넷째마당
딥러닝 기본기 다지기

11장 데이터 다루기

1 딥러닝과 데이터

2 피마 인디언 데이터 분석하기

3 판다스를 활용한 데이터 조사

4 중요한 데이터 추출하기

5 피마 인디언의 당뇨병 예측 실행

1 딥러닝과 데이터

1 딥러닝과 데이터

- **딥러닝과 데이터**
  - 세월이 흐르면서 쌓인 방대한 데이터를 빅데이터라고 함
  - 이 '빅데이터'는 분명히 머신 러닝과 딥러닝으로 하여금 사람에 버금가는 판단과 지능을 가질 수 있게끔 했음
  - 데이터양이 많다고 해서 무조건 좋은 결과를 얻을 수 있는 것은 아님
  - 데이터양도 중요하지만, 그 안에 '필요한' 데이터가 얼마나 있는가도 중요하기 때문임
  - 준비된 데이터가 우리가 사용하려는 머신 러닝과 딥러닝에 얼마나 효율적으로 사용되게끔 가공되었는지 역시 중요함

1 딥러닝과 데이터

- **딥러닝과 데이터**
  - 머신 러닝 프로젝트의 성공과 실패는 얼마나 좋은 데이터를 가지고 시작하느냐에 영향을 많이 받음
  - 여기서 좋은 데이터란 한쪽으로 치우치지 않고, 불필요한 정보가 대량으로 포함되어 있지 않으며, 왜곡되지 않은 데이터를 의미
  - 이러한 데이터를 만들기 위해 머신 러닝, 딥러닝 개발자들은 데이터를 직접 들여다보고 분석할 수 있어야 함
  - 내가 이루고 싶은 목적에 맞추어 가능한 한 많은 정보를 모았다면 이를 머신 러닝과 딥러닝에서 사용 할 수 있게 잘 정제된 데이터 형식으로 바꾸어야 함
  - 이 작업은 모든 머신 러닝, 딥러닝 프로젝트의 첫 단추이자 가장 중요한 작업

1 딥러닝과 데이터

- **딥러닝과 데이터**
  - 지금부터 데이터 분석에 가장 많이 사용하는 파이썬 라이브러리인 **판다스(pandas)**와 **맷플롯립(matplotlib)** 등을 사용해 우리가 다룰 데이터가 어떤 내용을 담고 있는지 확인하면서 딥러닝의 핵심 기술들을 하나씩 구현해 보자

2 피마 인디언 데이터 분석하기

2 피마 인디언 데이터 분석하기

▼ 그림 11-1 | 피마 인디언 옛 모습

2 피마 인디언 데이터 분석하기

- **피마 인디언 데이터 분석하기**
  - 비만은 유전일까?
  - 아니면 식습관 조절에 실패한 자신의 탓일까?
  - 비만이 유전 및 환경, 모두의 탓이라는 것을 증명하는 좋은 사례가 바로 미국 남서부에 살고 있는 피마 인디언의 사례
  - 피마 인디언은 1950년대까지만 해도 비만인 사람이 단 1명도 없는 민족이었음
  - 지금은 전체 부족의 60%가 당뇨, 80%가 비만으로 고통받고 있음
  - 이는 생존하기 위해 영양분을 체내에 저장하는 뛰어난 능력을 물려받은 인디언들이 미국의 기름진 패스트푸드 문화를 만나면서 벌어진 일
  - 피마 인디언을 대상으로 당뇨병 여부를 측정한 데이터는 data 폴더에서 찾을 수 있음(`data/pima-indians-diabetes3.csv`)

2 피마 인디언 데이터 분석하기

- **피마 인디언 데이터 분석하기**
  - 이제 준비된 데이터의 내용을 들여다보자
  - 주피터 노트북을 통해 열어 보면 모두 **768명**의 인디언으로부터 여덟 개의 정보와 한 개의 클래스를 추출한 데이터임을 알 수 있음

2 피마 인디언 데이터 분석하기

▼ 그림 11-2 | 피마 인디언 데이터의 샘플, 속성, 클래스 구분

속성 (x)　　클래스 (y)

	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

2 피마 인디언 데이터 분석하기

- 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)

