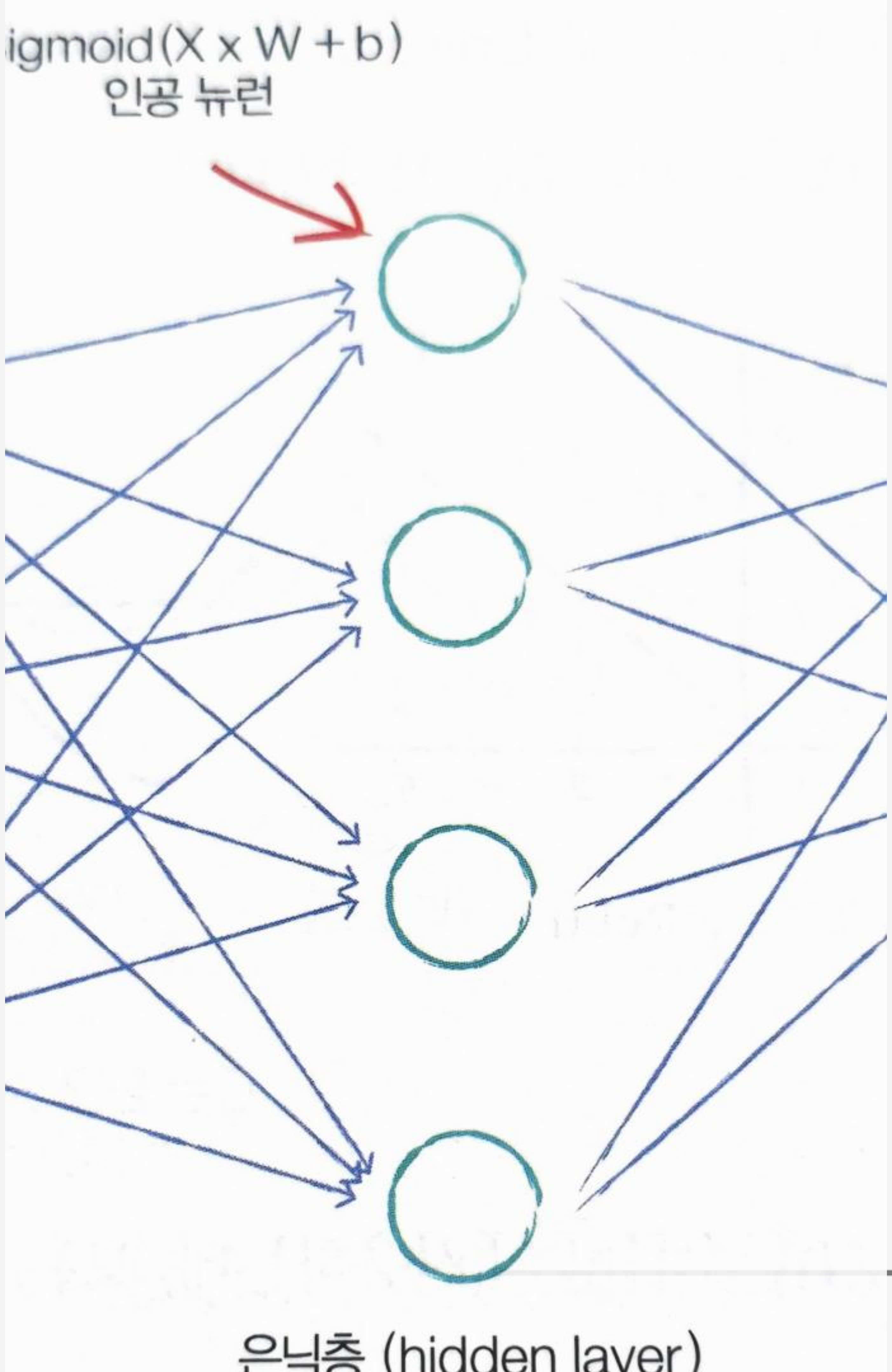
S.C.P 장혜선 2021/08/03

+

모델 설계하기



목차

01
생존율 예측
- 폐암 수술 환자의 생존율을 예측하는 모델 구현

02
당뇨병 예측
- 피마 인디언 당뇨병을 예측하는 모델 구현

01

생존율 예측

폐암 수술 환자의 생존율을 예측하는 모델 구현

“

수술하기 전
수술 후의 생존율을
예측할 방법이 없을까?

" 예측?

