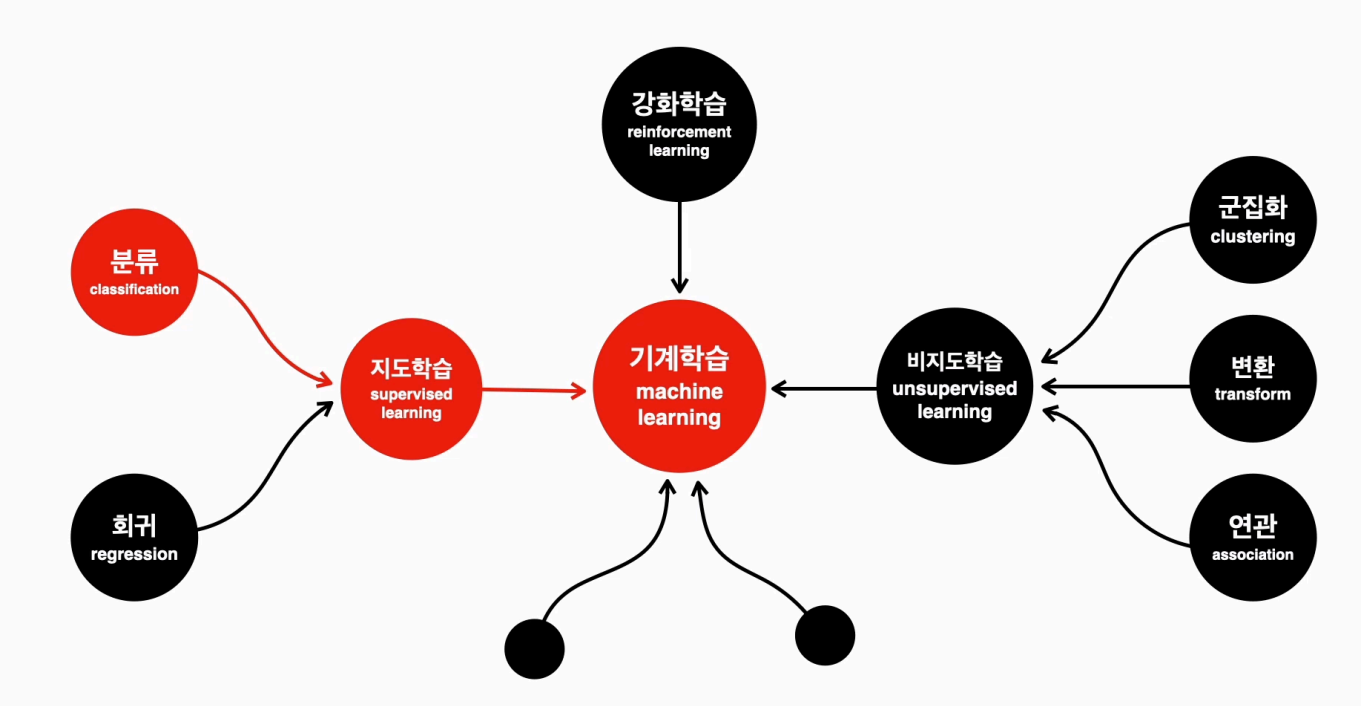
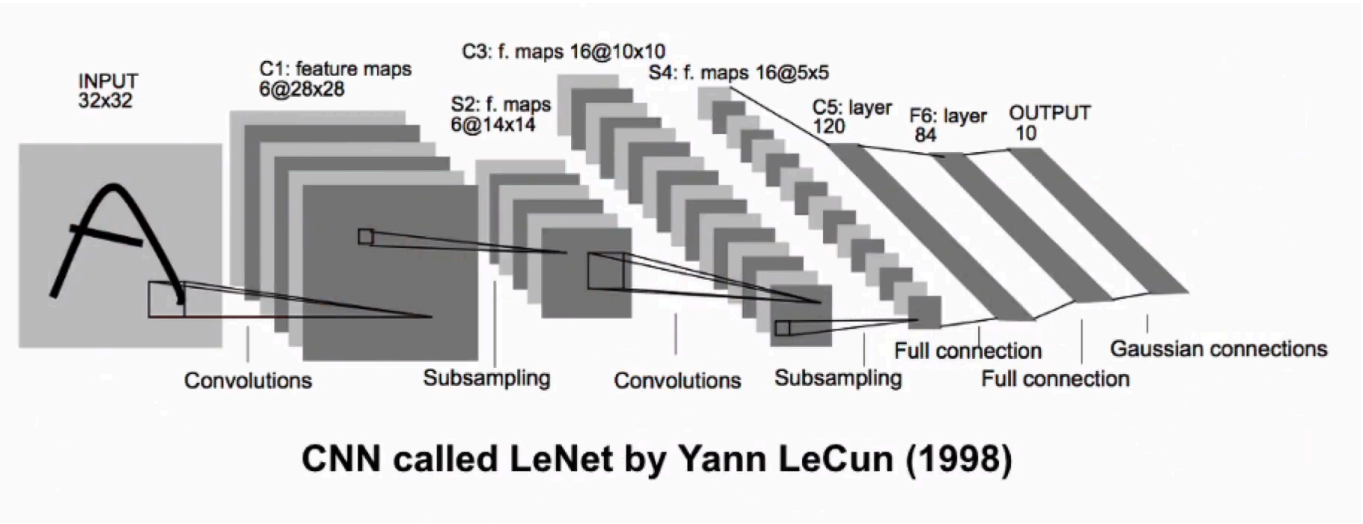
이미지 분류기