2 피마 인디언 데이터 분석하기

- **피마 인디언 데이터 분석하기**
  - 데이터의 각 정보가 의미하는 의학, 생리학 배경지식을 모두 알 필요는 없지만, 딥러닝을 구동하려면 반드시 속성과 클래스를 먼저 구분해야 함
  - 또한, 모델의 정확도를 향상시키기 위해서는 데이터를 추가하거나 재가공해야 할 수도 있음
  - 데이터의 내용과 구조를 파악하는 것이 중요함

3 판다스를 활용한 데이터 조사

3 판다스를 활용한 데이터 조사

- **판다스를 활용한 데이터 조사**
  - 데이터를 잘 파악하는 것이 딥러닝을 다루는 기술의 1단계라고 했음
  - 데이터의 크기가 커지고 정보량이 많아지면 데이터를 불러오고 내용을 파악할 수 있는 효과적인 방법이 필요함
  - 이때 가장 유용한 방법이 데이터를 시각화해서 눈으로 직접 확인해 보는 것
  - 지금부터 데이터를 불러와 그래프로 표현하는 방법을 알아보자
  - 데이터를 다룰 때는 데이터를 다루기 위해 만들어진 라이브러리를 사용하는 것이 좋음
  - 지금까지는 넘파이 라이브러리를 불러와 사용했는데, 넘파이의 기능을 포함하면서도 다양한 포맷의 데이터를 다루게 해 주는 판다스 라이브러리를 사용해서 데이터를 조사해 보자

3 판다스를 활용한 데이터 조사

- **판다스를 활용한 데이터 조사**
  - 이 실습에는 **판다스(pandas)**와 **시본(seaborn)** 라이브러리가 필요함
  - 코랩은 기본으로 제공하지만, 주피터 노트북을 이용해 실습 중이라면 다음 명령으로 두 라이브러리를 설치

```
!pip install pandas
!pip install seaborn
```

3 판다스를 활용한 데이터 조사

- 판다스를 활용한 데이터 조사

```
# 필요한 라이브러리를 불러옵니다.
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 깃허브에 준비된 데이터를 가져옵니다.
!git clone https://github.com/taehojo/data.git

# 피마 인디언 당뇨병 데이터셋을 불러옵니다.
df = pd.read_csv('./data/pima-indians-diabetes3.csv')
```

3 판다스를 활용한 데이터 조사

- **판다스를 활용한 데이터 조사**
  - 판다스 라이브러리의 read_csv() 함수로 csv 파일을 불러와 df라는 이름의 데이터 프레임으로 저장
  - csv란 comma separated values의 약어로, 쉼표(,)로 구분된 데이터들의 모음이란 뜻
  - csv 파일에는 데이터를 설명하는 한 줄이 파일 맨 처음에 나옴
  - 이를 **헤더**(header)라고 함

3 판다스를 활용한 데이터 조사

- **판다스를 활용한 데이터 조사**
  - 이제 불러온 데이터의 내용을 간단히 확인하고자 head() 함수를 이용해 데이터의 첫 다섯 줄을 불러오자

```
df.head(5)
```

3 판다스를 활용한 데이터 조사

- **판다스를 활용한 데이터 조사**
  - 다음과 같이 출력
  - 파이썬에서는 숫자를 0부터 세기 때문에 맨 첫 번째 행이 1이 아닌 0

	pregnant	plasma	pressure	thickness	insulin	bmi	pedigree	age	diabetes
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 판다스를 활용한 데이터 조사

- **판다스를 활용한 데이터 조사**
  - 이제 정상과 당뇨 환자가 각각 몇 명씩인지 조사해 보자
  - 불러온 데이터 프레임의 특정 칼럼을 불러오려면 df["칼럼명"]이라고 입력하면 됨
  - value_counts() 함수를 이용하면 각 컬럼의 값이 몇 개씩 있는지 알려 줌

```
df["diabetes"].value_counts()
```

3 판다스를 활용한 데이터 조사

- **판다스를 활용한 데이터 조사**
  - 다음과 같은 정보가 화면에 출력
  - 정상인 500명과 당뇨병 환자 268명을 포함, 총 768개의 샘플이 준비되어 있는 것을 알 수 있음

실행 결과

```
0    500
1    268
Name: diabetes, dtype: int64
```

3 판다스를 활용한 데이터 조사

- **판다스를 활용한 데이터 조사**
  - 정보별 특징을 좀 더 자세히 알고 싶으면 describe() 함수를 이용

```
df.describe()
```

