직선을 로지스틱 함수로 변환 (0~1 사이 값으로)	✓예측할 데이터와 train set과의 거리 계산 ✓가까운 [k개 이웃의 y]의 평균으로 예측	✓마진을 최대화 하는 초평면 찾기 ✓데이터 커널 변환	✓정보전달량 = 부모 불순도 - 자식 불순도 ✓정보 전달량이 가장 큰 변수를 기준으로 split	✓여러 개의 트리 ✓각각 예측 값의 평균 ✓행과 열에 대한 랜덤 : 조금씩 다른 트리들 생성	✓여러 개의 트리 ✓트리를 더해서 하나의 모델로 생성 ✓더해지는 트리는 오차를 줄이는 모델
전제 조건	✓NaN조치 ✓가변수화 ✓x들 간 독립	✓NaN조치 ✓가변수화 ✓x들 간 독립	✓NaN조치 ✓가변수화 ✓스케일링	✓NaN조치 ✓가변수화 ✓스케일링	✓NaN조치 ✓가변수화	✓NaN조치 ✓가변수화	✓NaN조치 ✓가변수화
성능	✓변수 선택 중요 ✓x가 많을 수록 복잡	✓변수 선택 중요 ✓x가 많을 수록 복잡	✓주요 hyper-parameter - n_neighbors : k 작을수록 복잡 - metric : 거리계산법	✓주요 hyper-parameter - C : 클수록 복잡 - gamma : 클수록 복잡	✓주요 hp - max_depth : 클수록 복잡 - min_samples_leaf : 작을수록 복잡	✓주요 hp 기본값으로도 충분! - n_estimators - max_features ✓기본값으로 생성된 모델 ==> 과적합 회피	✓주요 hp - n_estimators - learning_rate ✓XGB, LGBM : 과적합 회피를 위한 규제

회귀모델 평가

오차의 크기

$\hat{y}$: 예측값

오차 | 제곱 오차 | 절대값 오차 | 오차율

y	$\hat{y}$	$y-\hat{y}$	$(y-\hat{y})^2$	$\lvert y-\hat{y}\rvert$	$\left\lvert\frac{y-\hat{y}}{y}\right\rvert$
6	4				
5	6				
12	9				
2	2				

평균

MSE
RMSE

MAE

MAPE

평균 오차

평균 오차율

딥러닝 개념 - 학습 절차

✓ model.fit(x_train, y_train) 하는 순간…

단계① : 가중치에 (초기)값을 할당한다.
- *초기값은 랜덤으로 지정*

$$medv = 1 \cdot lstat + 3$$

$$\boldsymbol{medv = w_1 \cdot lstat + w_0}$$

단계② : (예측) 결과를 뽑는다.

medv	lstat	$\hat{y}$
20	10	13
10	11	14
8	15	18

단계③ : 오차를 계산한다.

$$mse = \frac{\sum(y - \hat{y})^2}{n} = \frac{7^2 + 6^2 + 8^2}{3}$$

forward propagation

단계④ : 오차를 줄이는 방향으로 가중치를 조정
- *Optimizer : GD, Adam…*

$$\boldsymbol{w_1 : 1 \rightarrow 0.8}$$
$$\boldsymbol{w_0 : 3 \rightarrow 3.3}$$

back propagation

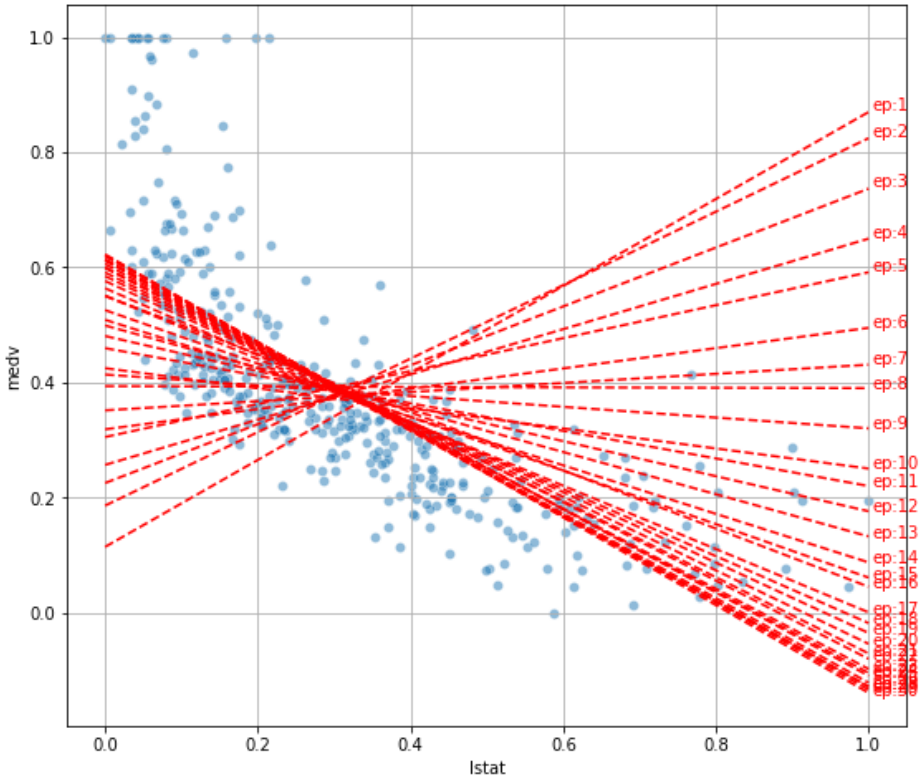
단계⑤ : 다시 단계①로 올라가 반복한다.
- *max iteration에 도달.(오차의 변동이 (거의) 없으면 끝.)*