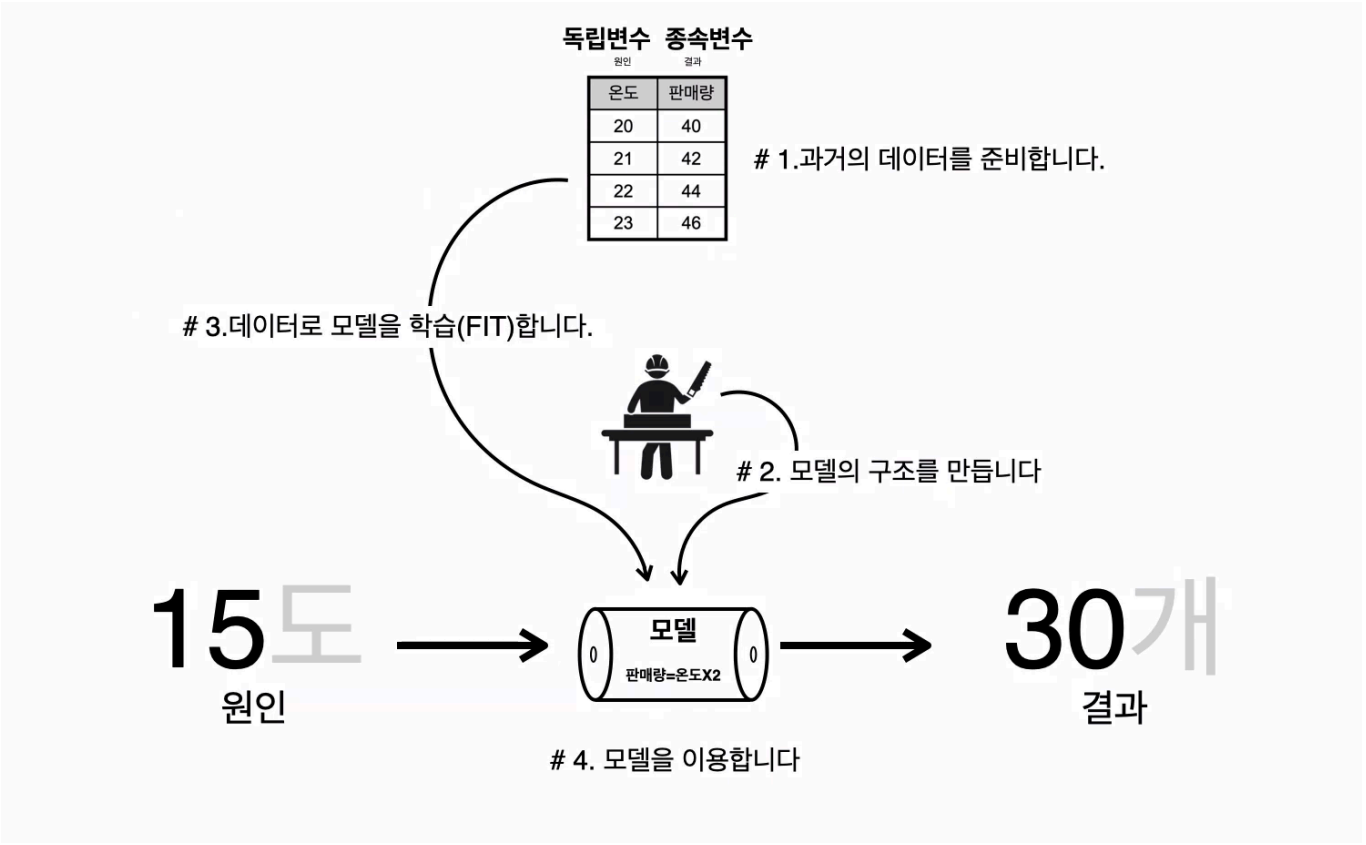
지금 부터 딥러닝을 만드는 것을 시작해 보겠습니다. 우리는 함께 이미지 분류기를 만들게 됩니다.