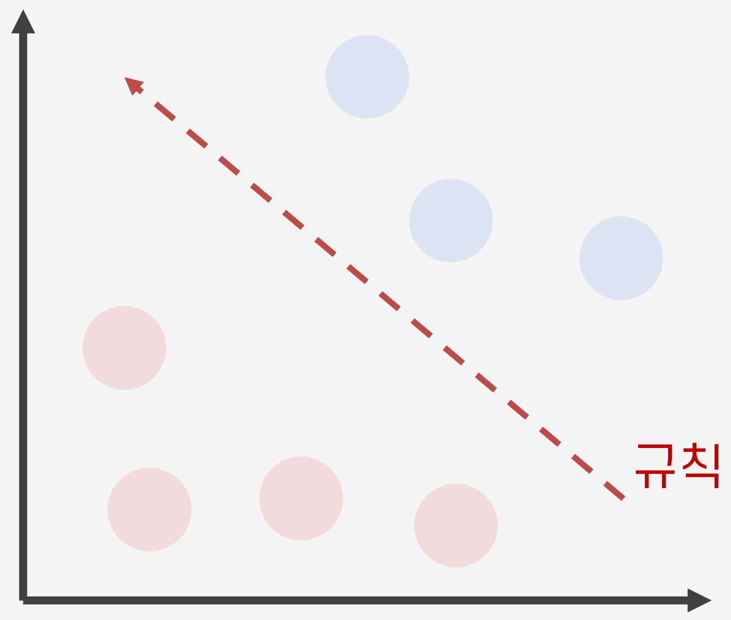
환자 데이터 입력

진료 기록 1 : 생존

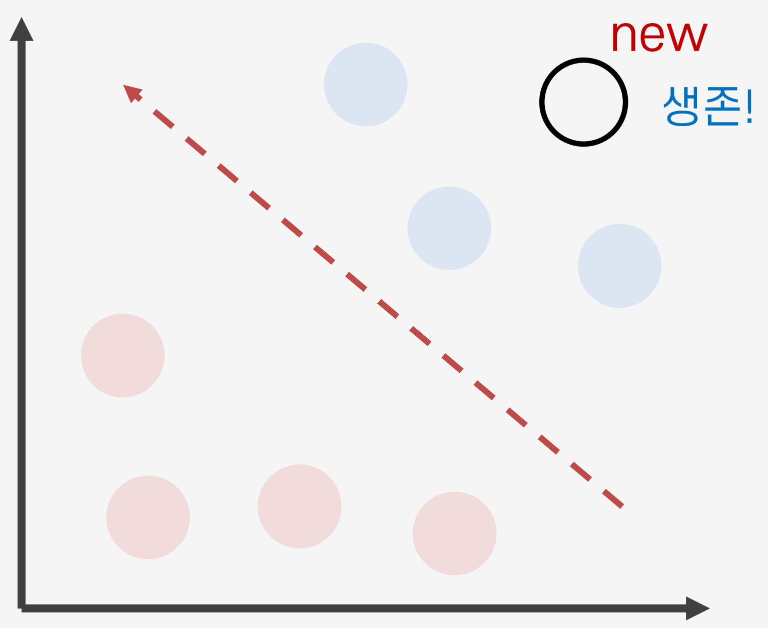
진료 기록 2 : 사망

진료 기록 3 : 생존

학습

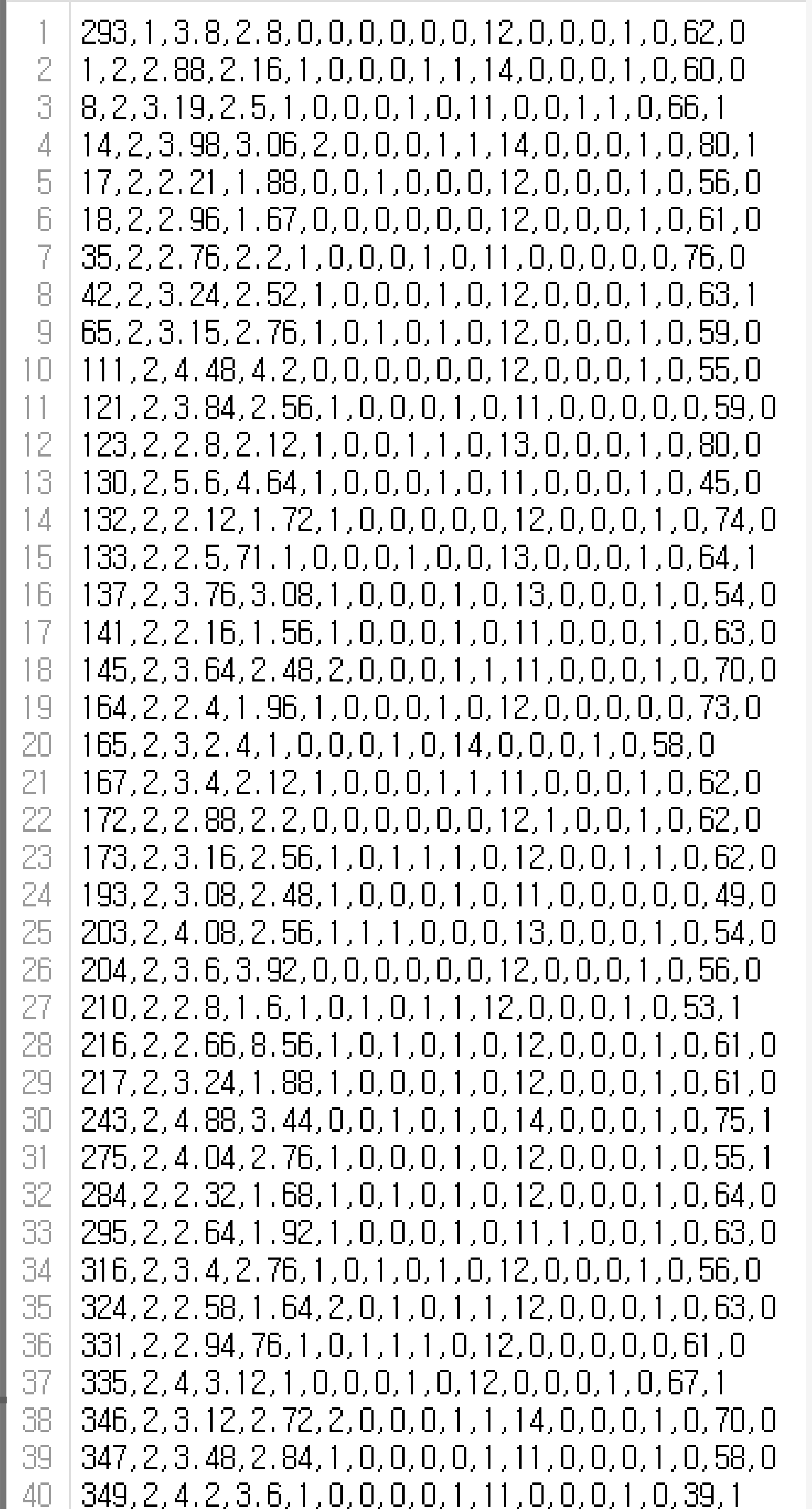


예측



기존 환자 데이터

- 수술 전 진단 데이터와 수술 후 생존 결과를 기록한 실제 의료 기록 데이터
- 18개 항목으로 구분
- 1~17까지 속성, 18번째 항목은 클래스



```
293,1,3.8,2.8,0,0,0,0,0,0,12,0,0,0,1,0,62,0
1,2,2.88,2.16,1,0,0,0,1,1,14,0,0,0,1,0,60,0
8,2,3.19,2.5,1,0,0,0,1,0,11,0,0,1,1,0,66,1
14,2,3.98,3.06,2,0,0,0,1,1,14,0,0,0,1,0,80,1
17,2,2.21,1.88,0,0,1,0,0,0,12,0,0,0,1,0,56,0
18,2,2.96,1.67,0,0,0,0,0,0,12,0,0,0,1,0,61,0
35,2,2.76,2.2,1,0,0,0,1,0,11,0,0,0,0,0,76,0
42,2,3.24,2.52,1,0,0,0,1,0,12,0,0,0,1,0,63,1
65,2,3.15,2.76,1,0,1,0,1,0,12,0,0,0,1,0,59,0
111,2,4.48,4.2,0,0,0,0,0,0,12,0,0,0,1,0,55,0
121,2,3.84,2.56,1,0,0,0,1,0,11,0,0,0,0,0,59,0
123,2,2.8,2.12,1,0,0,1,1,0,13,0,0,0,1,0,80,0
130,2,5.6,4.64,1,0,0,0,1,0,11,0,0,0,1,0,45,0
132,2,2.12,1.72,1,0,0,0,0,0,12,0,0,0,1,0,74,0
133,2,2.5,71.1,0,0,0,1,0,0,13,0,0,0,1,0,64,1
137,2,3.76,3.08,1,0,0,0,1,0,13,0,0,0,1,0,54,0
141,2,2.16,1.56,1,0,0,0,1,0,11,0,0,0,1,0,63,0