- 가중치(weight)의 다른 용어 **파라미터(parameter)**

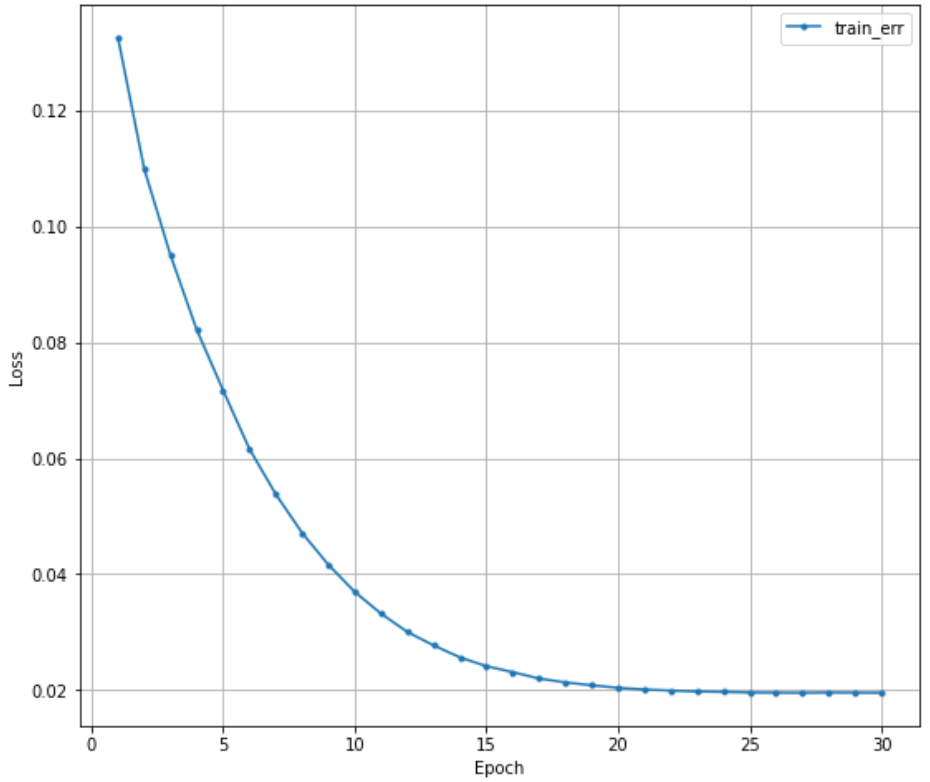
딥러닝 개념 - 학습 절차

✓30번 조정하며 최적의 Weight를 찾아가는 과정

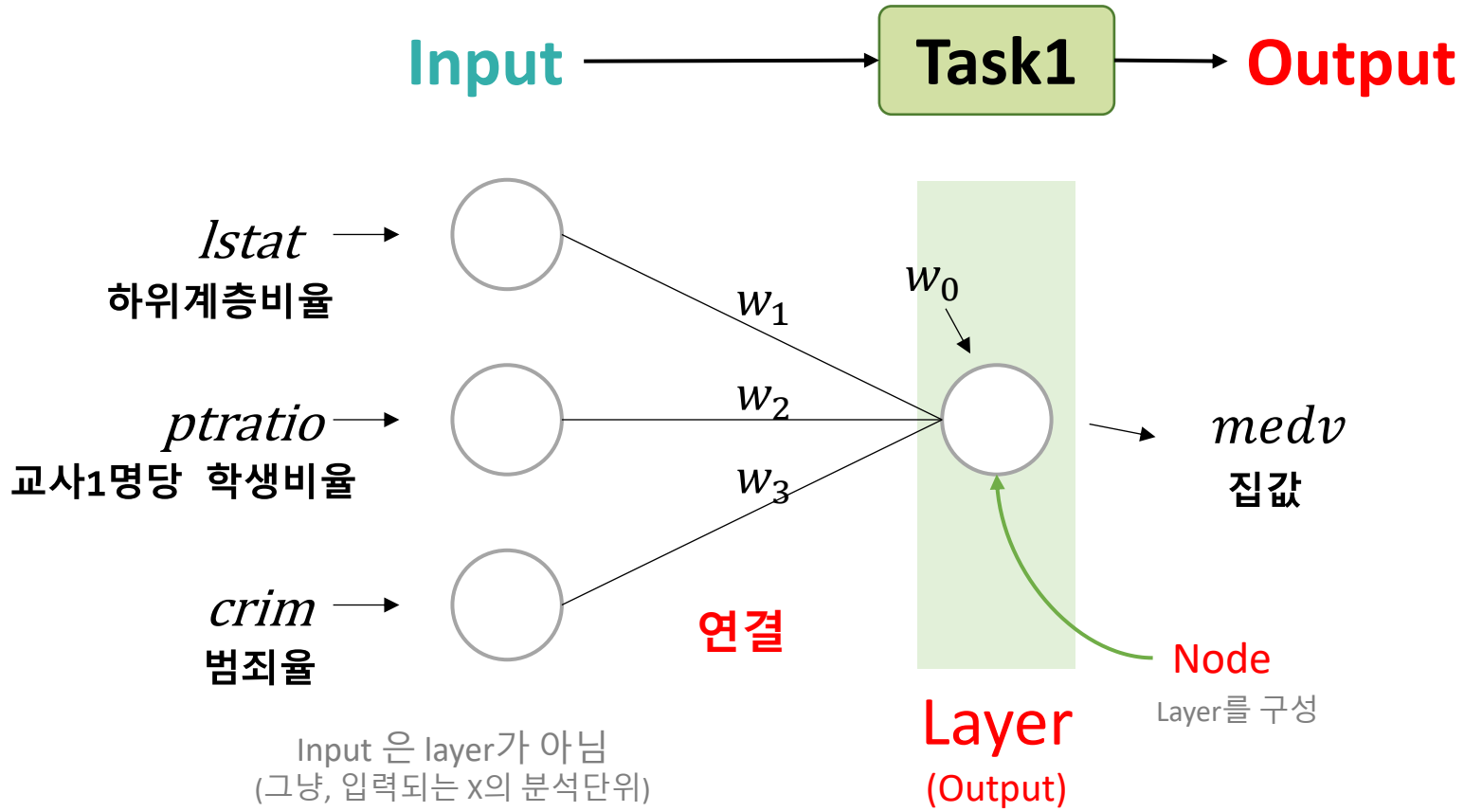
모델의 가중치가 업데이트되는 과정



모델의 오차가 줄어드는 과정

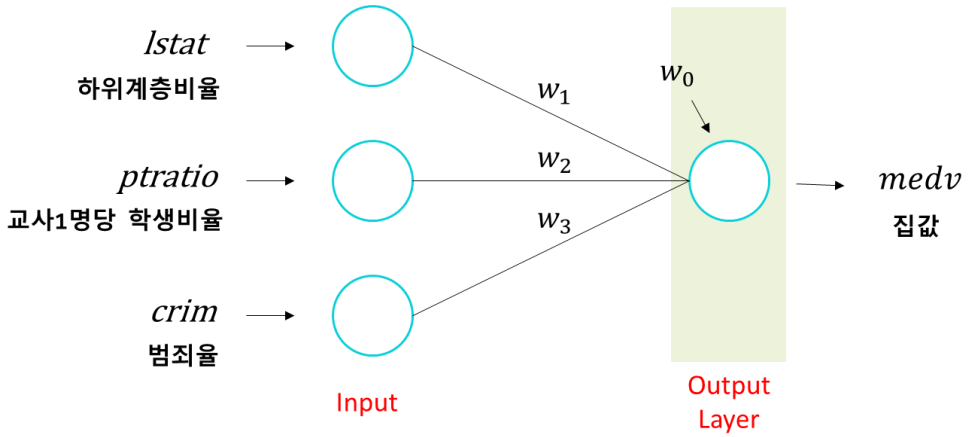


딥러닝 구조



$$medv = w_1 \cdot lstat + w_2 \cdot ptratio + w_3 \cdot crim + w_0$$

딥러닝 코드 - Dense



- ✓ **Input : `Input(shape = ( , ))`**
  - **분석단위**에 대한 shape
    - 1차원 : (feature 수,)
    - 2차원 : (rows, columns)
- ✓ **Output : `Dense( )`**
  - 예측 결과가 1개 변수(y가 1개 변수)

```
# 메모리 정리
clear_session()

# Sequential 타입
model = Sequential([Input(shape = (nfeatures,)),
                    Dense(1) ])
# 모델요약
model.summary()
```

3 *input* *output*

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 1)	4

Total params: 4 (16.00 B)
Trainable params: 4 (16.00 B)
Non-trainable params: 0 (0.00 B)

잠깐만요

위 모델 코드의 Input 함수를 첫 번째 Layer(Dense) 안에 옵션으로 포함 가능

```
Model = Sequential([Dense(1, input_shape=(3,)) ])
```

Compile

✓컴파일(Compile)

- 선언된 모델에 대해 몇 가지 설정을 한 후
- 컴퓨터가 이해할 수 있는 형태로 변환하는 작업

```
model.compile( optimizer = Adam(learning_rate = 0.1)
             , loss='mse')
```

✓loss function(오차함수)

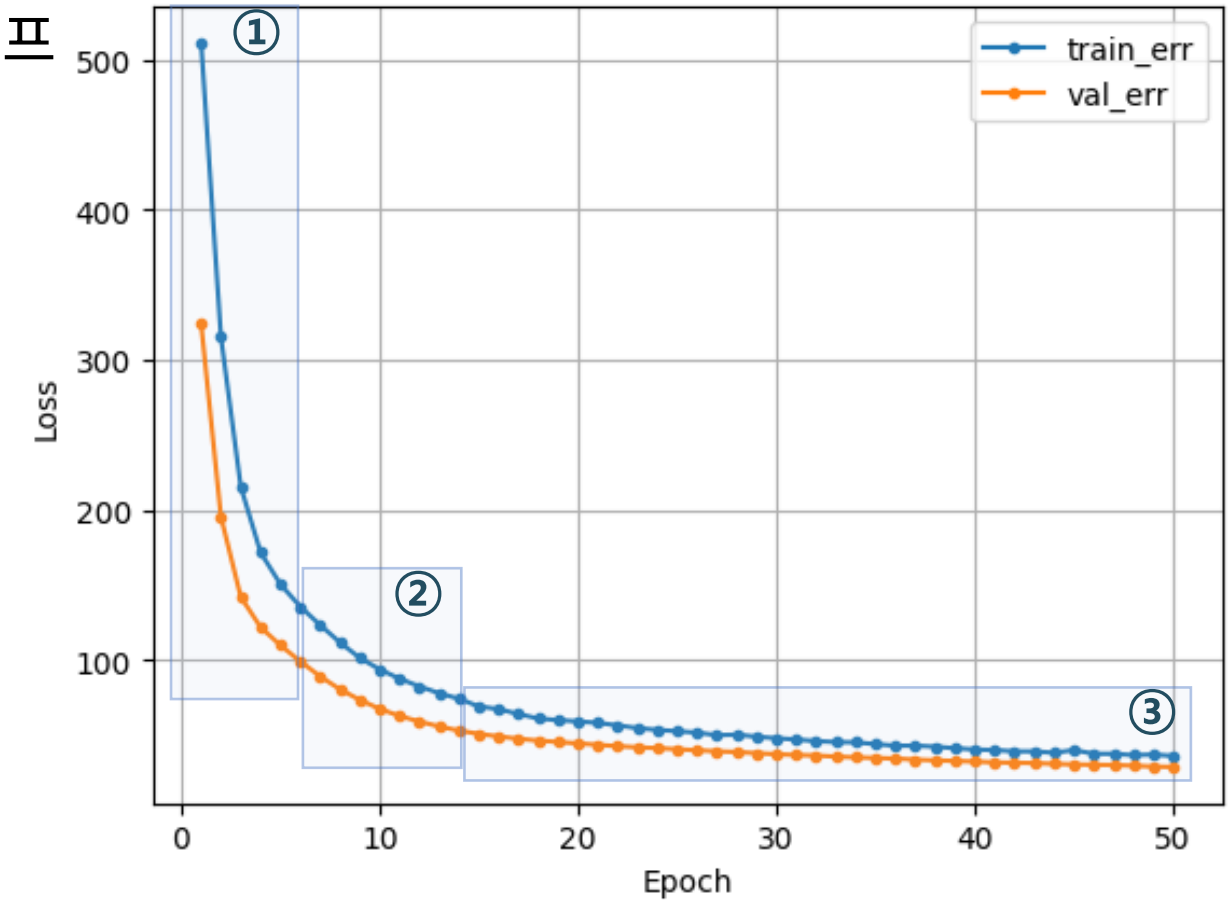
- Cost Function, Objective Function 과 같은(유사한) 의미
- 오차 계산을 무엇으로 할지 결정
- mse : mean squared error
  - 회귀모델 : mse
  - 분류모델 : cross entropy

