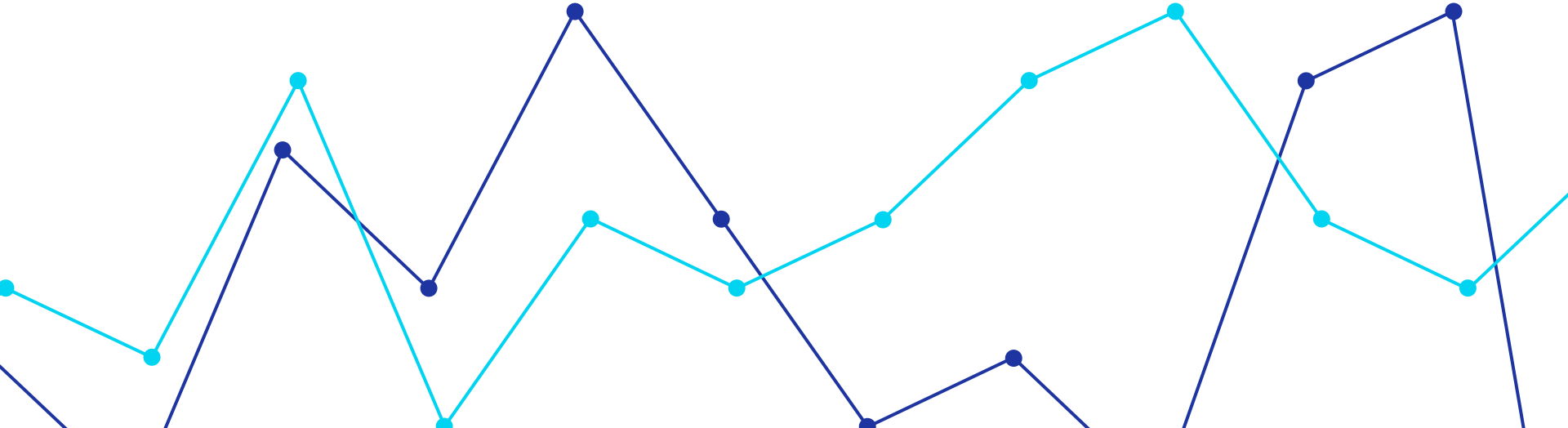
딥러닝의 동작 원리 _ 10-11장

2023 인공지능 기초 스터디 1팀



CONTENT

1부

1. 모델 설계하기
   - 모델의 정의
   - 입력층, 은닉층, 출력층

2. 오차함수(손실함수)
   -모델 컴파일
   -교차 엔트로피 계열의 함수
   -원-핫 인코딩함수

1. 데이터 다루기
   데이터 조사/가공

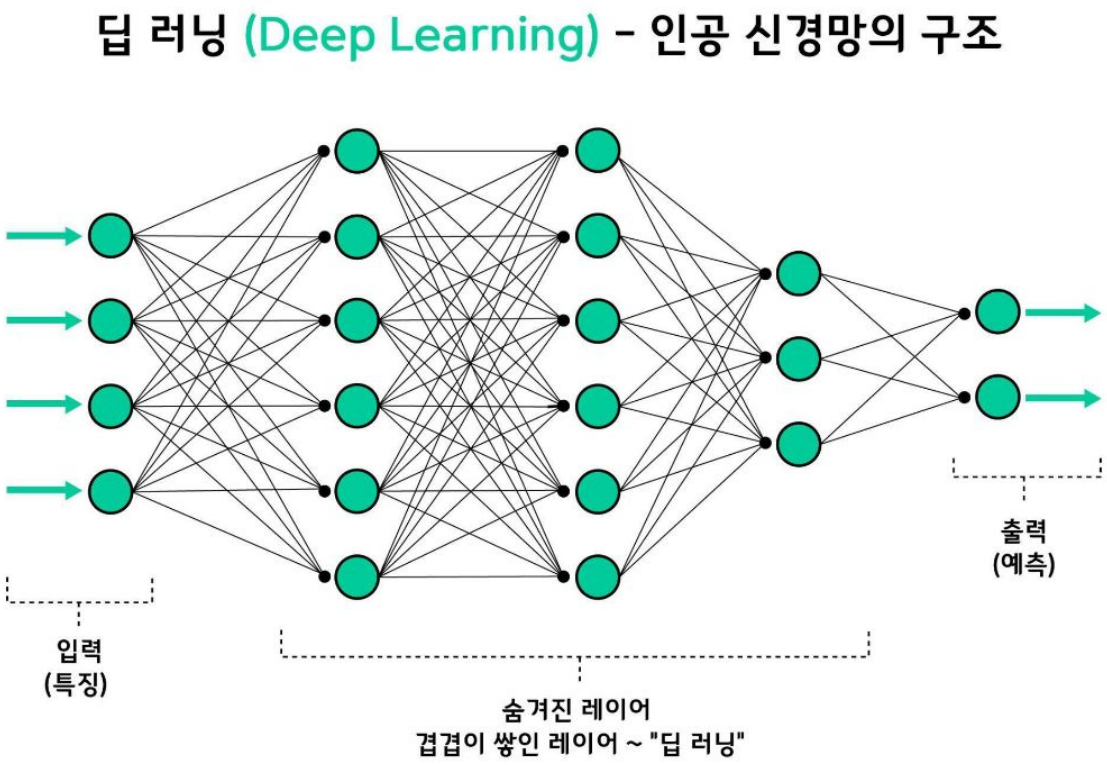
2. matplotlib 그래프 그리기
- Heatmap
- Seaborn 라이브러리