145,2,3.64,2.48,2,0,0,0,1,1,11,0,0,0,1,0,70,0
164,2,2.4,1.96,1,0,0,0,1,0,12,0,0,0,0,0,73,0
165,2,3,2.4,1,0,0,0,1,0,14,0,0,0,1,0,58,0
167,2,3.4,2.12,1,0,0,0,1,1,11,0,0,0,1,0,62,0
172,2,2.88,2.2,0,0,0,0,0,0,12,1,0,0,1,0,62,0
173,2,3.16,2.56,1,0,1,1,1,0,12,0,0,1,1,0,62,0
193,2,3.08,2.48,1,0,0,0,1,0,11,0,0,0,0,0,49,0
203,2,4.08,2.56,1,1,1,0,0,0,13,0,0,0,1,0,54,0
204,2,3.6,3.92,0,0,0,0,0,0,12,0,0,0,1,0,56,0
210,2,2.8,1.6,1,0,1,0,1,1,12,0,0,0,1,0,53,1
216,2,2.66,8.56,1,0,1,0,1,0,12,0,0,0,1,0,61,0
217,2,3.24,1.88,1,0,0,0,1,0,12,0,0,0,1,0,61,0
243,2,4.88,3.44,0,0,1,0,1,0,14,0,0,0,1,0,75,1
275,2,4.04,2.76,1,0,0,0,1,0,12,0,0,0,1,0,55,1
284,2,2.32,1.68,1,0,1,0,1,0,12,0,0,0,1,0,64,0
295,2,2.64,1.92,1,0,0,0,1,0,11,1,0,0,1,0,63,0
316,2,3.4,2.76,1,0,1,0,1,0,12,0,0,0,1,0,56,0
324,2,2.58,1.64,2,0,1,0,1,1,12,0,0,0,1,0,63,0
331,2,2.94,76,1,0,1,1,1,0,12,0,0,0,0,0,61,0
335,2,4,3.12,1,0,0,0,1,0,12,0,0,0,1,0,67,1
346,2,3.12,2.72,2,0,0,0,1,1,14,0,0,0,1,0,70,0
347,2,3.48,2.84,1,0,0,0,0,1,11,0,0,0,1,0,58,0
349,2,4.2,3.6,1,0,0,0,0,1,11,0,0,0,1,0,39,1
```