3 판다스를 활용한 데이터 조사

- **판다스를 활용한 데이터 조사**
  - 다음과 같은 내용이 출력
  - 정보별 샘플 수(count), 평균(mean), 표준편차(std), 최솟값(min), 백분위 수로 25%, 50%, 75%에 해당하는 값 그리고 최댓값(max)이 정리되어 보임

	pregnant	plasma	pressure	thickness	insulin	bmi	pedigree	age	diabetes
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
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	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

3 판다스를 활용한 데이터 조사

- **판다스를 활용한 데이터 조사**
  - 각 항목이 어느 정도의 상관관계를 가지고 있는지 알고 싶다면 다음과 같이 입력

```
df.corr()
```

3 판다스를 활용한 데이터 조사

- 판다스를 활용한 데이터 조사
  - 다음과 같이 출력

	pregnant	plasma	pressure	thickness	insulin	bmi	pedigree	age	diabetes
pregnant	1.000000	0.129459	0.141282	−0.081672	−0.073535	0.017683	−0.033523	0.544341	0.221898
plasma	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
pressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
thickness	−0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	−0.113970	0.074752
insulin	−0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	−0.042163	0.130548
bmi	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
pedigree	−0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
age	0.544341	0.263514	0.239528	−0.113970	−0.042163	0.036242	0.033561	1.000000	0.238356
diabetes	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

X = df.iloc[:, 0:8]

y = df.iloc[:, 8]

3 판다스를 활용한 데이터 조사

- **판다스를 활용한 데이터 조사**
  - 조금 더 알아보기 쉽게 이 상관관계를 그래프로 표현해 보자
  - 맷플롯립(matplotlib)은 파이썬에서 그래프를 그릴 때 가장 많이 사용되는 라이브러리
  - 이를 기반으로 조금 더 정교한 그래프를 그리게 해 주는 시본(seaborn) 라이브러리까지 사용해서 각 정보 간 상관관계를 가시화해 보자
  - 먼저 그래프의 색상과 크기를 정함

```
colormap = plt.cm.gist_heat     # 그래프의 색상 구성을 정합니다.
plt.figure(figsize=(12,12))     # 그래프의 크기를 정합니다.
```

3 판다스를 활용한 데이터 조사

- **판다스를 활용한 데이터 조사**
  - 시본 라이브러리 중 각 항목 간 상관관계를 나타내는 heatmap() 함수를 통해 그래프를 표시해 보자
  - heatmap() 함수는 두 항목씩 짝을 지은 후 각각 어떤 패턴으로 변화하는지 관찰하는 함수
  - 두 항목이 전혀 다른 패턴으로 변화하면 0을, 서로 비슷한 패턴으로 변할수록 1에 가까운 값을 출력

```
sns.heatmap(df.corr(), linewidths=0.1, vmax=0.5, cmap=colormap,
linecolor='white', annot=True)
plt.show()
```