학습 곡선

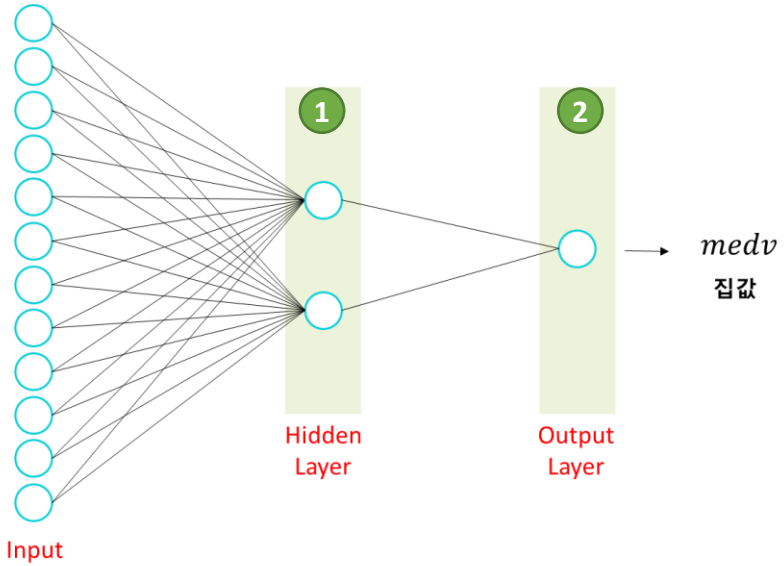
✓학습 곡선이란

- **모델 학습이 잘 되었는지 파악**하기 위한 그래프
  - 정답은 아니지만, **학습 경향을 파악**하는데 유용.

- 각 Epoch 마다 train error와 val error가 어떻게 줄어들고 있는지 확인
  - Train error, Val error 해석은 어떻게?



딥러닝 구조 - Hidden Layer



- ✓ **layer 여러 개 : 리스트[ ] 로 입력**
- ✓ **hidden layer** 1
  - **Activation** : 활성함수는 보통 'relu'를 사용
- ✓ **output layer** 2
  - 예측 결과가 1개

```
# Sequential 타입 모델 선언(입력은 리스트로!)
model3 = Sequential([Input(shape = (nfeatures,)),
                     1 Dense(2, activation = 'relu'),
                     2 Dense(1)   ])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 2)                 26

 dense_1 (Dense)             (None, 1)                 3

=================================================================
Total params: 29
Trainable params: 29
Non-trainable params: 0
_________________________________________________________________
```

활성화 함수 Activation Function

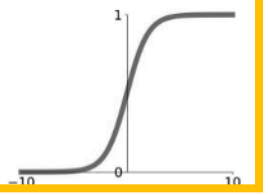
✓그래서 활성화 함수는…

- Hidden Layer에서는 : 선형함수를 비선형 함수로 변환
- Output Layer에서는 : 결과값을 다른 값으로 변환해 주는 역할
  - 주로 분류Classification 모델에서 필요

Hidden layer
국룰

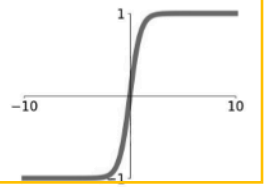
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



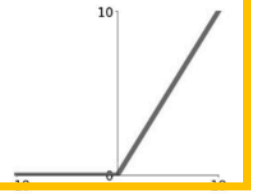
tanh

$$\tanh(x)$$



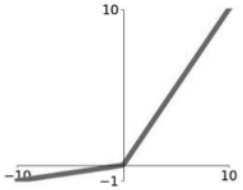
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

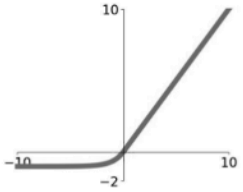


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



요약 : 회귀 모델링

✓ 딥러닝 전처리

- NaN 조치, 가변수화, 스케일링

✓ Layer

- 첫번째 Layer는 input_shape를 받는다.(분석단위의 shape)
  - 2차원 데이터셋의 분석단위 1차원 → shape는 (feature수,)
- Output layer의 node 수 : 1
- Activation Function
  - Hidden layer에 필요 :
    - 비선형 모델로 만들려고 ➔ hidden layer를 여럿 쌓아서 성능을 높이려고.
  - 회귀 모델링에서 Output Layer에는 활성화 함수 필요하지 않음!

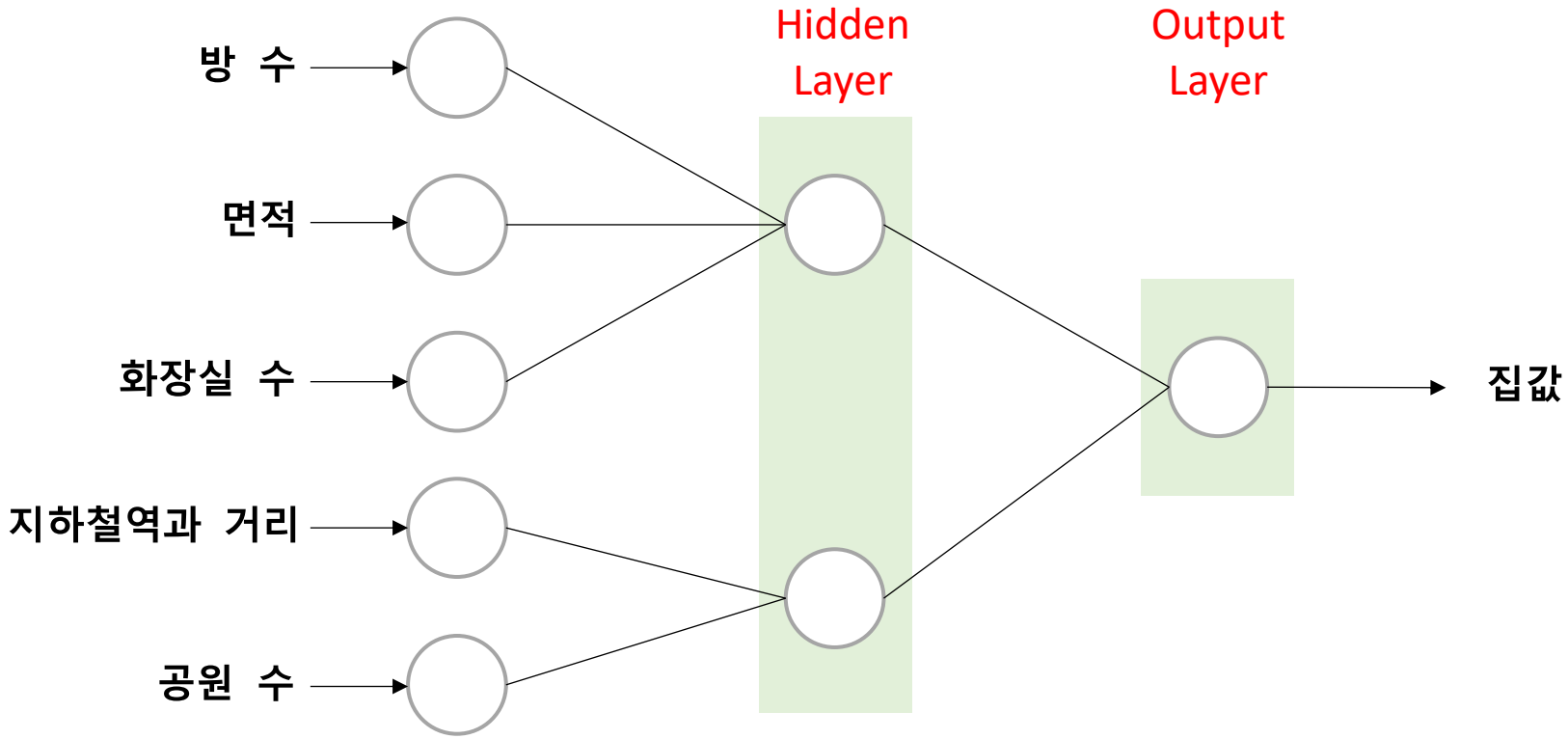
구분	Hidden Layer	Output Layer		Compile	
	Activation	Activation	Node수	optimizer	loss
Regression	relu	X	1	adam	mse

2일차 정리

Hidden Layer에서 무슨 일이 일어나는가?