+

“ 구현

- 데이터
- 모델 설정
- 실행
- 출력

```
Data_set = numpy.loadtxt("dataset/ThoraricSurgery.csv", delimiter=",")

X = Data_set[:,0:17] #속성
Y = Data_set[:,17]   #클래스

model = Sequential()
model.add(Dense(30, input_dim=17, activation='relu'))  #은닉층
model.add(Dense(1, activation='sigmoid')) #출력층

model.compile(loss='mean_squared_error', optimizer='adam',metrics=['accuracy'])
model.fit(X, Y, epochs=30, batch_size=10)

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

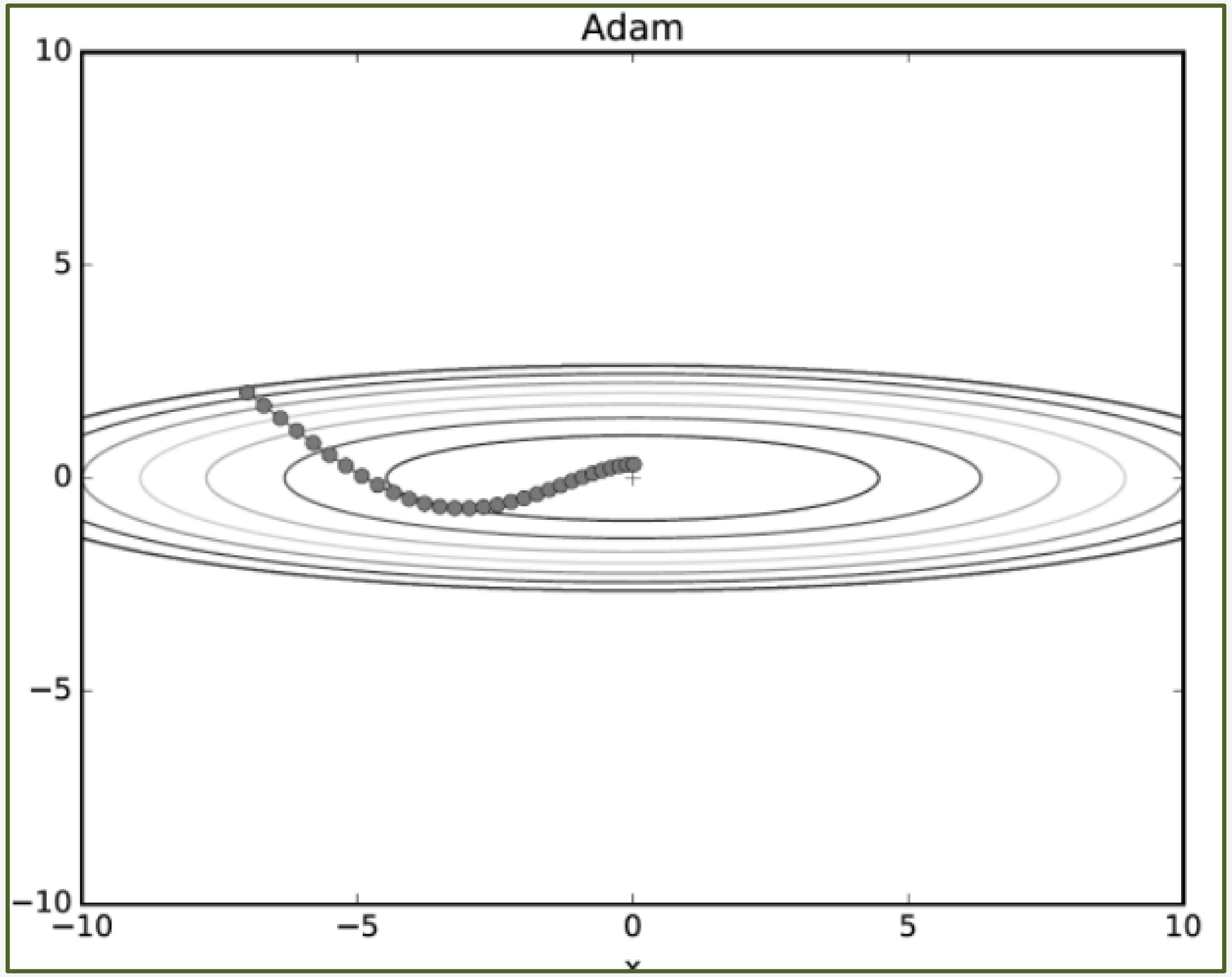
```
Accuracy: 0.8553
```

“ 아담?

- 잘 모르면 Adam

- 진동과 폭을 줄이는 모멘텀
  +
- 학습률을 조정해나가는 알엠에스프롭

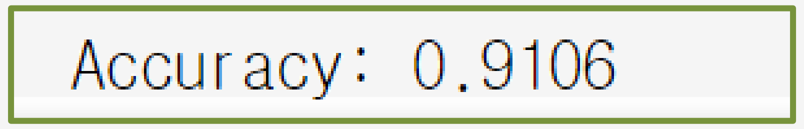
- Adam = 방향 + 스텝사이즈



"정확도 높이기

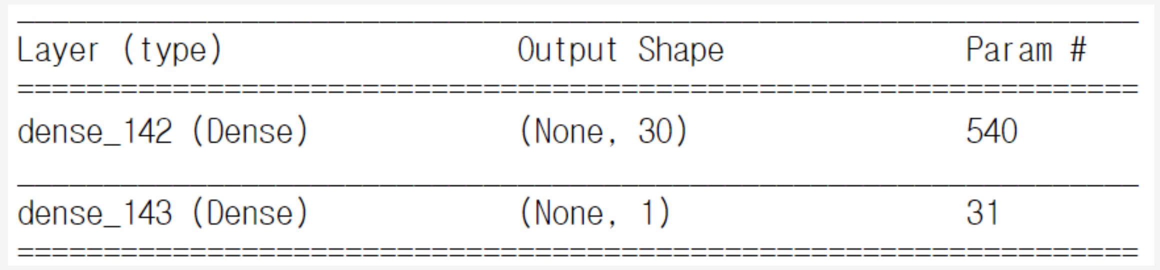
01. 학습 횟수

epochs=1500, batch_size=20

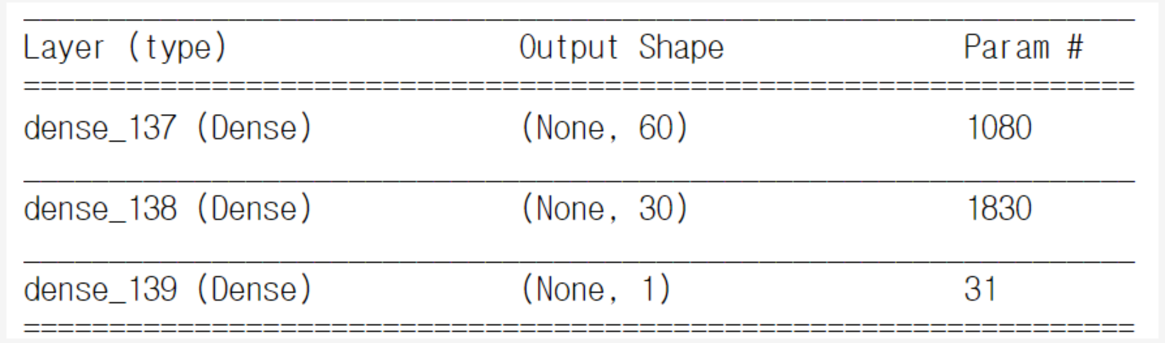