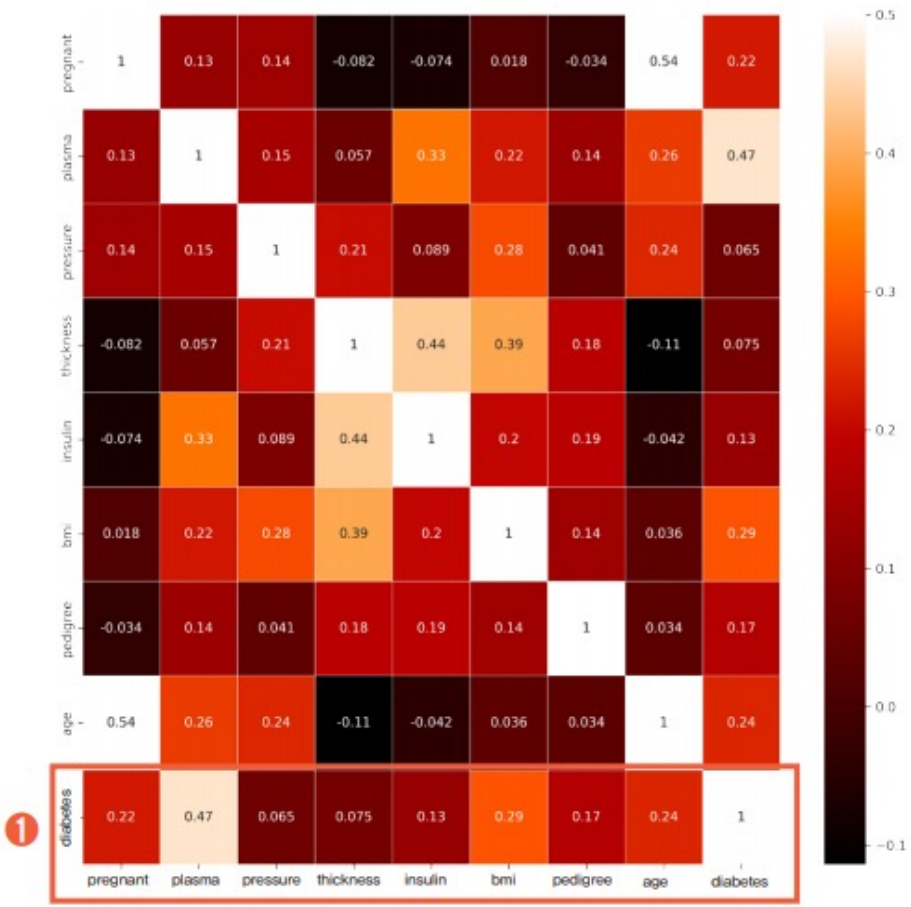
수치 표시

3 판다스를 활용한 데이터 조사

- **판다스를 활용한 데이터 조사**
  - vmax는 색상의 밝기를 조절하는 인자(vmin은 최소값)
  - cmap은 미리 정해진 맷플롯립 색상의 설정 값을 불러옴
  - 색상 설정 값은 https://matplotlib.org/users/colormaps.html에서 확인할 수 있음

3 판다스를 활용한 데이터 조사

▼ 그림 11-3 | 정보 간 상관관계 그래프



❶

3 판다스를 활용한 데이터 조사

- **판다스를 활용한 데이터 조사**
  - 그림 11-3에서 가장 눈여겨보아야 할 부분은 당뇨병 발병 여부를 가리키는 ❶ diabetes 항목
  - diabetes 항목을 보면 pregnant부터 age까지 상관도가 숫자로 표시되어 있고, 숫자가 높을수록 밝은 색상으로 채워져 있음

4 중요한 데이터 추출하기

4 중요한 데이터 추출하기

- **중요한 데이터 추출하기**
  - 앞서 그림 11-3을 살펴보면 plasma 항목(공복 혈당 농도)과 BMI(체질량 지수)가 우리가 예측하고자 하는 diabetes 항목과 상관관계가 높다는 것을 알 수 있음
  - 즉, 이 항목들이 예측 모델을 만드는 데 중요한 역할을 할 것으로 기대할 수 있음
  - 이제 이 두 항목만 따로 떼어 내어 당뇨의 발병 여부와 어떤 관계가 있는지 알아보자

4 중요한 데이터 추출하기

- **중요한 데이터 추출하기**
  - 먼저 plasma를 기준으로 각각 정상과 당뇨 여부가 어떻게 분포되는지 살펴보자
  - 다음과 같이 히스토그램을 그려 주는 맷플롯립 라이브러리의 hist() 함수를 이용