모델의 정의

딥러닝의 모델 설정 & 구동

1. 딥러닝의 구조를 짜고 층을 설정
2. 정해진 모델을 컴퓨터가 알아들을 수 있게끔 컴파일
3. 모델을 실제로 수행

입력층,은닉층,출력층



Colab csv파일

https://colab.research.google.com/drive/1r7_eB1SKsDnc2VlbzbDZ9f6pljO87LsE?hl=ko#scrollTo=QPYr-pzJtbDJ

딥러닝의 구조를 짜고 층을 설정

```
model = Sequential()
```

```
model.add(Dense(30, input_dim=17, activation='relu'))
```

레이어 추가

- 이 층에 30개 노드를 만들기
- 입력 데이터에서 몇 개의 값을 가져올지
- 활성화 함수 relu : 은닉층으로 학습

```
model.add(Dense(1, activation='sigmoid'))
```

레이어 추가
마지막 층 => 출력층

- 출력층 노드 1개
- sigmoid : 결과값 [0,1]

Pre_class 해설

Q1. 딥러닝의 모델을 설정하고 구동하는 부분에 대한 내용으로 옳지 않은 것을 모두 고르시오. *

- Sequential() 함수는 딥러닝의 구조를 한 층 한 층 쉽게 쌓아올릴 수 있게 해준다.
- model.add(Dense(4, activation = 'sigmoid')) 이 층의 활성화 함수는 셋 이상의 선택지 중 하나를 택하는 다중 클래스 분류 문제에서 출력층에 주로 사용된다.
- 레이어를 여러 개 추가할 때, 모든 Dense()에 항상 input_dim 인자를 써줘야 한다.
- model.add(Dense(1, activation = 'relu')) 이 층의 노드 수는 1개이다.
- .add() 메소드를 통해서 쉽게 레이어를 추가할 수 있다.