이미지분류기는 머신러닝 중에 지도학습 문제이고 분류 문제 입니다. 이미지를 학습하는 가장 기본적인 딥러닝 모델로 CNN 컴포지션 뉴럴 네트워크라는 새로운 딥러닝 구조도 배우게 됩니다.



지금 보시는 모델은 CNN 구조로 구현된 르넷이라는 이름의 모델입니다. 지금은 이 그림이 무얼 말하는지 하나도 알 수 없을 것입니다. 우리 수업이 마칠 때에 이 그림을 보는 눈이 달라져 있을 것입니다. 위그림이 굉장히 쉽게 느껴지고 코드가 머리속에 흘러가는 걸 경험하게 되실 것입니다. 그럼 시작해 보겠습니다.



이번 그림은 머신러닝 모델을 만드는 과정을 설명하고 있습니다. 이장치는 앞에 수업에서 봤던 그림입니다. 과거의 데이터를 준비하고 모델의 구조를 만들고 데이터로 모델을 학습하고 모델을 이용합니다. 이미지 학습을 위한 첫번째 전체 코드를 먼저 구경해 보겠습니다.

1.과거의 데이터를 준비합니다.

```
(독립, 종속), _ = tf.keras.datasets.mnist.load_data()
독립 = 독립.reshape(60000, 784) # 독립변수 만드는 곳
종속 = pd.get_dummies(종속) # 원핫 인코딩
print(독립.shape, 종속.shape)
```

2.모델의 구조를 만듭니다.

```
X = tf.keras.layers.Input(shape=[784]) # 독립변수 784개
H = tf.keras.layers.Dense(84, activation='swish')(X)
Y = tf.keras.layers.Dense(10, activation='softmax')(H) # 종속변수 10개
model = tf.keras.models.Model(X, Y)
model.compile(loss='categorical_crossentropy', metrics=['accuracy'])
```

3.데이터로 모델을 학습(FIT)합니다.