```
Accuracy: 0.9106
```

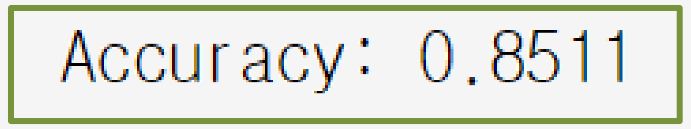
02. 층



```
Layer (type)                 Output Shape              Param #
=================================================================
dense_142 (Dense)            (None, 30)                540
_________________________________________________________________
dense_143 (Dense)            (None, 1)                 31
=================================================================
```



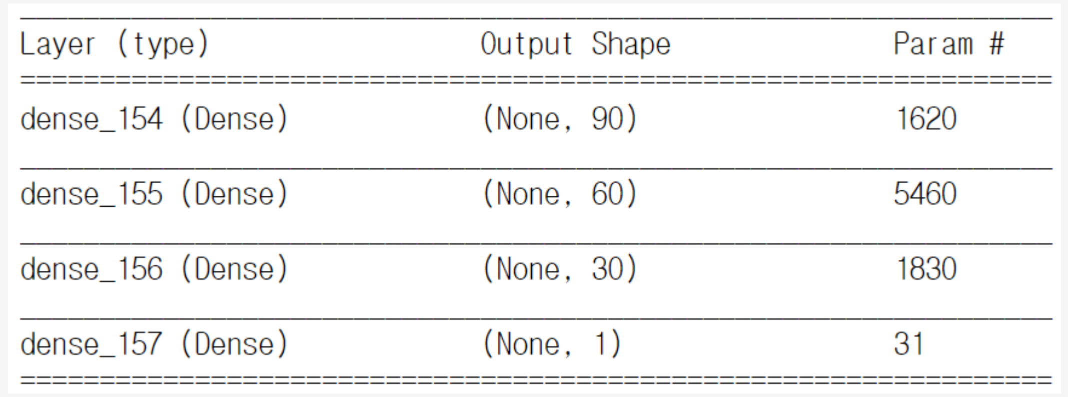
```
Layer (type)                 Output Shape              Param #
=================================================================
dense_137 (Dense)            (None, 60)                1080
_________________________________________________________________
dense_138 (Dense)            (None, 30)                1830
_________________________________________________________________
dense_139 (Dense)            (None, 1)                 31
=================================================================
```



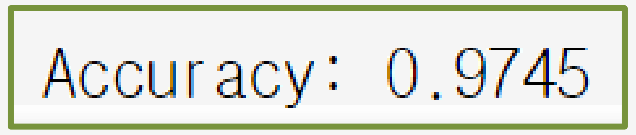
```
Accuracy: 0.8511
```

03. 둘 다

epochs=2000, batch_size=20



```
Layer (type)                 Output Shape              Param #
=================================================================
dense_154 (Dense)            (None, 90)                1620
_________________________________________________________________
dense_155 (Dense)            (None, 60)                5460
_________________________________________________________________
dense_156 (Dense)            (None, 30)                1830
_________________________________________________________________
dense_157 (Dense)            (None, 1)                 31
=================================================================
```



```
Accuracy: 0.9745
```

+

02

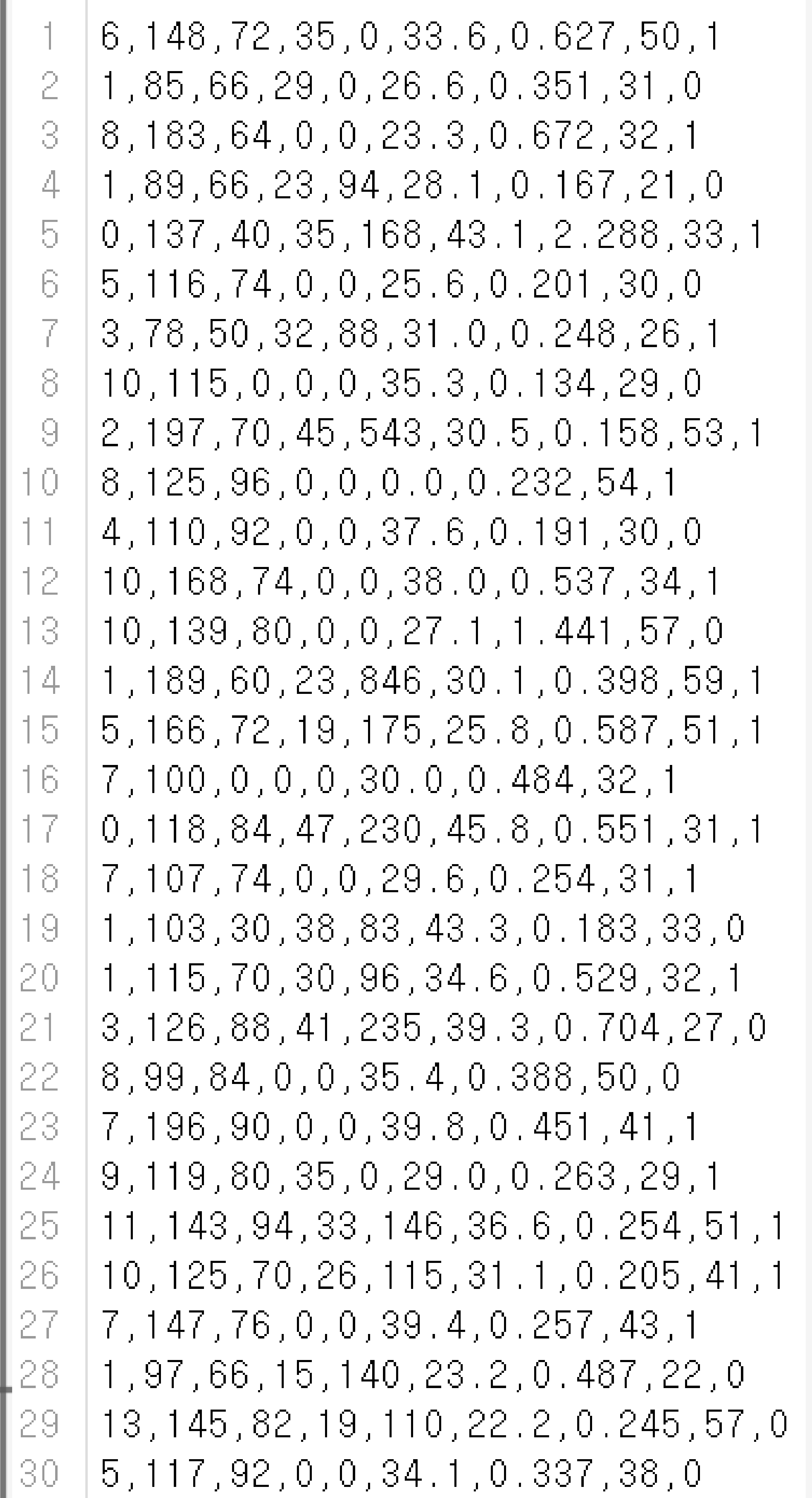
당뇨병 예측

피마 인디언 당뇨병을 예측하는 모델 구현

피마 인디언 데이터

- 768명/8개의 속성/1개의 클래스

1	2	3	4	5	6	7	8
임신 횟수	공복 혈당 농도	혈압	삼두근 피부 주름 두께	혈청 인슐린	체질량 지수	당뇨병 가족력	나이