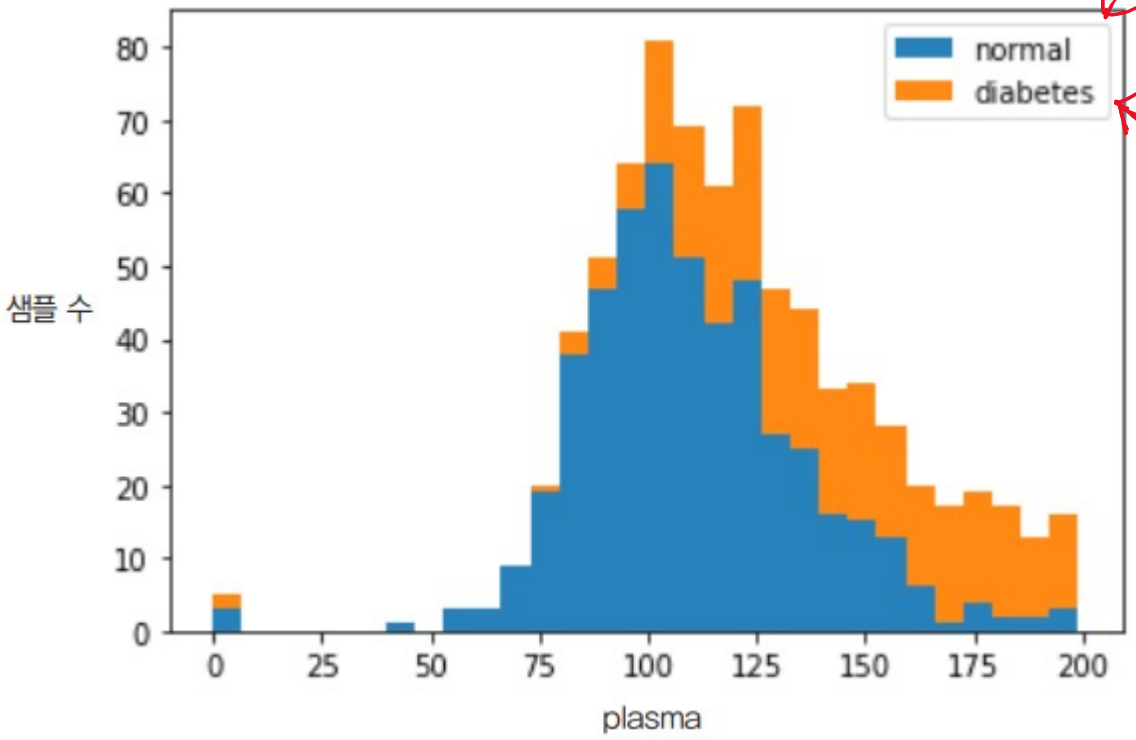
```
plt.hist(x=[df.plasma[df.diabetes==0], df.plasma[df.diabetes==1]], bins=30,
histtype='barstacked', label=['normal','diabetes'])
plt.legend()
```

4 중요한 데이터 추출하기

- **중요한 데이터 추출하기**
  - 가져오게 될 칼럼을 hist() 함수 안에 x축으로 지정
  - 여기서는 df 안의 plasma 칼럼 중 diabetes 값이 0인 것과 1인 것을 구분해 불러오게 했음
  - bins는 x축을 몇 개의 막대로 쪼개어 보여 줄 것인지 정하는 변수
  - barstacked 옵션은 여러 데이터가 쌓여 있는 형태의 막대바를 생성하는 옵션
  - 불러온 데이터의 이름을 각각 normal(정상)과 diabetes(당뇨)로 정함
  - 이를 실행시키면 그림 11-4와 같은 그래프가 형성