소프트맥스 함수

정답일 확률

0.8

0.2

0.4

Softmax 함수

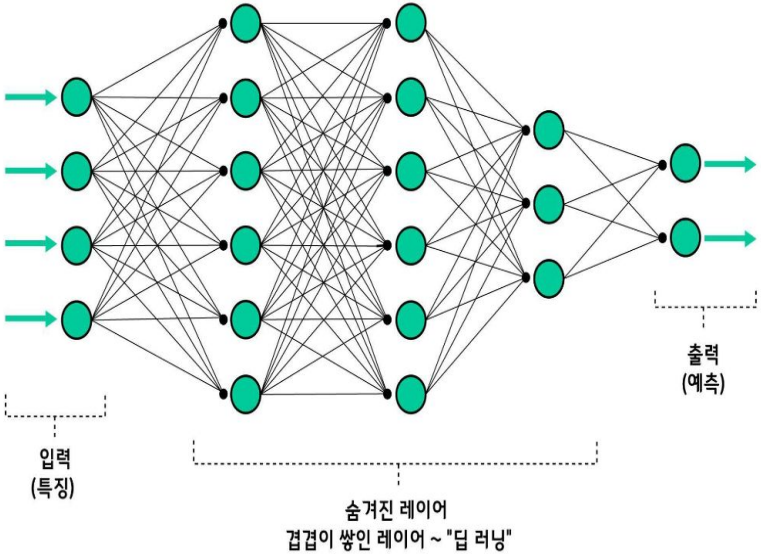
정답일 확률

0.7

0.05

0.25

깜짝 Quiz!



Q. 빨간색 물음표에 어떤 숫자를 써야할까요~?

```
model = Sequential()
model.add(Dense(6, input_dim = ? , activation = 'relu'))
model.add(Dense(? , activation = 'relu'))
model.add(Dense(? , activation = 'relu'))
model.add(Dense(? , activation = 'sigmoid'))
```

케라스 작업 흐름

1. 입력 텐서와 타깃 텐서로 이루어진 훈련 데이터를 정의.
2. 입력과 타깃을 매핑하는 층으로 이루어진 모델을 정의.
3. 손실 함수, 옵티마이저, (모니터링)측정 지표를 선택하여 학습 과정을 설정.
4. 훈련 데이터에 대해 모델의 fit()메서드를 반복적으로 호출.

Sequential 클래스→ 함수형 API

모델 정의

Sequential 클래스

```
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(32, activation='relu', input_shape=(784,)))
model.add(layers.Dense(10, activation='softmax'))
```

함수형 API

```
input_tensor = layers.Input(shape=(784,))
x = layers.Dense(32, activation='relu')(input_tensor)
output_tensor = layers.Dense(10, activation='softmax')(x)

model = models.Model(inputs=input_tensor, outputs=output_tensor)
```

모델 컴파일

모델을 학습 시키기 이전에 학습 방식에 대한 환경설정

```
model.compile( loss = 'mean_squared_error' , optimizer = ‘adam', metrics= [ 'accuracy' ])
```

- 손실 함수 (loss function) → 모델 최적화에 사용되는 목적 함수 (종류 : 뒷장 설명 참고)
- 정규화기 (optimizer) → 훈련 과정을 설명, 최적화 알고리즘 설정 (종류 : adam, sgd, rmsprop, adagrad 등)
- 평가지표 (metric) → 훈련을 모니터링 하기 위함 (종류 : 분류 - accuracy, 회귀 - mse, rmse, r2, mae, mspe, mape, msle 등)

오차함수

지도학습 과정에서 오차의 크기를 계산하기 위해 사용되는 함수
풀고자 하는 지도학습문제에 따라 적당한 오차 함수를 선택.

평균 제곱 계열

평균제곱오차
(mean_squared_error)

평균 절대 오차
(mean_absolute_error)

평균 절대 백분율 오차
(mean_absolute_percentage_error)

평균 제곱 로그 오차
(mean_squared_logarithmic_error)

교차 엔트로피 계열

이항 교차 엔트로피
(binary_crossentropy)

범주형 교차 엔트로피
(categorical_crossentropy)

원-핫 인코딩

표현하고 싶은 단어의 인덱스에 1의 값을,
다른 인덱스에는 0의 값을 부여하는 백터 표현 방식

예)

I am not a psychopath. Anderson.

단어	단어 인덱스	원-핫 백터
I	0	[1, 0, 0, 0, 0, 0]
am	1	[0, 1, 0, 0, 0, 0]
not	2	[0, 0, 1, 0, 0, 0]
a	3	[0, 0, 0, 1, 0, 0]
psychopath	4	[0, 0, 0, 0, 1, 0]
Anderson	5	[0, 0, 0, 0, 0, 1]

교차 엔트로피 오차

원-핫 인코딩일때의 출력 함수

$$E = -\sum_k t_k \log y_k$$

```
def cee(y, t):
    delta = 1e-7
    return -np.sum(t*np.log(y+delta))
```