```
6,148,72,35,0,33.6,0.627,50,1
1,85,66,29,0,26.6,0.351,31,0
8,183,64,0,0,23.3,0.672,32,1
1,89,66,23,94,28.1,0.167,21,0
0,137,40,35,168,43.1,2.288,33,1
5,116,74,0,0,25.6,0.201,30,0
3,78,50,32,88,31.0,0.248,26,1
10,115,0,0,0,35.3,0.134,29,0
2,197,70,45,543,30.5,0.158,53,1
8,125,96,0,0,0.0,0.232,54,1
4,110,92,0,0,37.6,0.191,30,0
10,168,74,0,0,38.0,0.537,34,1
10,139,80,0,0,27.1,1.441,57,0
1,189,60,23,846,30.1,0.398,59,1
5,166,72,19,175,25.8,0.587,51,1
7,100,0,0,0,30.0,0.484,32,1
0,118,84,47,230,45.8,0.551,31,1
7,107,74,0,0,29.6,0.254,31,1
1,103,30,38,83,43.3,0.183,33,0
1,115,70,30,96,34.6,0.529,32,1
3,126,88,41,235,39.3,0.704,27,0
8,99,84,0,0,35.4,0.388,50,0
7,196,90,0,0,39.8,0.451,41,1
9,119,80,35,0,29.0,0.263,29,1
11,143,94,33,146,36.6,0.254,51,1
10,125,70,26,115,31.1,0.205,41,1
7,147,76,0,0,39.4,0.257,43,1
1,97,66,15,140,23.2,0.487,22,0
13,145,82,19,110,22.2,0.245,57,0
5,117,92,0,0,34.1,0.337,38,0
```

구현

✓ 데이터

✓ 모델 설정

✓ 실행

✓ 출력

```
Accuracy: 0.9805
```

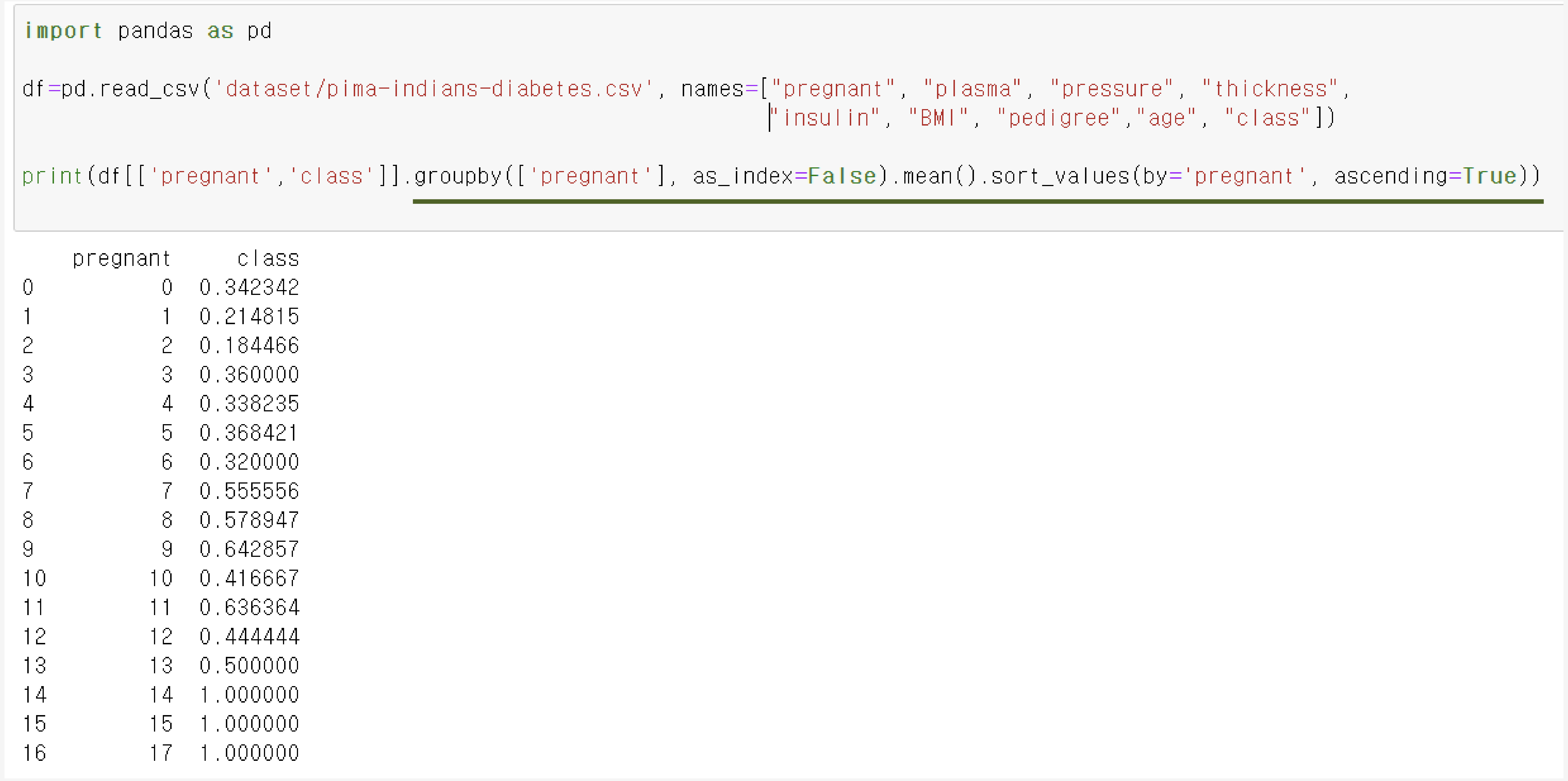
```
from keras.models import Sequential
from keras.layers import Dense
import numpy
import tensorflow as tf