4 중요한 데이터 추출하기

▼ 그림 11-4 | plasma를 기준으로 정상과 당뇨 여부 표시



df.diabetes = 0

df.diabetes = 1

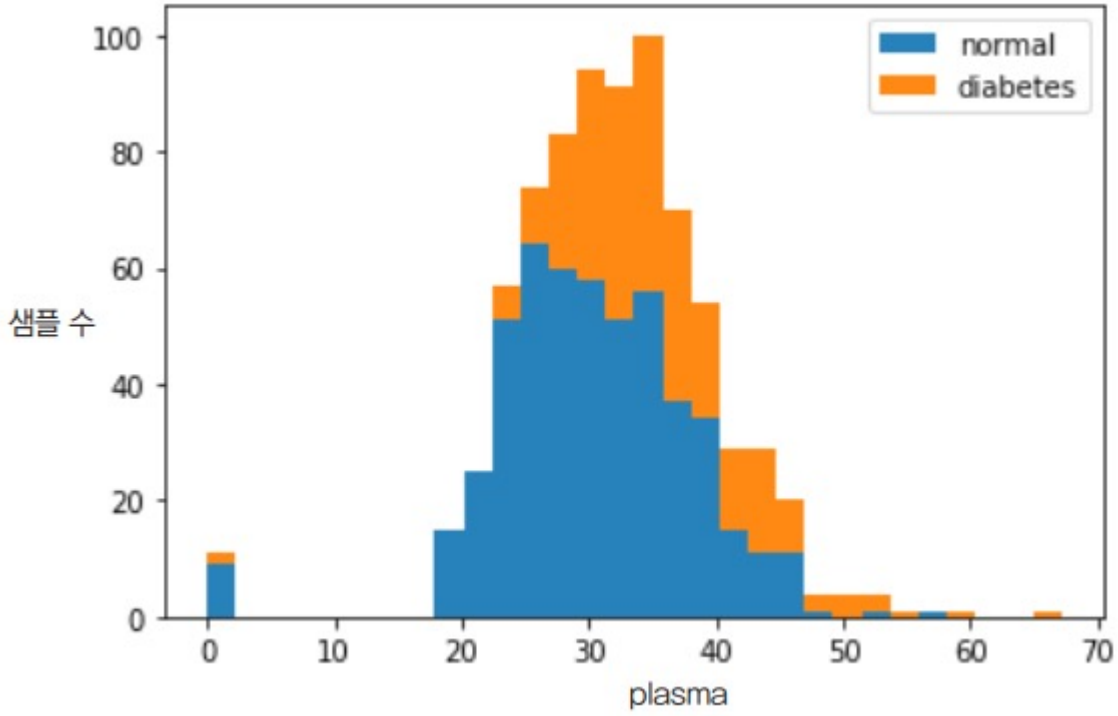
4 중요한 데이터 추출하기

- **중요한 데이터 추출하기**
  - plasma 수치가 높아질수록 당뇨인 경우가 많음을 알 수 있음
  - 마찬가지 방법으로, 이번에는 BMI를 기준으로 각각 정상과 당뇨가 어느 정도 비율로 분포하는지 살펴보자

```
plt.hist(x=[df.bmi[df.diabetes==0], df.bmi[df.diabetes==1]], bins=30,
histtype='barstacked', label=['normal','diabetes'])
plt.legend()
```

4 중요한 데이터 추출하기

▼ 그림 11-5 | BMI를 기준으로 정상과 당뇨 여부 표시



4 중요한 데이터 추출하기

- **중요한 데이터 추출하기**
  - BMI가 높아질 경우 당뇨의 발병률도 함께 증가하는 추세를 볼 수 있음
  - 이렇게 결과에 미치는 영향이 큰 항목을 발견하는 것이 데이터 전처리 과정 중 하나
  - 이 밖에도 데이터에 빠진 값이 있다면 평균이나 중앙값으로 대치하거나, 흐름에서 크게 벗어나는 이상치를 제거하는 과정 등이 데이터 전처리에 포함될 수 있음
  - SVM이나 랜덤 포레스트처럼 일반적인 머신 러닝에서는 데이터 전처리 과정이 성능 향상에 중요한 역할을 함

5 피마 인디언의 당뇨병 예측 실행

5 피마 인디언의 당뇨병 예측 실행

- **피마 인디언의 당뇨병 예측 실행**
  - 이제 텐서플로의 케라스를 이용해서 예측을 실행해 보자
  - 판다스 라이브러리를 사용하기 때문에 iloc[] 함수를 사용해 X와 y를 각각 저장
  - iloc는 데이터 프레임에서 대괄호 안에 정한 범위만큼 가져와 저장하게 함

```
# 깃허브에 준비된 데이터를 가져옵니다.
!git clone https://github.com/taehojo/data.git

# 피마 인디언 당뇨병 데이터셋을 불러옵니다.
df = pd.read_csv('./data/pima-indians-diabetes3.csv')

X = df.iloc[:,0:8]   # 세부 정보를 X로 지정합니다.
y = df.iloc[:,8]     # 당뇨병 여부를 y로 지정합니다.
```

* :iloc:
integer location

* df.:iloc[행인덱스, 열인덱스]

슬라이싱

5 피마 인디언의 당뇨병 예측 실행

- **피마 인디언의 당뇨병 예측 실행**
  - 다음과 같이 모델 구조를 설정

```
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu', name='Dense_1'))
model.add(Dense(8, activation='relu', name='Dense_2'))
model.add(Dense(1, activation='sigmoid', name='Dense_3'))
model.summary()
```

5 피마 인디언의 당뇨병 예측 실행

- **피마 인디언의 당뇨병 예측 실행**
  - 이전과 달라진 점은 은닉층이 하나 더 추가되었다는 것
  - 층과 층의 연결을 한눈에 볼 수 있게 해 주는 model.summary() 부분이 추가
  - model.summary()의 실행 결과는 다음과 같음