실제로 사용할 때
작은 값을 더함으로서
INF가 발생하지 않도록 하는 역할

딥러닝) 신경망 학습 , 손실 함수 (오차 제곱합 , 교차 엔트로피 오차), 미니배치 (tistory.com)
Ⅵ. 언어와 소통 – 5. 자연어처리 기법(2) : 원-핫 인코딩, Word2Vec – AI4School

모델 컴파일

```
model.compile( loss = 'mean_squared_error' )
```

```
model.compile( loss = 'binary_crossentropy' )
```

용어 정리

	A	B	C	D	E	F	G
1	class	cap-shape	cap-surfac	cap-color	bruises	odor	gill-attachr
2	p	x	s	n	t	p	f
3	e	x	s	y	t	a	f
4	e	b	s	w	t	l	f
5	p	x	y	w	t	p	f
6	e	x	s	g	f	n	f
7	e	x	y	y	t	a	f
8	e	b	s	w	t	a	f
9	e	b	y	w	t	l	f

속성

샘플

가중치
업데이트 횟수

클래스

에포크(epochs)
2 일 경우
10 일 경우

batch_size
2 일 경우
4 일 경우

(8/2)* 2
(8/4)* 10

실습 과제 설명

1. 딥러닝과 데이터

' 머신러닝 프로젝트는 **좋은 데이터**의 영향을 받음 '

좋은 데이터란?

① 알고자 하는 정보를 잘 담고 있는 데이터

② 편향 X

③ 왜곡 X

데이터 크기 ↑ + 정보량 ↑

데이터
조사

데이터
가공

데이터
시각화
(그래프)

2. 피마 인디언 데이터 분석하기

〈 1950년 〉

유전 + 환경

〈 현재 〉

전체 부족

60% 당뇨

80% 비만

❗ 딥러닝 구동하려면, 속성과 클래스를 먼저 구분

2. 피마 인디언 데이터 분석하기

❗ 딥러닝 구동하려면, 속성과 클래스를 먼저 구분

속성 / 클래스

샘플	정보 1	정보 2	정보 3	…	정보 8	당뇨병 여부
1번째 인디언	6	148	72	…	50	1
2번째 인디언	1	85	66	…	31	0
3번째 인디언	8	183	64	…	32	1
…	…	…	…	…	…	…
768번째 인디언	1	93	70	…	23	0

- 샘플 수: 768
- 속성: 8
  - 정보 1 (pregnant): 과거 임신 횟수
  - 정보 2 (plasma): 포도당 부하 검사 2시간 후 공복 혈당 농도(mm Hg)
  - 정보 3 (pressure): 확장기 혈압(mm Hg)
  - 정보 4 (thickness): 삼두근 피부 주름 두께(mm)
  - 정보 5 (insulin): 혈청 인슐린(2-hour, mu U/ml)
  - 정보 6 (BMI): 체질량 지수(BMI, weight in kg/(height in $m)^2$)
  - 정보 7 (pedigree): 당뇨병 가족력
  - 정보 8 (age): 나이
- 클래스: 당뇨(1), 당뇨 아님(0)

3. pandas를 활용한 데이터 조사

CSV
(comma separated values file)

: 콤마(,)로 구분된 데이터들의 모음

header

	pregnant	plasma	pressure	thickness	insulin	BMI	pedigree	age	class
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

3. pandas를 활용한 데이터 조사

info( ) 함수

: 정보의 형식

Range Index: 768 entries, 0 to 767

Data	Columns (total 9)		
pregnant	768	non-null	int64
plasma	768	non-null	int64
pressure	768	non-null	int64
thickness	768	non-null	int64
insulin	768	non-null	int64
BMI	768	non-null	float64
pedigree	768	non-null	float64
age	768	non-null	int64
class	768	non-null	int64
Dtypes: float64(2) int64(7)			
Memory usage: 54.1 KB			

describe( ) 함수

: 샘플 수, 평균 등이 정리되어 추출

	pregnant	plasma	pressure	thickness	insulin	BMI	pedigree	age	class
count	768	768	768	768	768	768	768	768	768
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.88416	0.331329	11.760232	0.476951
min	0	0	0	0	0	0	0.078	21	0
25%	1	99	62	0	0	27.3	0.24375	24	0
50%	3	117	72	23	30.5	32	0.3725	29	0
75%	6	140.25	80	32	127.25	36.6	0.62625	41	1
max	17	199	122	99	846	67.1	2.42	81	1