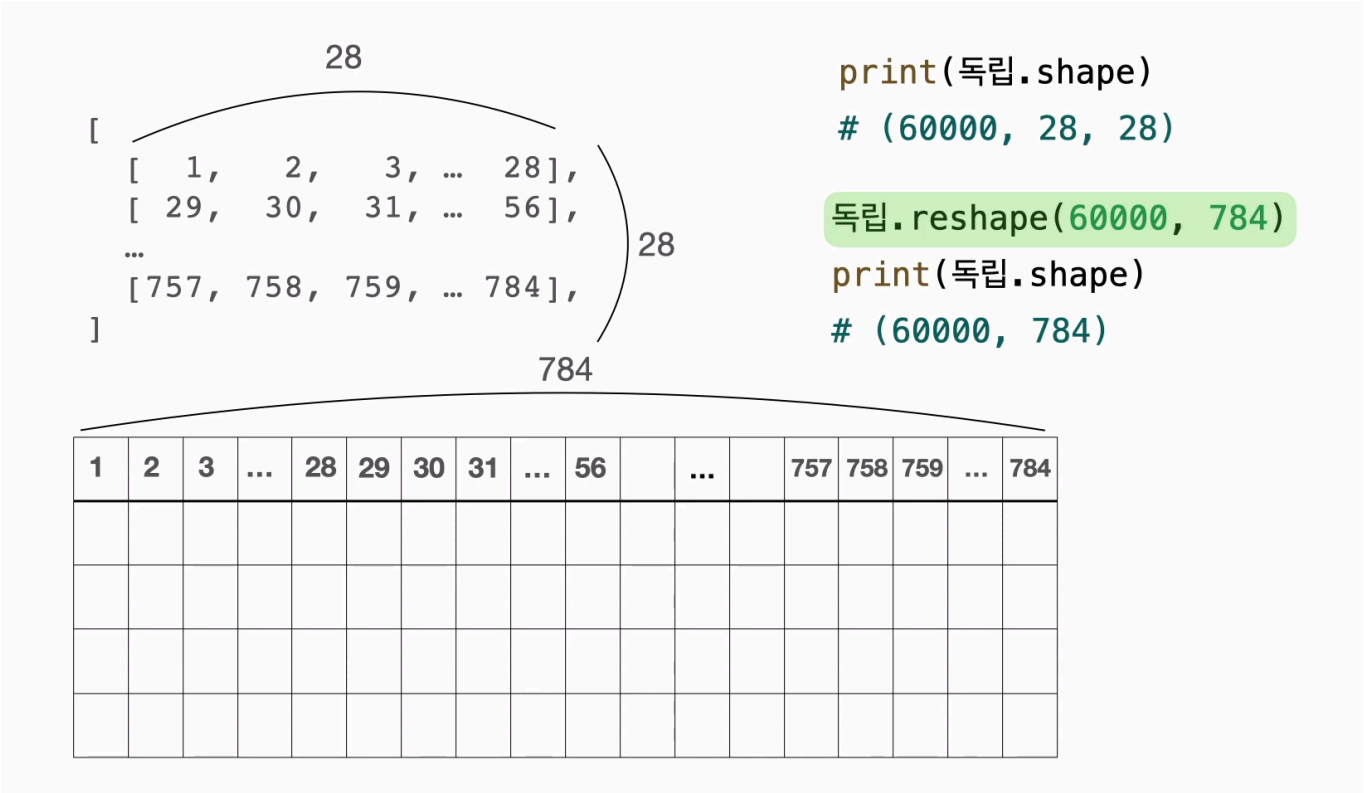
```
model.fit(독립, 종속, epochs=10)
```

4.모델을 이용합니다.

```
pred = model.predict(독립[0:5])
pd.DataFrame(pred).round(2)
```

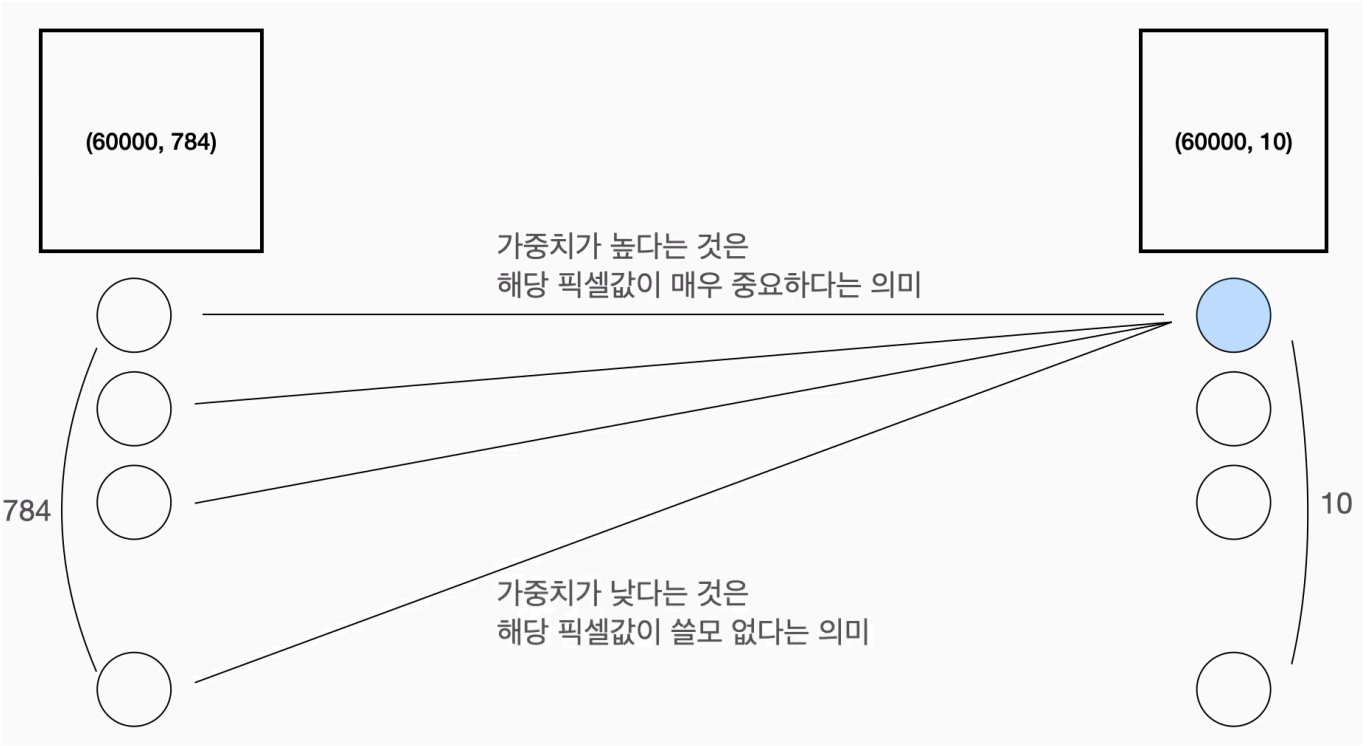
코드를 보시니 어떤가요? 표를 학습했던 이전에 모델과 동일한 모습의 모델 구조가 보이시나요? 독립변수가 784개의 컬럼으로 되어 있고, 종속변수가 10개의 컬럼으로 되어 있는 표를 학습하는 모델입니다. 0부터 9까지 10개의 범주에 숫자가 있으니 원핫인코딩을 하여 종속변수가 10개의 컬럼이 되는 건 딱 보이네요. 그런데 이미지를 학습하기로 했는데 표를 학습한 모델이라? 이상하지 않으신가요? 독립변수를 만드는 부분을 눈여겨 보면 그 비밀이 숨겨져 있습니다. 어쩐지 코드를 사용하는게 어렵지 않아 보이네요.

여기 한장의 이미지 데이터가 있습니다. (28, 28) 2차원 형태입니다.