5 피마 인디언의 당뇨병 예측 실행

- 피마 인디언의 당뇨병 예측 실행

실행 결과

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Dense_1 (Dense)              (None, 12)                108
_________________________________________________________________
Dense_2 (Dense)              (None, 8)                 104
_________________________________________________________________
Dense_3 (Dense)              (None, 1)                 9
=================================================================
Total params: 221
Trainable params: 221
Non-trainable params: 0
```

❶ ❷ ❸ ❹

5 피마 인디언의 당뇨병 예측 실행

- **피마 인디언의 당뇨병 예측 실행**
  - ❶ Layer 부분은 층의 이름과 유형을 나타냄
  - 각 층의 이름은 자동으로 정해지는데, 따로 이름을 만들려면 Dense() 함수 안에 name='층 이름'을 추가해 주면 됨
  - 입력층과 첫 번째 은닉층을 연결해 주는 Dense_1층, 첫 번째 은닉층과 두 번째 은닉층을 연결하는 Dense_2층, 그리고 두 번째 은닉층과 출력층을 연결하는 Dense_3층이 만들어졌음을 알 수 있음

5 피마 인디언의 당뇨병 예측 실행

- **피마 인디언의 당뇨병 예측 실행**
  - ❷ Output Shape 부분은 각 층에 몇 개의 출력이 발생하는지 나타냄
  - 쉼표(,)를 사이에 두고 괄호의 앞은 행(샘플)의 수, 뒤는 열(속성)의 수를 의미
  - 행의 수는 batch_size에 정한 만큼 입력되므로 딥러닝 모델에서는 이를 특별히 세지 않음
  - 괄호의 앞은 None으로 표시
  - 여덟 개의 입력이 첫 번째 은닉층을 지나며 12개가 되고, 두 번째 은닉층을 지나며 여덟 개가 되었다가 출력층에서는 한 개의 출력을 만든다는 것을 알 수 있음

5 피마 인디언의 당뇨병 예측 실행

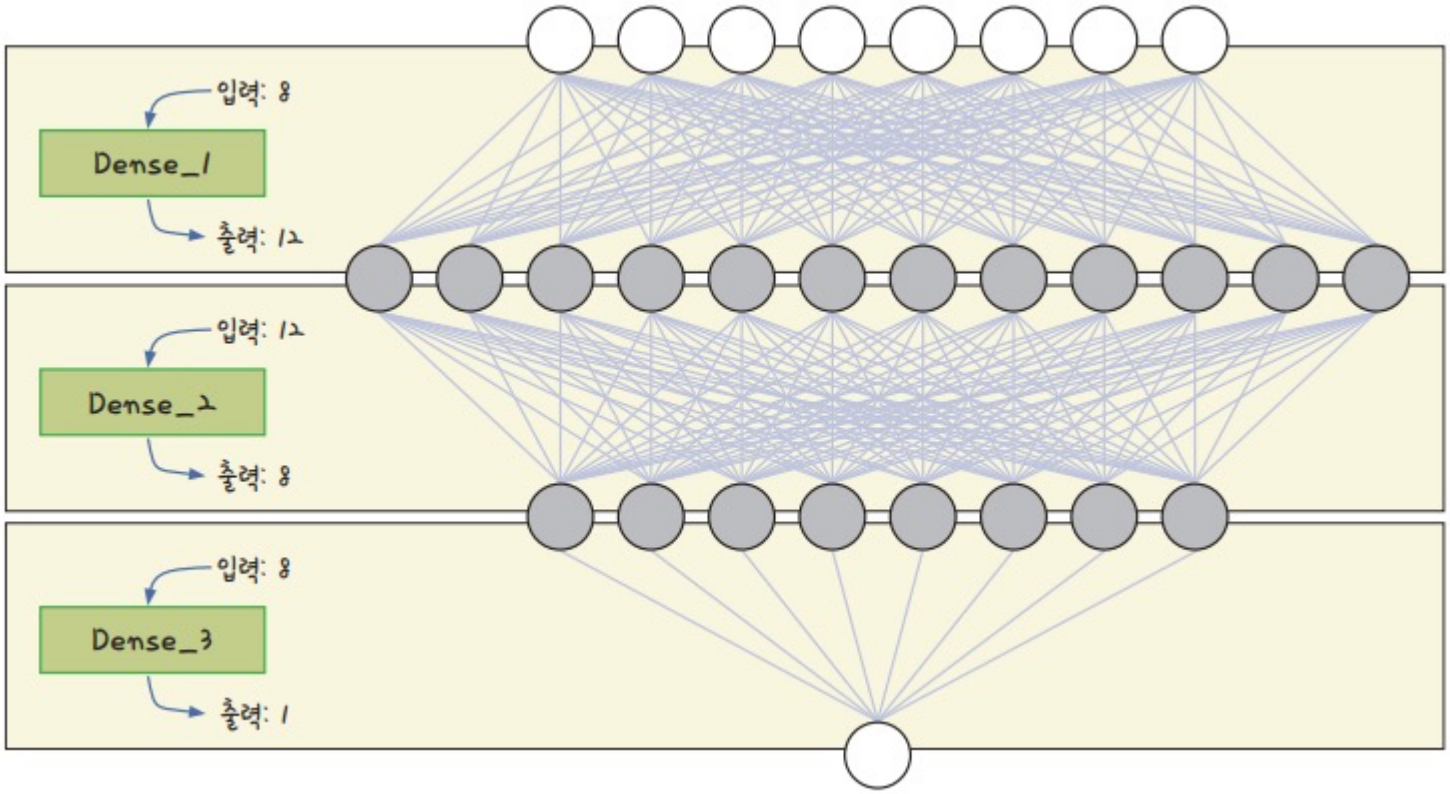
- **피마 인디언의 당뇨병 예측 실행**
  - ❸ Param 부분은 파라미터 수, 즉 총 가중치와 바이어스 수의 합을 나타냄
  - 예를 들어 첫 번째 층의 경우 입력 값 8개가 층 안에서 12개의 노드로 분산되므로 가중치가 8×12 = 96개가 되고, 각 노드에 바이어스가 한 개씩 있으니 전체 파라미터 수는 96 + 12 = 108이 됨

5 피마 인디언의 당뇨병 예측 실행

- **피마 인디언의 당뇨병 예측 실행**
  - ❹부분은 전체 파라미터를 합산한 값
  - Trainable params는 학습을 진행하면서 업데이트가 된 파라미터들이고, Non-trainable params는 업데이트가 되지 않은 파라미터 수를 나타냄

5 피마 인디언의 당뇨병 예측 실행