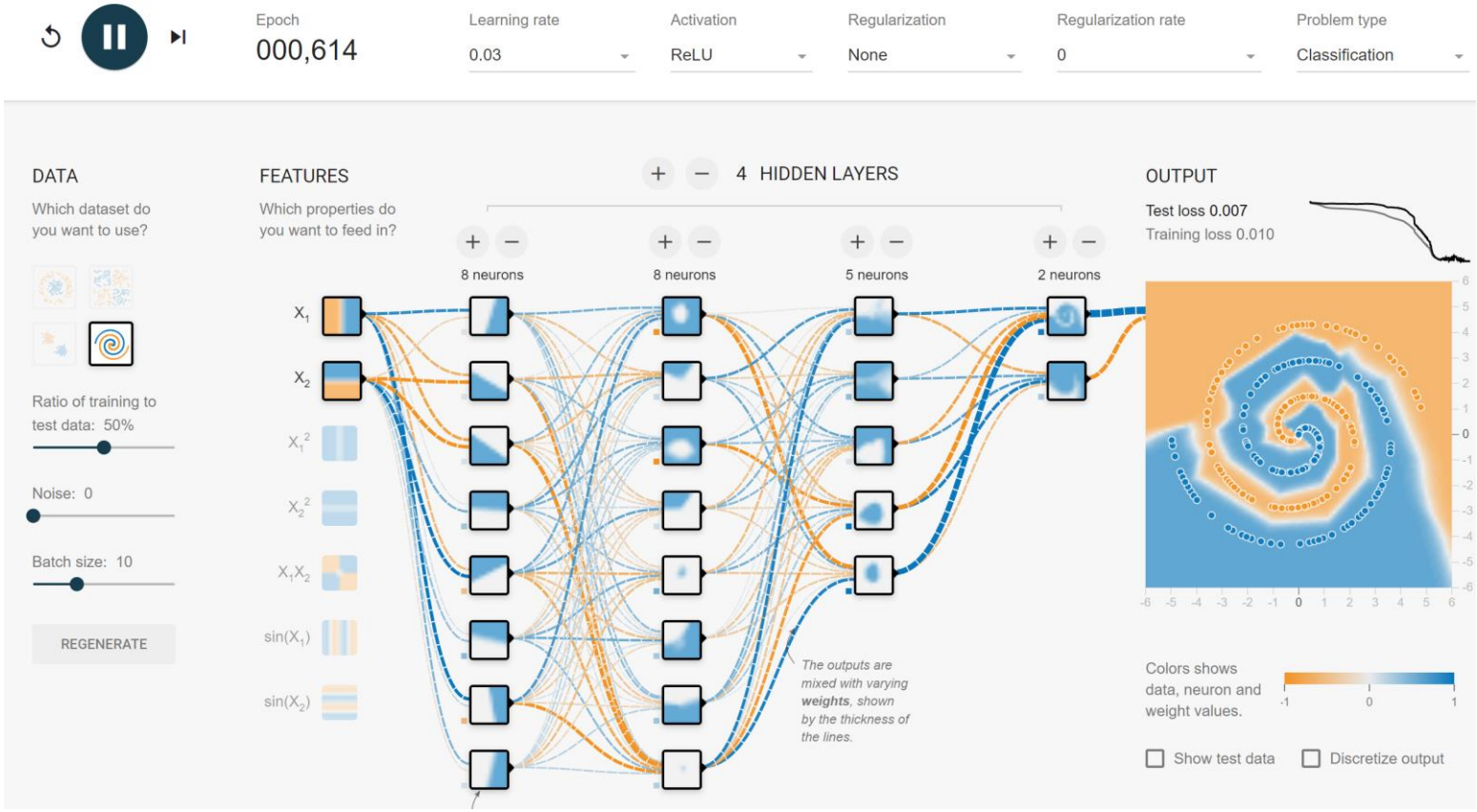
✓ **연결**

- 모든 노드 간에 연결을 할 수도 있지만(Fully Connected)
- 아래처럼 연결을 제어할 수도 있습니다.(Locally Connected)

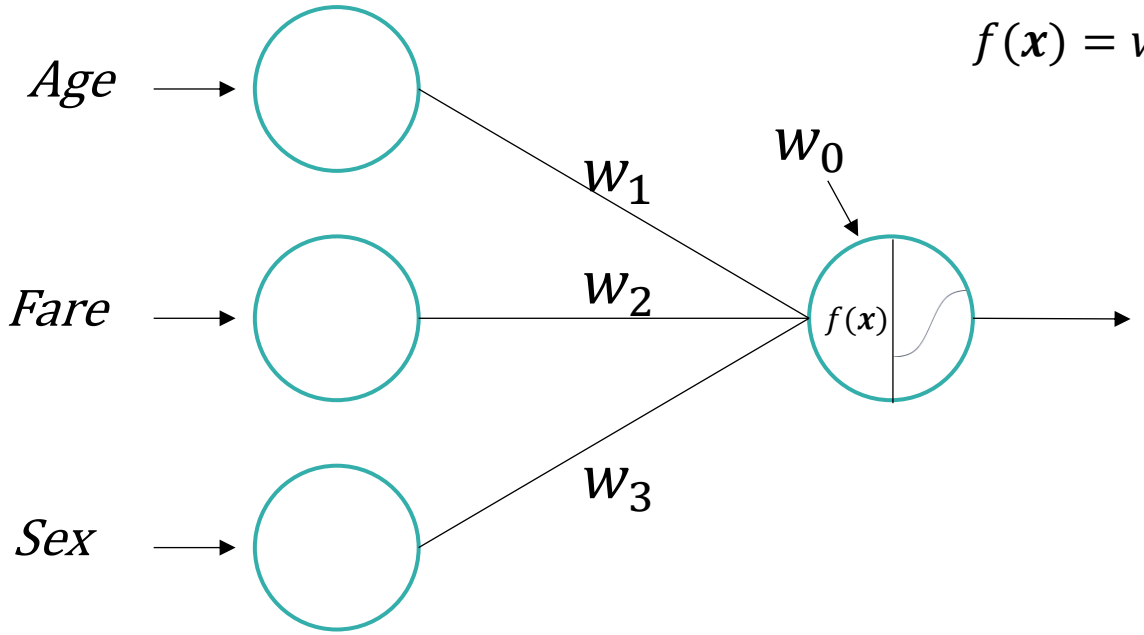


Hidden Layer에서 무슨 일이 일어나는가?

- ✓ **Tensorflow Playground** https://playground.tensorflow.org/
- ✓ https://bit.ly/487HdL1

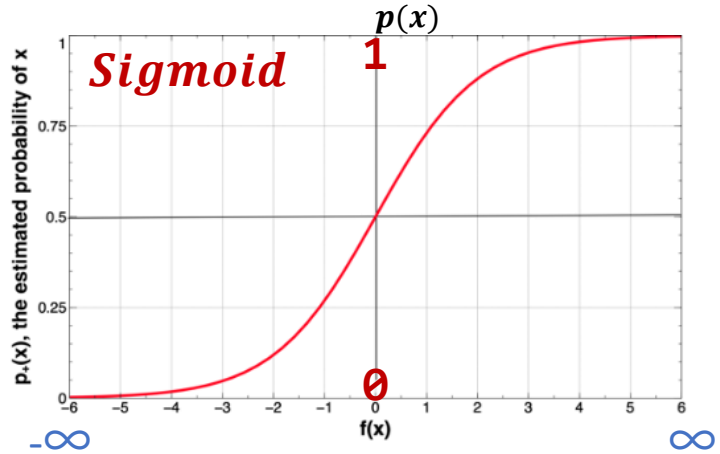


딥러닝 구조 - 이진분류



$$f(\boldsymbol{x}) = w_1 \cdot \boldsymbol{Age} + w_2 \cdot \boldsymbol{Fare} + w_3 \cdot \boldsymbol{Sex} + w_0$$

$$p_+(\mathbf{x}) = \frac{1}{1 + e^{-f(\mathbf{x})}}$$



✓ **Node의 결과를 변환해주는 함수가 필요**

- 그것을 **활성 함수**Activation Function 라고 합니다.

딥러닝 구조 – 활성 함수Activation Function

✓node의 결과를 변환시켜 주는 역할

Layer	Activation Function		기능
Hidden Layer	ReLU		좀 더 깊이 있는 학습(Deep Learning)을 시키려고. (Hidden Layer를 여러 층 쌓으려고) (선형 모델을 비선형 모델로 바꾸려고)
Output Layer	회귀	X	X
	이진분류	**sigmoid**	결과를 0, 1로 변환하기 위해
	다중분류	**softmax**	각 범주에 대한 결과를 범주별 확률 값으로 변환

딥러닝 구조 – Loss Function : binary_crossentropy

✓실제 값과 예측 값이 다음과 같다고 합시다.