seed = 0
numpy.random.seed(seed)
tf.random.set_seed(seed)


dataset = numpy.loadtxt("dataset/pima-indians-diabetes.csv", delimiter=",")
X = dataset[:,0:8]
Y = dataset[:,8]


model = Sequential()
model.add(Dense(36,input_dim=8, activation='relu'))
model.add(Dense(24, activation='relu'))
model.add(Dense(12, activation='relu'))
model.add(Dense(1, activation='sigmoid'))


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


model.fit(X, Y, epochs=3000, batch_size=20)


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

“

정보마다
발병률에
얼마큼 영향을 미칠까?

+

임신횟수와 당뇨병 발병률

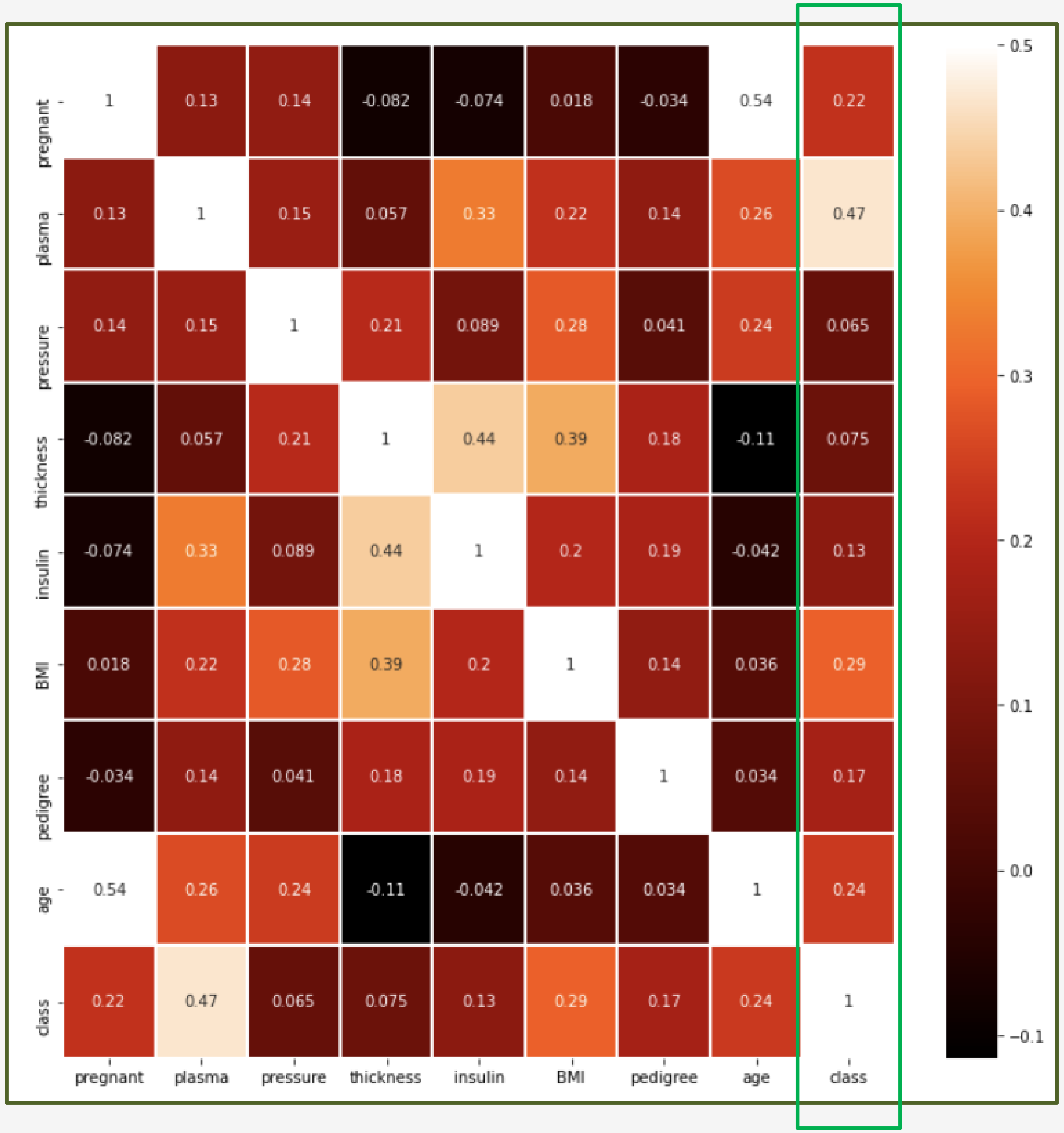

```
import pandas as pd

df=pd.read_csv('dataset/pima-indians-diabetes.csv', names=["pregnant", "plasma", "pressure", "thickness",
                                                            "insulin", "BMI", "pedigree","age", "class"])

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