▼ 그림 11-6 | 피마 인디언 당뇨병 예측 모델의 구조



5 피마 인디언의 당뇨병 예측 실행

- **피마 인디언의 당뇨병 예측 실행**

  은닉층의 개수를 왜 두 개로 했나요? 그리고 노드의 수는 왜 각각 12개와 8개로 했나요?

  - 입력과 출력의 수는 정해져 있지만, 은닉층은 몇 층으로 할지, 은닉층 안의 노드는 몇 개로 할지에 대한 정답은 없음
  - **자신의 프로젝트에 따라 설정해야 함 : Hyper parameter**
  - 여러 번 반복하면서 최적 값을 찾아내는 것이 좋으며, 여기서는 임의의 수로 12와 8을 설정했고 설명의 편의성을 위해 두 개의 은닉층을 만들었음
  - 여러분이 직접 노드의 수와 은닉층의 개수를 바꾸어 보면서 더 좋은 정확도가 나오는지 시험해 보자

5 피마 인디언의 당뇨병 예측 실행

- 피마 인디언의 당뇨병 예측 실행
  - 전체 코드는 다음과 같음

실습 | 피마 인디언의 당뇨병 예측하기

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# pandas 라이브러리를 불러옵니다.
import pandas as pd

# 깃허브에 준비된 데이터를 가져옵니다.
!git clone https://github.com/taehojo/data.git

# 피마 인디언 당뇨병 데이터셋을 불러옵니다.
df = pd.read_csv('./data/pima-indians-diabetes3.csv')
```

5 피마 인디언의 당뇨병 예측 실행

- 피마 인디언의 당뇨병 예측 실행

```
X = df.iloc[:,0:8]   # 세부 정보를 X로 지정합니다.
y = df.iloc[:,8]     # 당뇨병 여부를 y로 지정합니다.

# 모델을 설정합니다.
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu', name='Dense_1'))
model.add(Dense(8, activation='relu', name='Dense_2'))
model.add(Dense(1, activation='sigmoid', name='Dense_3'))
model.summary()
```

5 피마 인디언의 당뇨병 예측 실행

- 피마 인디언의 당뇨병 예측 실행

```
# 모델을 컴파일합니다.
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])

# 모델을 실행합니다.
history = model.fit(X, y, epochs=100, batch_size=5)
```

5 피마 인디언의 당뇨병 예측 실행

- 피마 인디언의 당뇨병 예측 실행

실행 결과

```
Epoch 1/100
154/154 [==============================] - 2s 2ms/step - loss: 1.7955 -
accuracy: 0.5430
... (중략) ...
Epoch 100/100
154/154 [==============================] - 0s 2ms/step - loss: 0.5705 -
accuracy: 0.7161
```

- 100번을 반복한 현재, 약 71.61%의 정확도를 보이고 있음