3. pandas를 활용한 데이터 조사

	pregnant	class
0	6	1
1	1	0
2	8	1
3	1	0
4	0	1
5	5	0
...	...	...
765	5	0
766	1	1
767	1	0

```
print(df[['pregnant', 'class']])
```

→ 해당 열을 확인할 수 있음

4. 데이터 가공하기

```
print(df[['pregnant', 'class']].groupby(['pregnant'],
as_index=False).mean().sort_values(by='pregnant', ascending=True))
```

groupby( ) 함수 : 정보 기준의 새 그룹 생성

as_index() 함수 : 새로운 인덱스 생성 여부

mean( ) 함수 : 평균 값 추출

4. 데이터 가공하기

```
print(df[['pregnant', 'class']].groupby(['pregnant'],
 as_index=False).mean().sort_values(by='pregnant', ascending=True))
```

sort_values( ) 함수 : 컬럼을 정리

Ascending : 오름차순

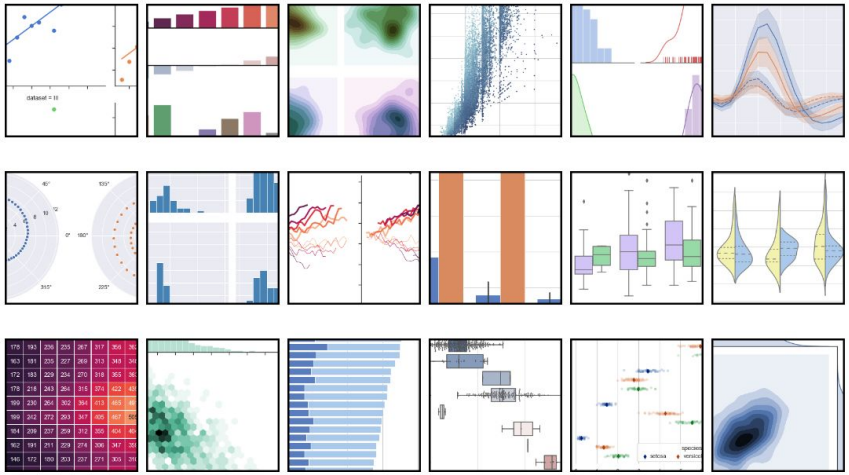
	pregnant	class
0	0	0.342342
1	1	0.214815
2	2	0.184466
3	3	0.36
4	4	0.338235
5	5	0.368421
6	6	0.32
7	7	0.555556
8	8	0.578947
9	9	0.642857
10	10	0.416667

Pre_class 해설

Q2. 다음은 데이터 조사 / 가공 / 그래프 표현과 관련된 함수이다. 각 함수의 설명으로 올바르지 않은 것을 고르시오. *

- describe()함수는 평균, 최댓값 등 정보 별 특징을 좀 더 자세히 출력하는 함수이다.
- ascending은 오름차순을 의미하며, 값이 False 일 경우 내림차순으로 정렬된다.
- heatmap은 두 항목씩 짝을 지은 뒤, 각각 어떤 패턴으로 변화하는 지를 관찰하는 함수이다.
- heatmap()에서 서로 비슷한 패턴일 수록 1에 가까운 값을, 반대로 연관이 없으면 0에 가까운 값을 출력한다.
- as_index의 값이 False면 해당 그룹을 새로운 인덱스로 지정된다.