- 우리는 이를 하나의 숫자(오차)로 평가해야 합니다.

y	$\hat{y}$
1	0.9
0	0.3
0	0.4
1	0.7
0	0.5
0	0.7
1	0.5

✓이진 분류 모델에서 사용되는 loss function

➔ binary_crossentropy

	$\hat{y} \approx 1$	$\hat{y} \approx 0$
$y = 1$	err 작게 (≈ 0)	err 크게 ($\approx \infty$)
$y = 0$	err 크게 ($\approx \infty$)	err 작게 (≈ 0)

딥러닝 구조 – Loss Function : binary_crossentropy

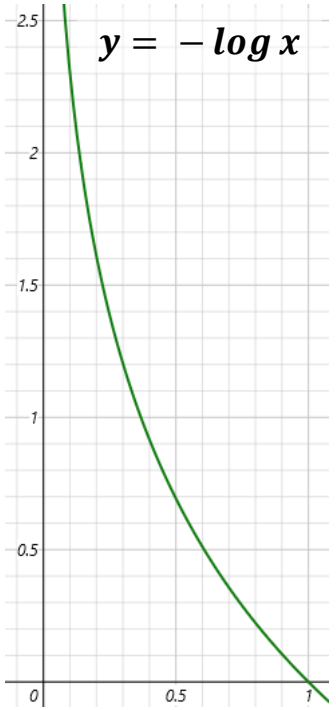
✓이 오차들의 평균

- Binary_crossentropy (log loss) $-\frac{1}{n}\sum(y \cdot log\,\hat{y} + (1-y) \cdot log(1-\hat{y}))$

y	$\hat{y}$
1	0.9
0	0.3
0	0.4
1	0.7
0	0.5
0	0.7
1	0.5

err_1	err_0
0.11	
	0.36
	0.51
0.36	
	0.69
	1.2
0.69	

이 오차들의 평균 계산



분류 모델 평가 : 예측 값 후속 처리

✓ 회귀와 다른 점

- 분류 모델 출력 층의 활성화 함수(시그모이드)
  - 예측 결과 : 0 ~ 1 사이 확률 값

✓ 예측 결과에 대한 후속 처리

- 결과를 .5 기준으로 잘라서 1, 0으로 변환
  - np.where(조건문, 참일 때 값, 거짓일 때 값)

```
1 pred = model.predict(x_val)
2 print(pred[:5])
```

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
[[0.65503603]
 [0.17545699]
 [0.22257133]
 [0.1949883 ]
 [0.66214496]]
```

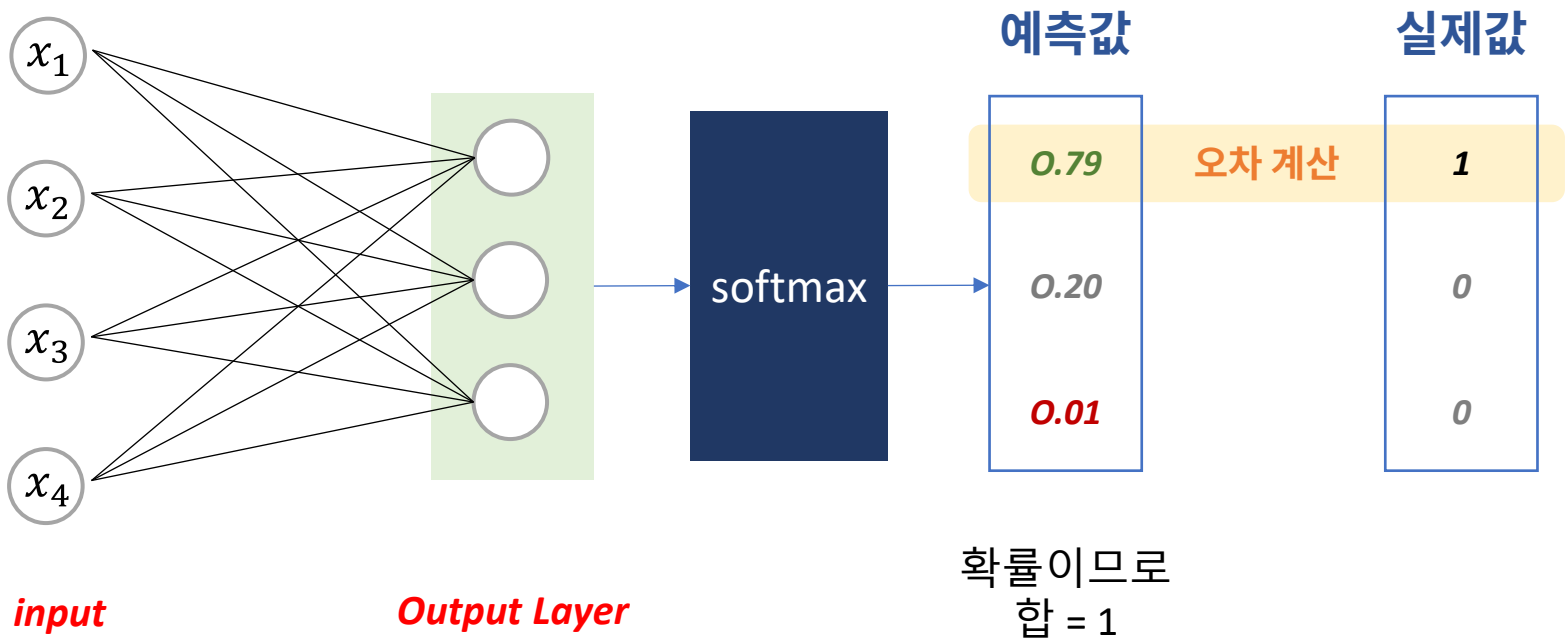
```
1 # activation이 sigmoid --> 0 ~ 1 사이의 확률값.
2 # 그러므로 cut-off value(보통 0.5)를 기준으로 잘라서 0과 1로 만들어 준다.
3 pred = np.where(pred >= .5, 1, 0)
4 print(pred[:5])
```

```
[[1]
 [0]
 [0]
 [0]
 [1]]
```

딥러닝 구조 - Output Layer

✓다중 분류 오차 계산 : Cross Entropy

- (다른 건 모르겠고) 실제 값 1인 Class와 예측 확률 비교



다중 분류 모델링을 위한 전처리

✓ 다중 분류 : Y가 범주이고, 범주가 3개 이상

- Y 값에 대한 전처리와 Loss Function
- 우리는 방법1을 이용하겠습니다.

	방법1	방법2
Y 전처리	✓ 정수 인코딩(0부터 시작) ▪ 예 : setosa(0) versicolor(1) virginica(2)	✓ One-hot Encoding [[0. 0. 1.] [0. 0. 1.] [1. 0. 0.] [0. 0. 1.] [1. 0. 0.]]
Loss Function	sparse_categorical_crossentropy	categorical_crossentropy

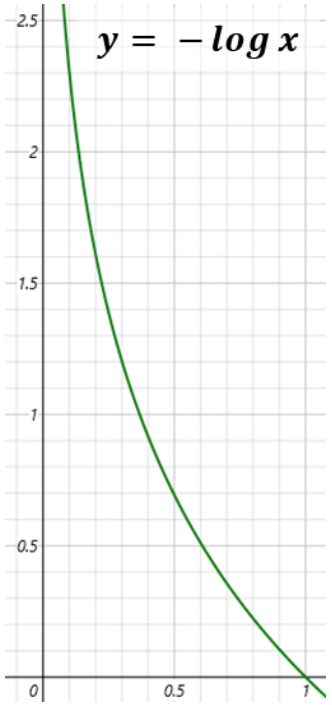
➢ **두 loss function은 수학적으로 동일**

[방법1]정수 인코딩 + sparse_categorical_crossentropy

✓ y : Integer Encoding

- loss='sparse_categorical_crossentropy'
- Y는 인덱스로 사용됨 : 해당 인덱스의 예측 확률로 계산. Loss = $-\log \hat{y}$

y	$\hat{y}$	err
1	[0.1, 0.6, 0.3]	$-log(0.6)$ = 0.51
0	[0.3, 0.2, 0.5]	$-log(0.3)$ = 1.20
2	[0.2, 0.4, 0.4]	$-log(0.4)$ = 0.91
1	[0.1, 0.9, 0.0]	$-log(0.9)$ = 0.10
0	[0.7, 0.1, 0.2]	$-log(0.7)$ = 0.36
0	[0.5, 0.3, 0.2]	$-log(0.5)$ = 0.69
2	[0.1, 0.1, 0.8]	$-log(0.8)$ = 0.22



다중 분류 모델의 평가

✓ 이진 분류와 다른 점

- 다중 분류 모델 **출력 층**
  - 노드 수 : 다중 분류 클래스의 수와 동일
  - 활성화 함수 : **softmax**

✓ 예측결과에 대한 후속 처리

- ① 예측결과 : 각 클래스별 확률값
- ② 그 중 가장 큰 값의 인덱스로 변환
  - np.argmax()

1
```
pred = model.predict(x_val)
pred[:5]
```