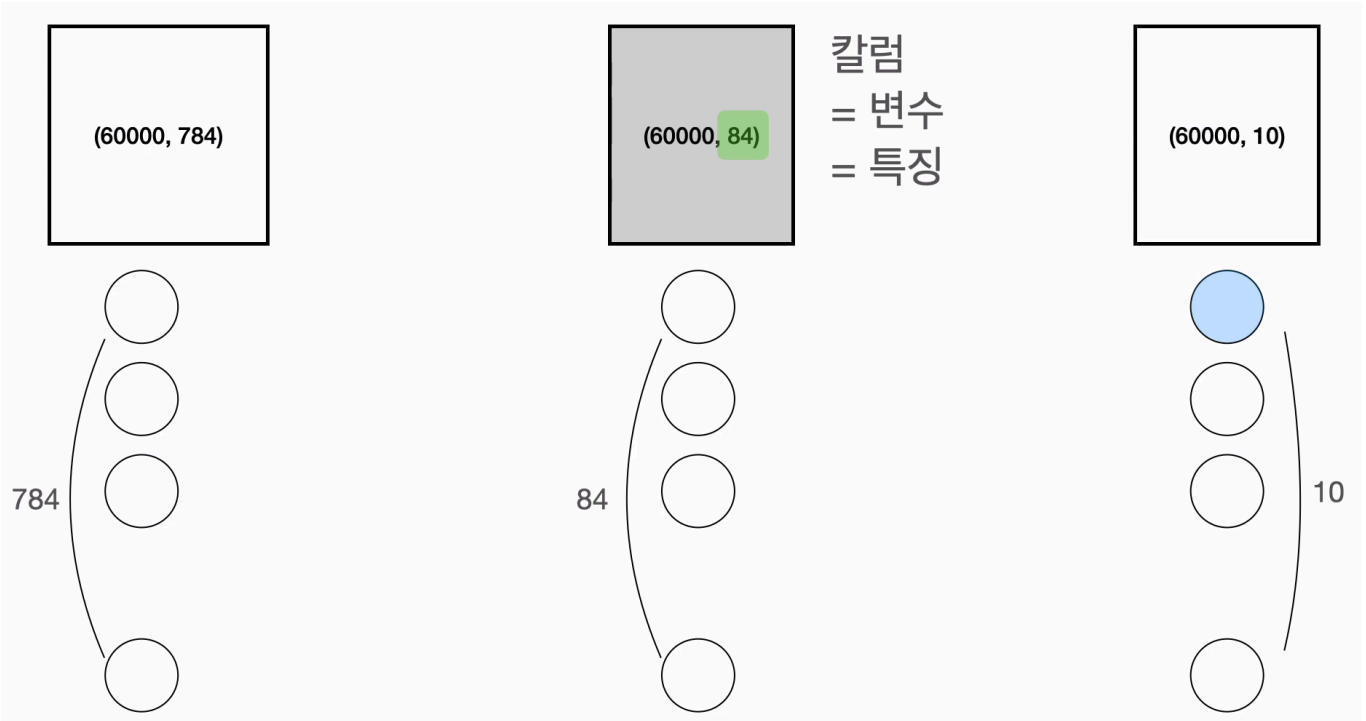
그리고 하단에 있는 표를 준비합니다. 컬럼 784개로 되어 있는 표입니다. 우리가 할 일은 준비한 이미지 데이터를 이 표에 우겨 넣는 것입니다. 첫번째 행은 처음 28개의 칸에 두번째는 그다음 28칸, 마지막행까지 모든 행을 28개의 칸에 순차적으로 섞어 넣으면 이미지 한장의 784개의 컬럼으로 옮길 수 있습니다. 전체 6만장의 데이터는 (60000, 28, 28)의 형태 였습니다. 784개의 컬럼을 가진 표에 형태로 변형을 해 준것이죠 이렇게 데이터의 형태를 변형 해주는 도구가 바로 reshape 입니다. 데이터가 표로 바뀌면 우리는 아무런 두려움이 없습니다. 표를 학습 시키는 건 너무나 쉽습니다..

우리가 알고 있는 모델을 조금더 살펴보겠습니다. 컬럼이 784개인 독립변수 컬럼이 열개인 종속변수를 가지고 학습을 한다면 입력층의 노드는 784개 출력층은 10개의 노드로 이루어진 모델을 만들게 됩니다.



만약 히든레이어 없이 모델을 만들었다면 첫번째 결과 그 숫자가 0인지 아닌지를 판단하는 수식에서 입력층 노드의 모든 입력의 가중치를 붙여서 판단하게 되겠죠. 모든 픽셀 값에 대해 가중치를 학습하게 되는 것입니다. 학습이 끝났을 때 우리는 다음과 같이 해석할 수 있겠습니다. 가중치가 높게 부여된 픽셀은 0이라고 판단하기에 중요한 픽셀이다. 가중치가 0에 가까운 픽셀은 0이라고 판단하는 데에 불필요한 픽셀이다 라구요.

84개의 노드를 가진 히든레이어를 추가하여 모델을 만들어 봅시다.



그것은 중간 결과로 컬럼 84개를 가진 표가 만들어지는 것입니다. 히든 레이어가 있는 상황에서는 첫번째 결과가 0인지 아닌지를 위한 수식에서 히든레이어의 모든 노드가 입력으로 사용되는 것이고 모든 입력의 가중치를 붙여서 판단하게 되겠죠. 이 84개의 중간 결과는 인공신경망 구조안에서 가중치 학습을 통해 컴퓨터 스스로 만들어낸 값입니다. 우리는 앞서 각

컬럼은 특징 이라는 이름을 가지고 있다고 배웠습니다. 이곳의 특징 84개는 컴퓨터가 인공신경망 구조 안에서 학습을 통해 스스로 찾은 것입니다. 그래서 노드가 84개 있는 은닉층을 추가한 행동은 컴퓨터에게 이런 명령을 하는 것입니다.