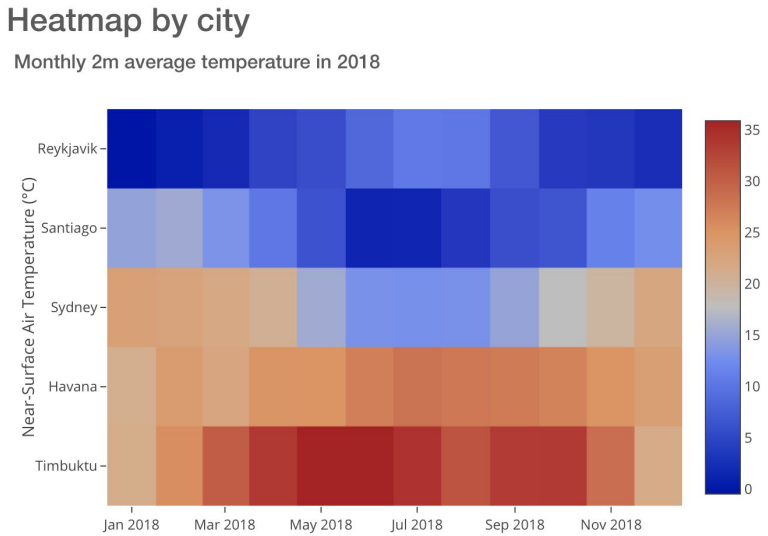
matplotlib 그래프 도시



seaborn(sns)

데이터 시각화에 사용되는 라이브러리

matplotlib 그래프 도시



Heatmap

데이터를 색깔로 변환시켜 2차원 평면에서의 분포를 나타내는 시각화 방법

matplotlib 그래프 도시

Q3. Seaborn 라이브러리에 대해 옳지 않은 것을 고르시오. * 1점

- ◯ 통상적으로 matplotlib 라이브러리와 함께 사용된다.
- ◯ 히트맵을 그리기 위해 .heatmap() 함수를 사용한다.
- ◯ .heatmap() 함수의 인자 중 vmax는 셀의 최대 밝기를 설정하는 인자이다.
- ◯ 히스토그램을 그릴 때에는 plt.hmap 함수와 함께 사용한다.
- ◯ annot인자의 값을 True로 설정하면 히트맵의 축 이름이 명시된다.

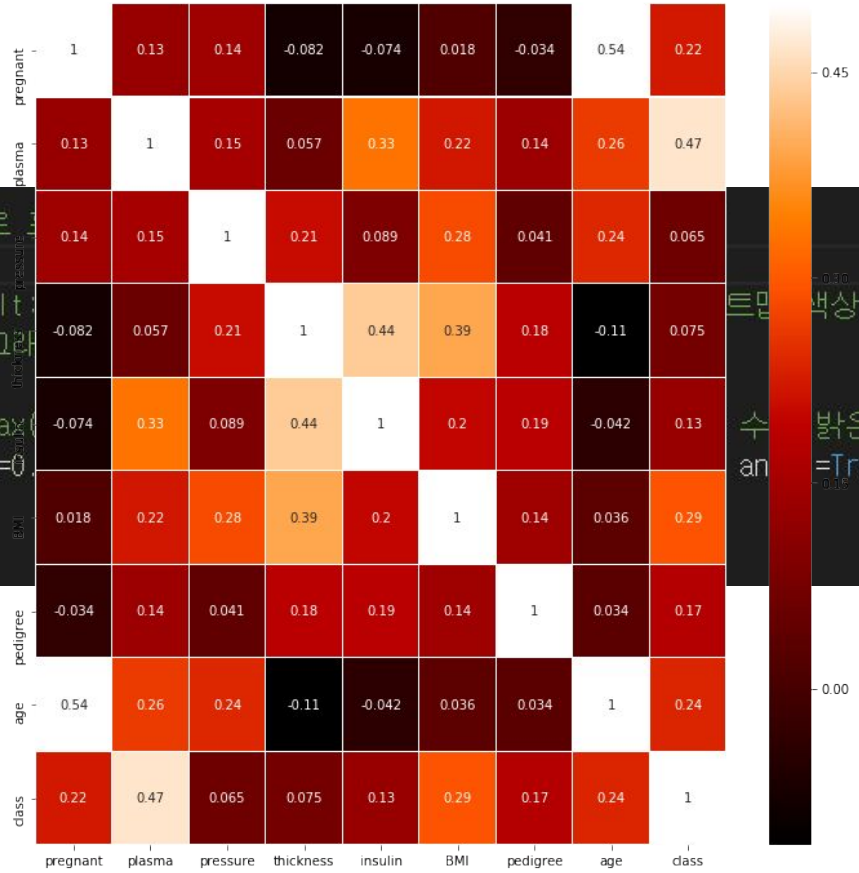
matplotlib 그래프 도시

```
# 데이터 간의 상관관계를 그래프로
colormap = plt.cm.gist_heat    #plt
plt.figure(figsize=(12,12))    #그라

# 그래프의 속성을 결정합니다. vmax(
sns.heatmap(df.corr(),linewidths=0

plt.show()
```