```
2/2 [==============================] - 0s 5ms/step
array([[9.6893382e-01, 3.0914659e-02, 1.5143632e-04],
       [1.7674142e-02, 7.9808110e-01, 1.8424478e-01],
       [2.1283615e-02, 7.2939050e-01, 2.4932583e-01],
       [1.0448594e-03, 3.7526634e-01, 6.2368864e-01],
       [4.4328466e-02, 8.7283564e-01, 8.2836017e-02]], dtype=float32)
```

2
```
pred_1 = pred.argmax(axis=1)
pred_1
```

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

요약 : 모델링

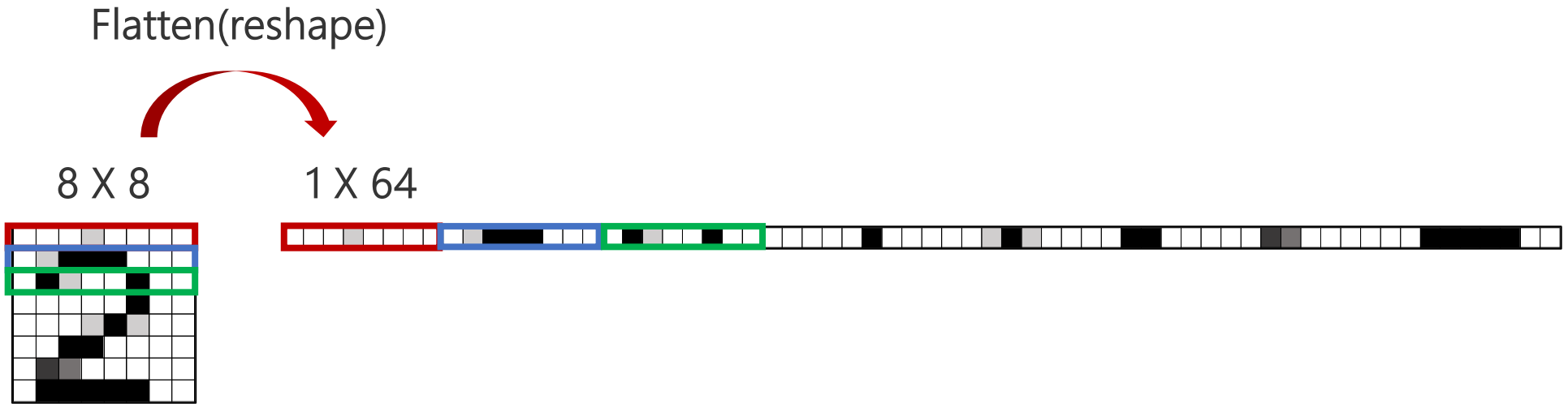
구분	Hidden Layer	Output Layer		Compile		예측	
	Activation	Activation	Node 수	optimizer	loss	결과	후속조치
Regression	relu	X	1	adam	mse	숫자	X
2-Class	relu	sigmoid	1	adam	binary_crossentropy	0~1 확률값	np.where()
Multi-Class	relu	softmax	Class수	adam	sparse_categorical_crossentropy categorical_crossentropy	각 클래스에 대한 확률값	np.argmax()

3일차 정리

이미지를 Dense Layer에 연결하여 모델링

✓ **2차원 이미지 데이터를 1차원으로 펼치는 Flatten함수가 필요합니다.**

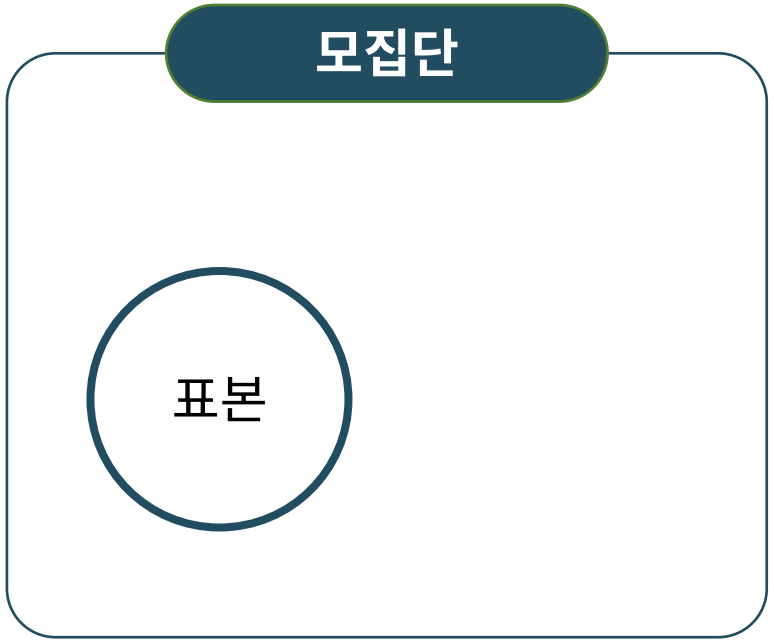
- Flatten은 다음과 같은 작업을 수행합니다.
- (Dense Layer = Fully Connected Layer)



모델링의 목적

✓모델링의 목적

- 학습용 데이터에 있는 패턴으로,
  그 외 데이터(모집단 전체)를 적절히 예측
- 학습한 패턴(모델)은,
  - 학습용 데이터를 잘 설명할 뿐만 아니라,
  - 모집단의 **다른 데이터(val, test)도 잘 예측**해야 함

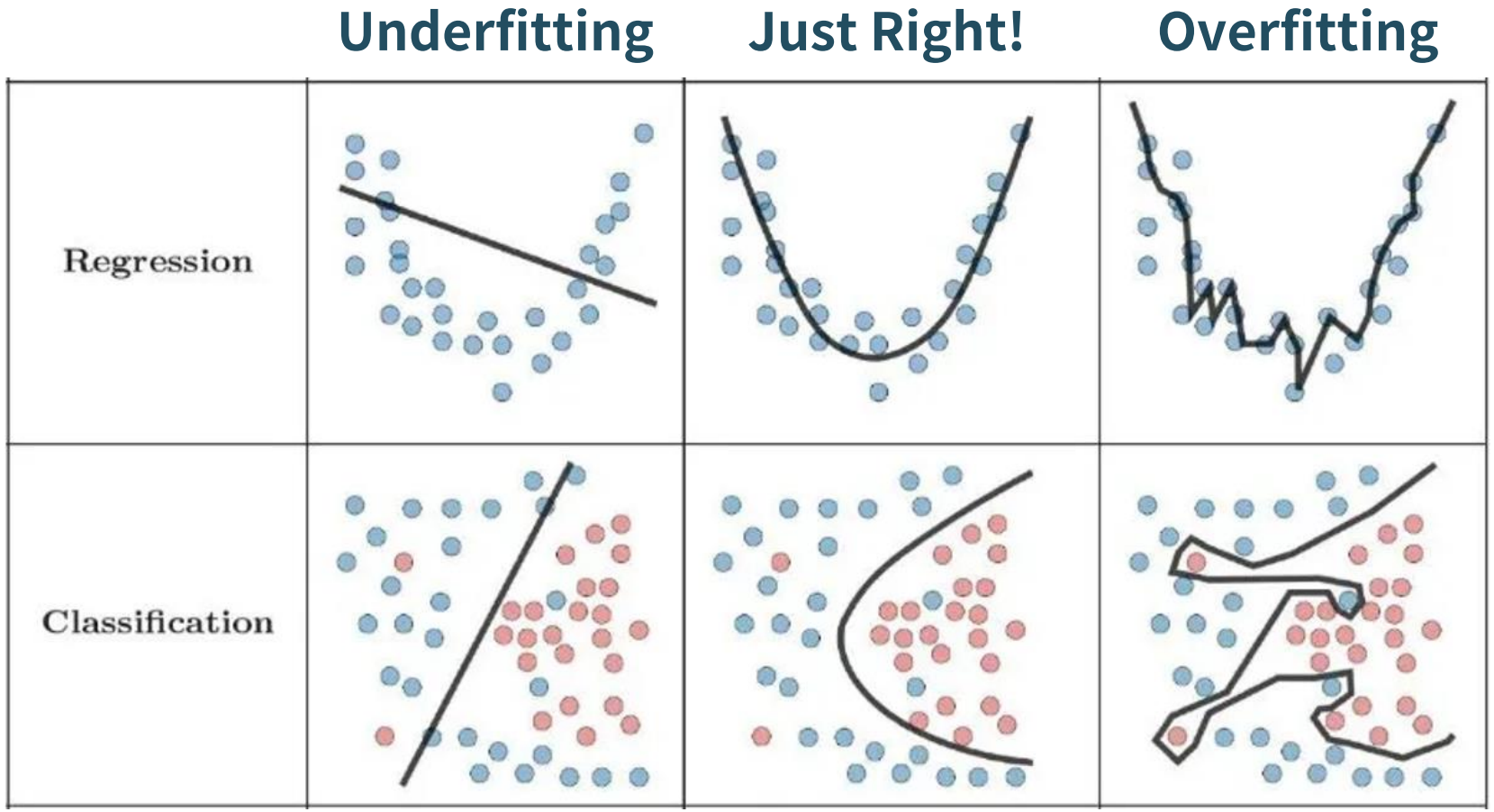


✓모델의 복잡도

- **너무 단순한** 모델 : train, val 성능이 떨어짐
- **적절히 복잡한** 모델 : 적절한 예측력
- **너무 복잡한** 모델 : train 성능 높고, val 성능 떨어짐

Underfitting과 Overfitting

✓ 모델(알고리즘)마다 복잡도를 결정하는 요인이 있음.



성능 최적화와 과적합의 관계

✓모델의 **복잡도** : 학습용 데이터의 패턴을 반영하는 정도

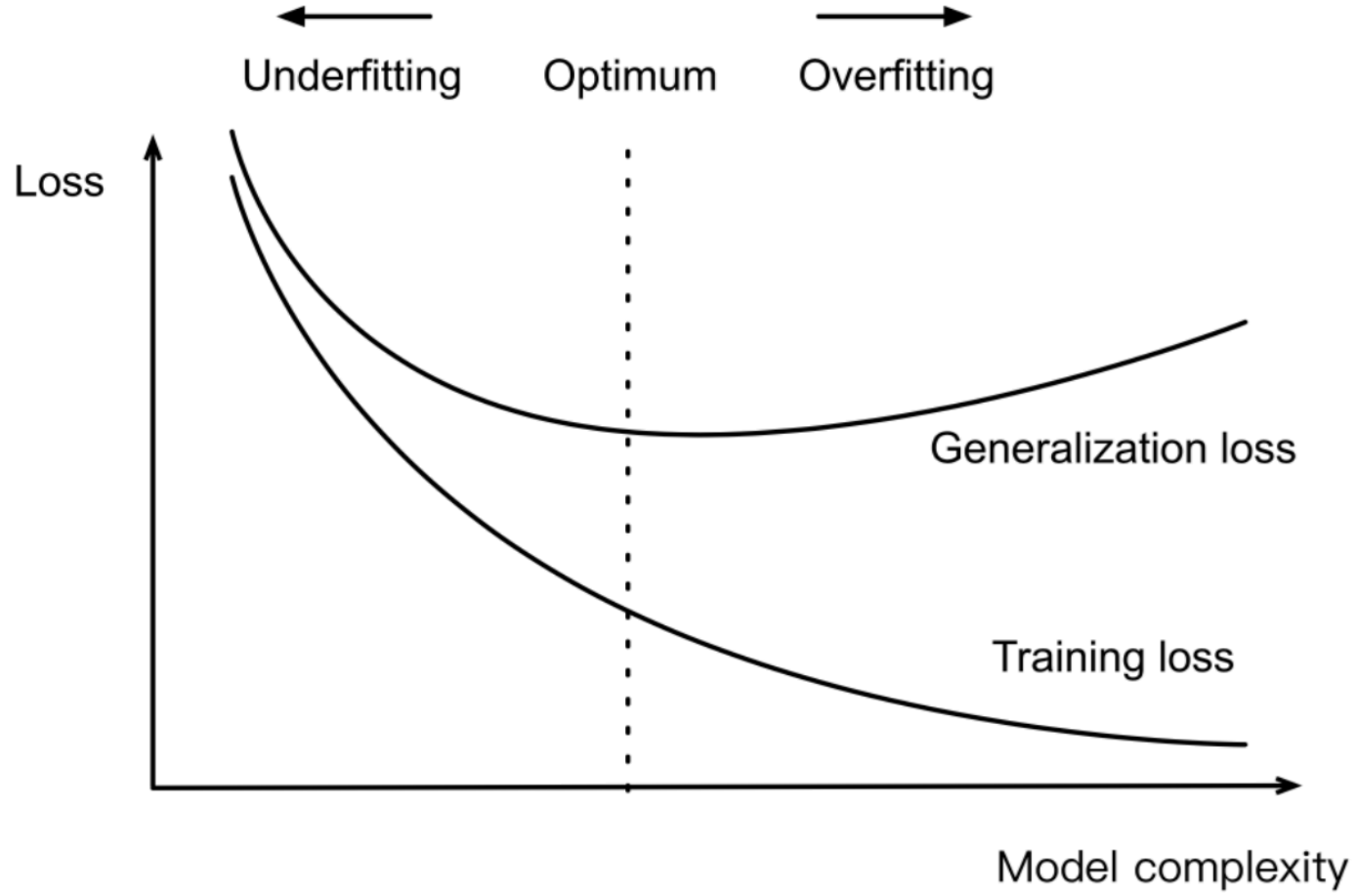
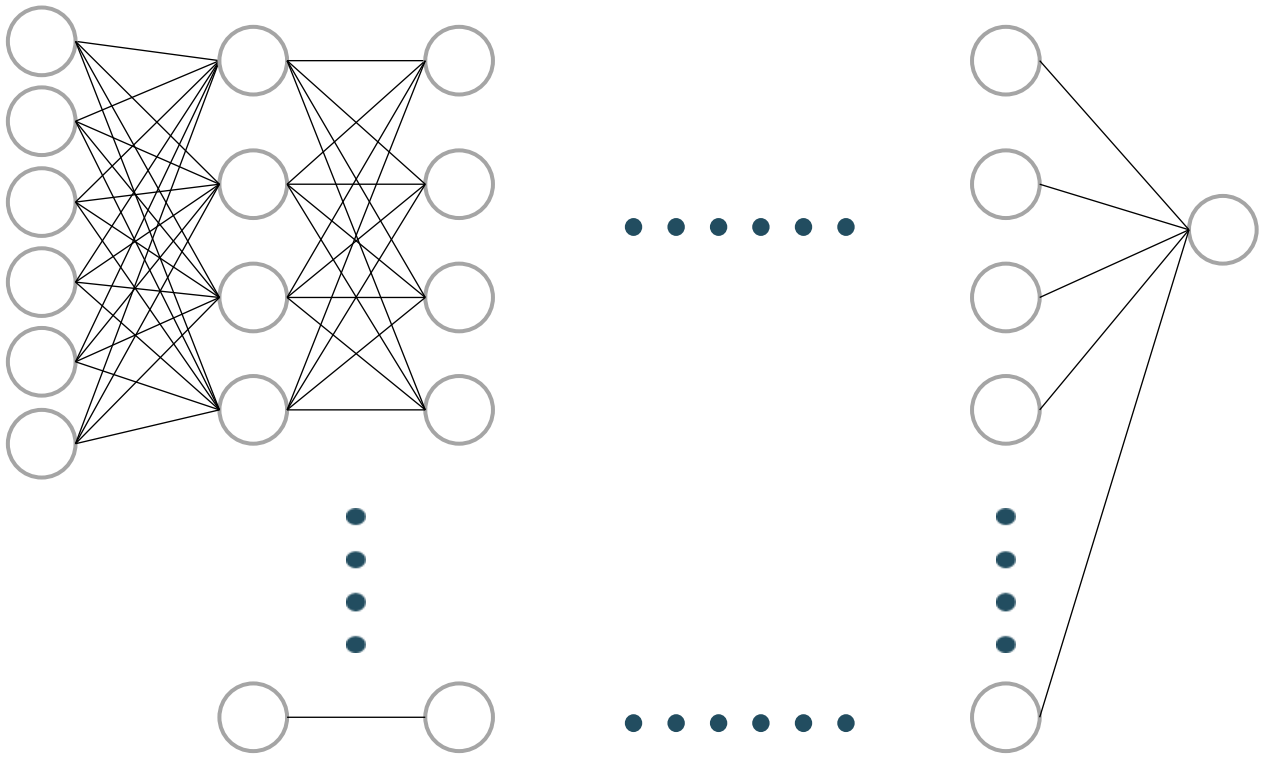


그림 출처 : https://vitalflux.com/model-complexity-overfitting-in-machine-learning/

딥러닝의 복잡도 : hidden layer 수, node 수

Layer가 많을 수록 복잡

Node가 많을 수록 복잡



최근 딥러닝 모델링 추세

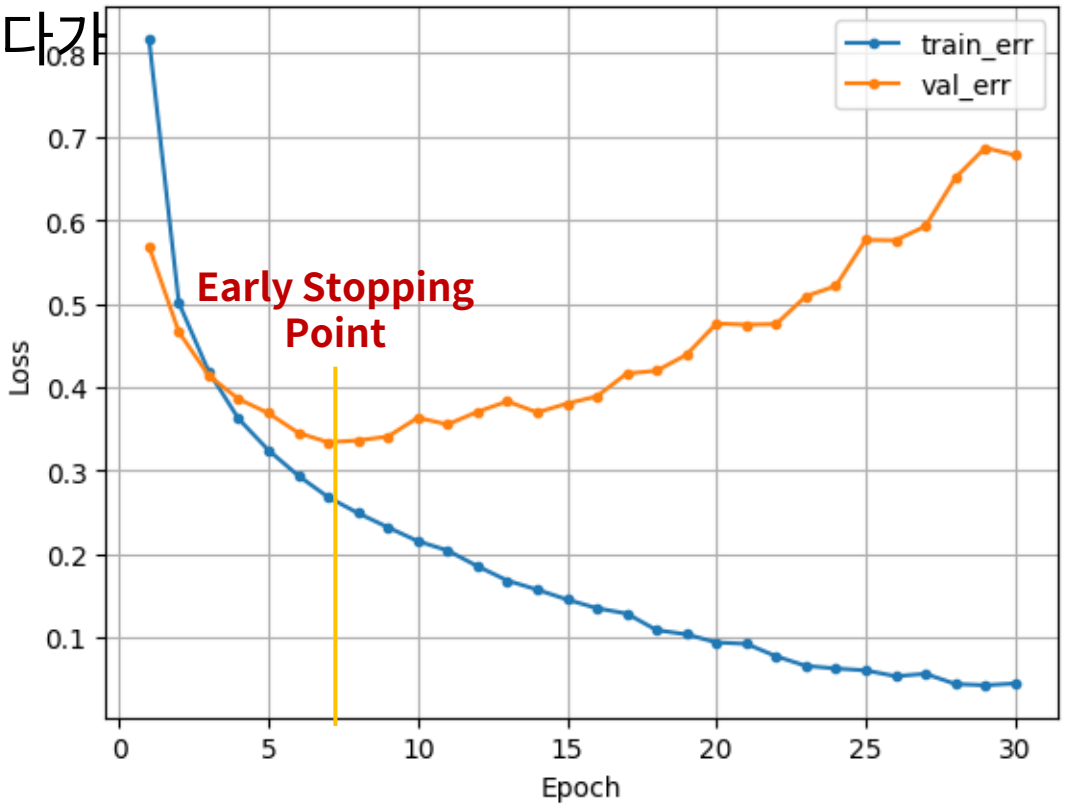
✓가능한한 복잡한 구조의 모델 설계

- 성능을 높이기 위해.
- 그러나 과적합 문제는?
  - 규제 기법(Regularization)을 통해 해결
  - Early Stopping 미리 멈춤
  - Dropout 뉴런 비활성화
  - 가중치 규제

미리 멈춤 Early Stopping

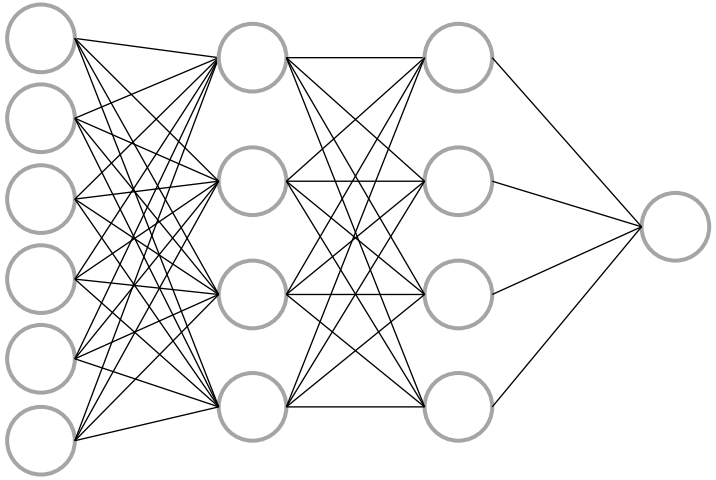
✓반복 횟수(epoch)가 많으면 과적합 될 수 있음

- 항상 과적합이 발생되는 것은 아닙니다.
- 반복횟수가 증가할 수록 val error가 줄어들다가 어느 순간부터 다시 증가할 수 있습니다.
- val error가 더 이상 줄지 않으면 멈춰라
  ➔ Early Stopping
- 일반적으로 train error는 계속 줄어듦

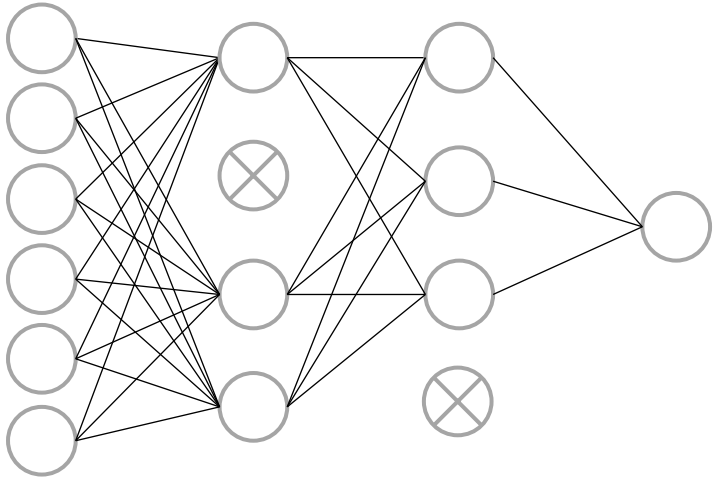


연결을 임의로 끊기 Dropout

Dropout 적용 전



Dropout 적용 후



모델 저장하기 : 최종 모델 저장하기

✓모델 저장하기

- model.save('파일이름.keras')
  - 딥러닝 모델의 메소드로 .save가 제공됩니다.
  - **파일이름.keras** 파일 저장(keras 2.11 이상 버전에서 지원)
    - 이전 버전인 .h5도 지원, 그러나 .keras 권장