```
    pregnant     class
0          0  0.342342
1          1  0.214815
2          2  0.184466
3          3  0.360000
4          4  0.338235
5          5  0.368421
6          6  0.320000
7          7  0.555556
8          8  0.578947
9          9  0.642857
10        10  0.416667
11        11  0.636364
12        12  0.444444
13        13  0.500000
14        14  1.000000
15        15  1.000000
16        17  1.000000
```

상관관계 확인하기

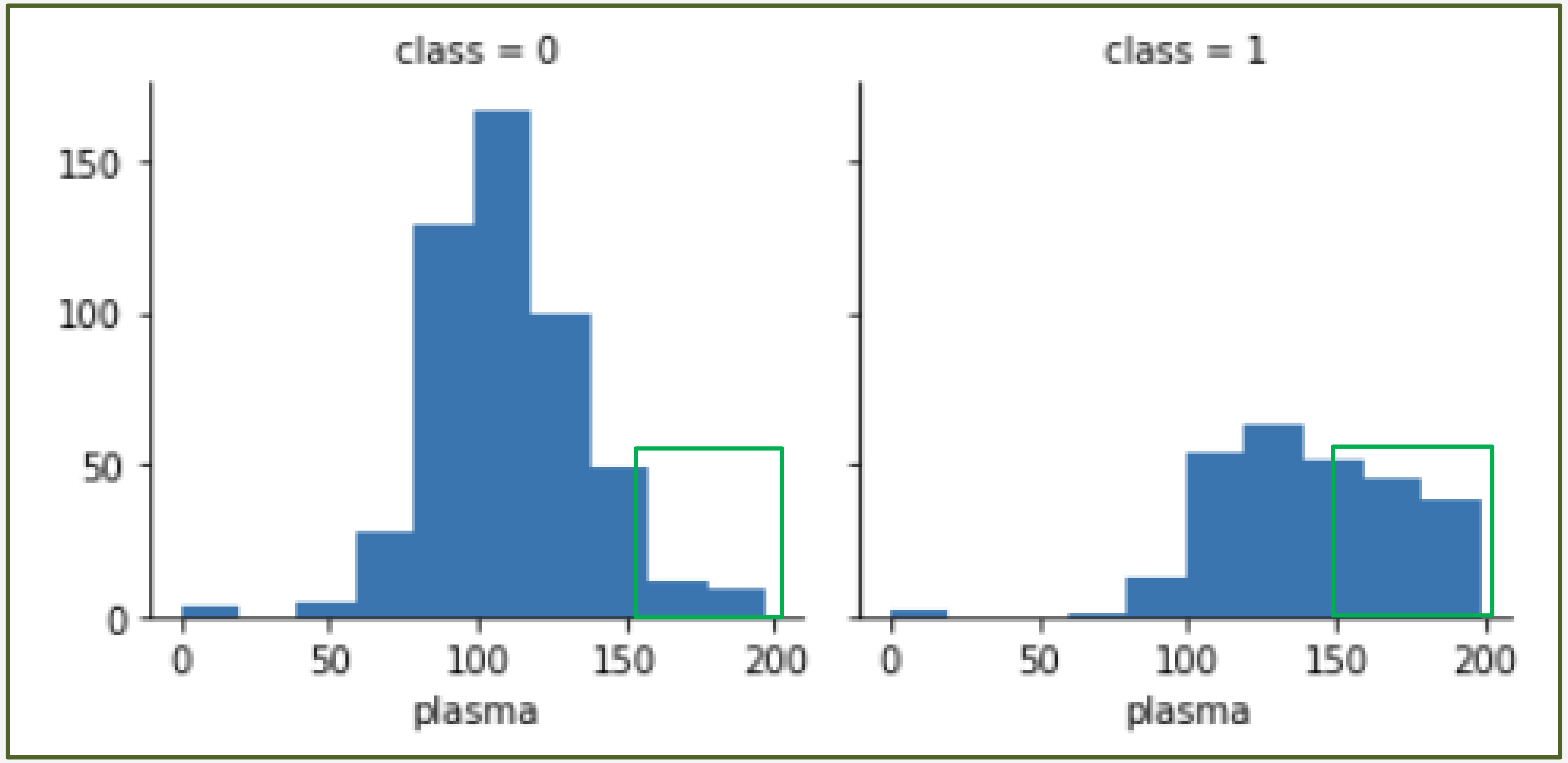
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

df=pd.read_csv('dataset/pima-indians-diabetes.csv', names=["pregnant", "plasma", "pressure", "thickness",
                                                          "insulin", "BMI", "pedigree","age", "class"])

plt.figure(figsize = (12, 12))
sns.heatmap(df.corr(), linewidths=0.1, vmax=0.5, cmap=plt.cm.gist_heat, linecolor='white', annot=True)

plt.show()
```

상관관계 확인하기



	pregnant	plasma	pressure	thickness	insulin	BMI	pedigree	age	class
pregnant	1	0.13	0.14	-0.082	-0.074	0.018	-0.034	0.54	0.22
plasma	0.13	1	0.15	0.057	0.33	0.22	0.14	0.26	0.47
pressure	0.14	0.15	1	0.21	0.089	0.28	0.041	0.24	0.065
thickness	-0.082	0.057	0.21	1	0.44	0.39	0.18	-0.11	0.075
insulin	-0.074	0.33	0.089	0.44	1	0.2	0.19	-0.042	0.13
BMI	0.018	0.22	0.28	0.39	0.2	1	0.14	0.036	0.29
pedigree	-0.034	0.14	0.041	0.18	0.19	0.14	1	0.034	0.17
age	0.54	0.26	0.24	-0.11	-0.042	0.036	0.034	1	0.24
class	0.22	0.47	0.065	0.075	0.13	0.29	0.17	0.24	1

상관관계 확인하기



```
Accuracy: 0.7461
```

+

발표를 들어주셔서 감사합니다 :)