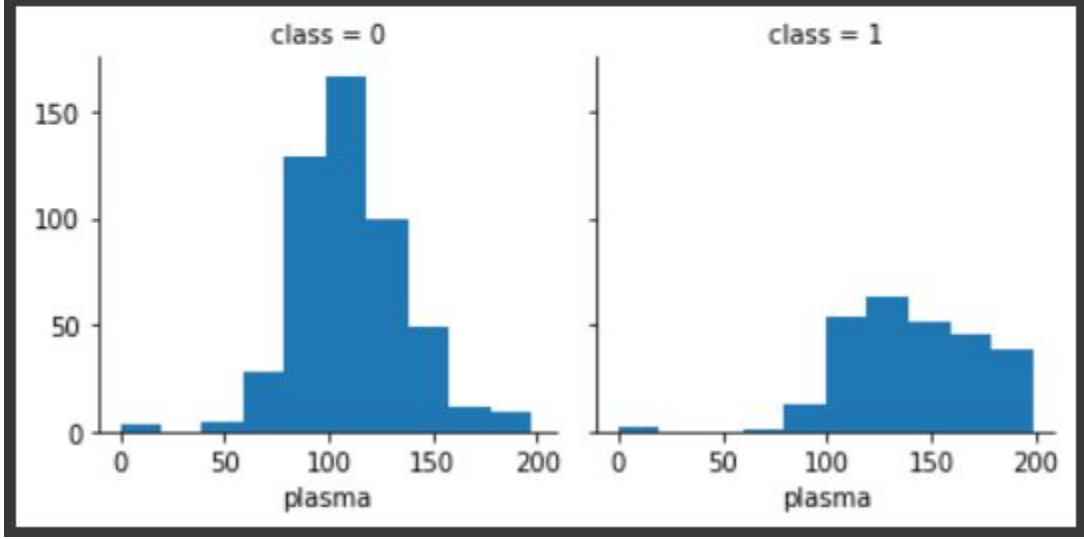
```
트맵 색상
수 밝은 색으로 표시되게 합니다.
an =True)  #annot : 셀에 값 명시 여부
```



matplotlib 그래프 도시

```
grid = sns.FacetGrid(df, col='class')     #FacetGrid : 서브 그래프
grid.map(plt.hist, 'plasma',  bins=10)    #hist : 히스토그램  | bins : 가로축 셀 개수

plt.show()
```



matplotlib 그래프 도시

```
# 딥러닝을 구동하는 데 필요한 케라스 함수를 불러옵니다.
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# 필요한 라이브러리를 불러옵니다.
import numpy as np
import tensorflow as tf

#시드값 설정; seed : random함수로 난수 생성 시 동일한 패턴의 난수배열이 생성되도록 하는 설정값
np.random.seed(3)
tf.random.set_seed(3)
```

matplotlib 그래프 도시

```
# 모델을 설정합니다.
model = Sequential()
# Dense 함수 설정 (노드 12개, 입력값 8개, 활성화함수 relu함수)
model.add(Dense(12, input_dim=8, activation='relu'))

# What is Relu?
# 렐루 함수란, 기존의 시그모이드 함수의 기울기 소실이라는 단점을 보완한 활성화 함수이다.
# 기울기 소실이란 활성화함수 개형으로 인해 경사하강법(SGD) 적용 시 갈수록 기울기의 값이 0으로 수렴하는 현상이다.
# relu 함수는 양의 값을 그대로, 음의 값을 0으로 반환하여, 기울기가 1로 일정하다.

model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

matplotlib 그래프 도시

```
# 모델을 컴파일합니다.
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
# 오차는 이항교차엔트로피 사용, 최적화는 adam 사용, 척도는 정확성 사용용

# 모델을 실행합니다.
model.fit(X, Y, epochs=200, batch_size=10)

# 결과를 출력합니다.
print("₩n Accuracy: %.4f" % (model.evaluate(X, Y)[1]))
```

감사합니다!