```
model1.save('hanky.keras')
```

✓모델 로딩하기

- load_model 함수는 별도로 불러와야 합니다.
- 경로를 맞춰주고 .keras 파일을 읽어오면 그대로 사용 가능합니다.

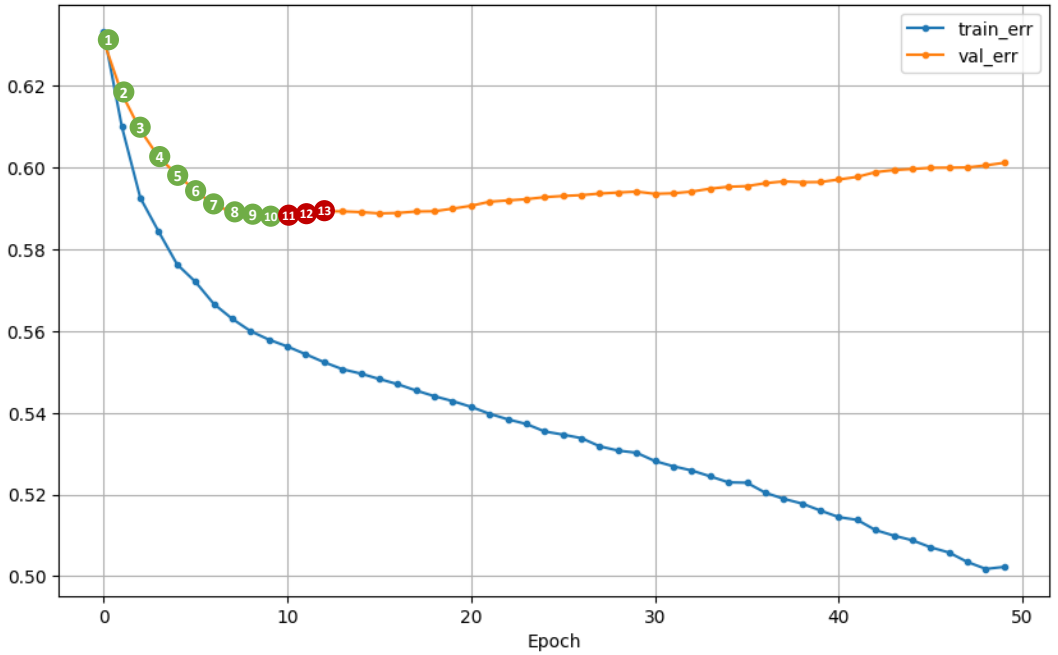
```
from keras.models import load_model
model2 = load_model('hanky.keras')
```

모델 저장하기 : 중간 체크포인트 저장하기

✓파일 저장 ModelCheckpoint

- 성능이 개선되는 구간 epoch 1 ~ 10 까지 파일 저장
- epoch11 부터는, 성능이 가장 좋은 epoch10 보다 성능이 개선되지 않음