"컴퓨터야 이 이미지들이 **0~9**까지 중 어느 숫자인지 판단하기 위해 가장 좋은 특징 **84**개를 찾아줘"

라고 말이죠. 즉 최종 결과를 더 잘 낼 수 있는 특징 84개를 인공신경망이 찾은 겁니다. 그래서 인공 신경망의 "특징 자동추측기"라는 별명이 있습니다.

코드를 다시보면 위에 코드만으로 충분히 좋습니다. 전혀 문제가 없는데요 다만 이 다음 과정을 위해 위 코드를 조금 변형 하려고 합니다.

1.과거의 데이터를 준비합니다.

```
(독립, 종속), _ = tf.keras.datasets.mnist.load_data()
독립 = 독립.reshape(60000, 784) # 이것을 사용 안함.
종속 = pd.get_dummies(종속) # 원핫 인코딩
print(독립.shape, 종속.shape)
```

리쉐이프 대신 플래튼이라는 것을 사용할 것입니다. 그래서 리셰이프 를 지웁니다. 그러면 모델의 모양은 (60000, 784)가 아닌 (60000, 28, 28)이 됩니다.

- 1.과거의 데이터를 준비합니다.

  ```
  (독립, 종속), _ = tf.keras.datasets.mnist.load_data()
  종속 = pd.get_dummies(종속) # 원핫 인코딩
  print(독립.shape, 종속.shape)
  ```

그래서 아래 입력 부분에 셰프는 784가 아닌 28 X 28 로 고칩니다. 플래튼 코드를 추가합니다. 리쉐이프애 해당하는 기능을 모델 내에서 동작시키는 역할을 합니다. 히든 레이어의 입력을 X 에서 H 로 바꿔주는 걸 잊지 말아야 하겠습니다.

- 2.모델의 구조를 만듭니다.

  ```
  X = tf.keras.layers.Input(shape=[28, 28]) # 여기가 28, 28로 변함
  H = tf.keras.layers.Flatten()(X) # 여기에 플래튼 추가
  H = tf.keras.layers.Dense(84, activation='swish')(H) # X 에서 -> H로
  Y = tf.keras.layers.Dense(10, activation='softmax')(H) # 종속변수 10개
  model = tf.keras.models.Model(X, Y)
  model.compile(loss='categorical_crossentropy', metrics=['accuracy'])
  ```

- 3.데이터로 모델을 학습(FIT)합니다.

  model.fit(독립, 종속, epochs=10)

- 4.모델을 이용합니다.

  pred = model.predict(독립[0:5])
  pd.DataFrame(pred).round(2)

자 이미지의 픽셀을 표로 나타내어 학습하게 되는 모델을 완성했습니다. 입력의 모양과 플래튼 레이어를 기억하면 됩니다.

```python
# -*- coding: utf-8 -*-
import tensorflow as tf
import pandas as pd

(독립, 종속), _ = tf.keras.datasets.mnist.load_data()
print(독립.shape, 종속.shape)

독립 = 독립.reshape(60000, 784)
종속 = pd.get_dummies(종속)
print(독립.shape, 종속.shape)

X = tf.keras.layers.Input(shape=[784])
H = tf.keras.layers.Dense(84, activation='swish')(X)
Y = tf.keras.layers.Dense(10, activation='softmax')(H)
model = tf.keras.models.Model(X, Y)
model.compile(loss='categorical_crossentropy', metrics=['accuracy'])

model.fit(독립, 종속, epochs=10)

pred = model.predict(독립[0:5])
pd.DataFrame(pred).round(2)

종속[0:5]

(독립, 종속), _ = tf.keras.datasets.mnist.load_data()
print(독립.shape, 종속.shape)

# 독립 = 독립.reshape(60000, 784)
종속 = pd.get_dummies(종속)
print(독립.shape, 종속.shape)

X = tf.keras.layers.Input(shape=[28, 28])
H = tf.keras.layers.Flatten()(X)
H = tf.keras.layers.Dense(84, activation='swish')(H)
Y = tf.keras.layers.Dense(10, activation='softmax')(H)
model = tf.keras.models.Model(X, Y)
model.compile(loss='categorical_crossentropy', metrics=['accuracy'])

model.fit(독립, 종속, epochs=5)

pred = model.predict(독립[0:5])
pd.DataFrame(pred).round(2)
```