```
!wget -O diabetes.csv https://raw.githubusercontent.com/plotly/datasets/master/diabetes.csv
```

```
--2024-05-09 02:53:40--  https://raw.githubusercontent.com/plotly/datasets/master/diabetes.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 23873 (23K) [text/plain]
Saving to: ‘diabetes.csv’

diabetes.csv        100%[===================>]  23.31K  --.-KB/s    in 0.007s

2024-05-09 02:53:40 (3.34 MB/s) - ‘diabetes.csv’ saved [23873/23873]
```

```
!ls
!pwd
```

```
diabetes.csv  sample_data
/content
```

```
!tail diabetes.csv
```

```
1,106,76,0,0,37.5,0.197,26,0
6,190,92,0,0,35.5,0.278,66,1
2,88,58,26,16,28.4,0.766,22,0
9,170,74,31,0,44,0.403,43,1
9,89,62,0,0,22.5,0.142,33,0
10,101,76,48,180,32.9,0.171,63,0
2,122,70,27,0,36.8,0.34,27,0
5,121,72,23,112,26.2,0.245,30,0
1,126,60,0,0,30.1,0.349,47,1
1,93,70,31,0,30.4,0.315,23,0
```

```
from torch import nn, optim, from_numpy
import numpy as np
from numpy import genfromtxt

xy = genfromtxt('/content/diabetes.csv', delimiter=',', dtype=np.float32)
x_data = from_numpy(xy[:, 0:-1])
y_data = from_numpy(xy[:, [-1]])
print(f'X\'s shape: {x_data.shape} | Y\'s shape: {y_data.shape}')
```

```
X's shape: torch.Size([769, 8]) | Y's shape: torch.Size([769, 1])
```

```
class Model(nn.Module):
    def __init__(self):
        """
        In the constructor we instantiate two nn.Linear module
        """
        super(Model, self).__init__()
        self.l1 = nn.Linear(8, 10)
        self.l2 = nn.Linear(10, 4)
        self.l3 = nn.Linear(4, 20)
        self.l4 = nn.Linear(20, 1)

        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        """
        In the forward function we accept a Variable of input data and we must return
        a Variable of output data. We can use Modules defined in the constructor as
        well as arbitrary operators on Variables.
        """
        out1 = self.sigmoid(self.l1(x))
        out2 = self.sigmoid(self.l2(out1))
        out3 = self.sigmoid(self.l3(out2))
        y_pred = self.sigmoid(self.l4(out3))
        return y_pred

model = Model()

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.1)
```

```
for epoch in range(200):
    y_pred = model(x_data)

    loss = criterion(y_pred, y_data)
    print(f'Epoch: {epoch + 1}/200 | Loss: {loss.item():.4f}')

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

```
Epoch: 1/200 | Loss: nan
Epoch: 2/200 | Loss: nan
Epoch: 3/200 | Loss: nan
Epoch: 4/200 | Loss: nan
Epoch: 5/200 | Loss: nan
Epoch: 6/200 | Loss: nan
Epoch: 7/200 | Loss: nan
Epoch: 8/200 | Loss: nan
Epoch: 9/200 | Loss: nan
Epoch: 10/200 | Loss: nan
Epoch: 11/200 | Loss: nan
Epoch: 12/200 | Loss: nan
Epoch: 13/200 | Loss: nan
Epoch: 14/200 | Loss: nan
Epoch: 15/200 | Loss: nan
Epoch: 16/200 | Loss: nan
Epoch: 17/200 | Loss: nan
Epoch: 18/200 | Loss: nan
Epoch: 19/200 | Loss: nan
Epoch: 20/200 | Loss: nan
Epoch: 21/200 | Loss: nan
Epoch: 22/200 | Loss: nan
Epoch: 23/200 | Loss: nan
Epoch: 24/200 | Loss: nan
Epoch: 25/200 | Loss: nan
Epoch: 26/200 | Loss: nan
Epoch: 27/200 | Loss: nan
Epoch: 28/200 | Loss: nan
Epoch: 29/200 | Loss: nan
Epoch: 30/200 | Loss: nan
Epoch: 31/200 | Loss: nan
Epoch: 32/200 | Loss: nan
Epoch: 33/200 | Loss: nan
Epoch: 34/200 | Loss: nan
Epoch: 35/200 | Loss: nan
Epoch: 36/200 | Loss: nan
Epoch: 37/200 | Loss: nan
Epoch: 38/200 | Loss: nan
Epoch: 39/200 | Loss: nan
Epoch: 40/200 | Loss: nan
Epoch: 41/200 | Loss: nan
Epoch: 42/200 | Loss: nan
Epoch: 43/200 | Loss: nan
Epoch: 44/200 | Loss: nan
Epoch: 45/200 | Loss: nan
Epoch: 46/200 | Loss: nan
Epoch: 47/200 | Loss: nan
Epoch: 48/200 | Loss: nan
Epoch: 49/200 | Loss: nan
Epoch: 50/200 | Loss: nan
Epoch: 51/200 | Loss: nan
Epoch: 52/200 | Loss: nan
Epoch: 53/200 | Loss: nan
Epoch: 54/200 | Loss: nan
Epoch: 55/200 | Loss: nan
Epoch: 56/200 | Loss: nan
Epoch: 57/200 | Loss: nan
Epoch: 58/200 | Loss: nan
```