```
...
Epoch 8: val_loss improved from 0.59081 to 0.58926, saving model to /content/008.h5
5/5 [==============================] - 0s 27ms/step - loss: 0.5630 - val_loss: 0.5893
Epoch 9/50
1/5 [=====>........................] - ETA: 0s - loss: 0.6025
Epoch 9: val_loss improved from 0.58926 to 0.58859, saving model to /content/009.h5
5/5 [==============================] - 0s 27ms/step - loss: 0.5600 - val_loss: 0.5886
Epoch 10/50
1/5 [=====>........................] - ETA: 0s - loss: 0.5417
Epoch 10: val_loss improved from 0.58859 to 0.58808, saving model to /content/010.h5
5/5 [==============================] - 0s 22ms/step - loss: 0.5579 - val_loss: 0.5881
Epoch 11/50
1/5 [=====>........................] - ETA: 0s - loss: 0.4853
Epoch 11: val_loss did not improve from 0.58808
5/5 [==============================] - 0s 18ms/step - loss: 0.5562 - val_loss: 0.5883
Epoch 12/50
1/5 [=====>........................] - ETA: 0s - loss: 0.5444
Epoch 12: val_loss did not improve from 0.58808
...
```



Function API 코드 연습

Sequential

- Sequential 함수 안에 리스트로 레이어 입력

```
clear_session()

model = Sequential([
        Dense(18 ,input_shape = (nfeatures, ),
              activation = 'relu' ),
        Dense(4, activation='relu') ,
        Dense(1) ])

model.summary()
```

Functional

- 레이어 : 앞 레이어 연결 지정
- Model 함수로 **시작**과 **끝** 연결해서 선언

```
clear_session()

il = Input(shape=(nfeatures, ))
hl1 = Dense(18, activation='relu')(il)
hl2 = Dense(4, activation='relu')(hl1)
ol = Dense(1)(hl2)

model = Model(inputs = il, outputs = ol)

model.summary()
```

설문 질문들

✓평가 지표 해석

- 평가지표 해석 및 비즈니스의미 도출
- MAE, MSE, MAPE, Classification Report

✓모델 구조

- 히든 레이어의 노드 수 정하기
- 히든 레이어의 활성화 함수

✓모델 후속조치

- np.where
- np.argmax

설문 질문들

✓그외

- resampling 알려주셔서 와인실습에서 4번 샘플이너무 적어서 와인실습에 적용해봤는데 그래프가 이상한데 시간이 된다면 한번 해봐주실수 있나요
  ➔ 콜랩파일 저에게 공유해주세요.^^