```
import numpy as np
```

```
class Perceptron:
  def __init__(self,n,neta=0.1):
    self.w=np.random.randn(n+1)/np.sqrt(n)
    self.neta=neta
  def step(self,w_sum):
    if w_sum>0:
      return 1
    else:
      return 0
  def fit(self,X,Y,epochs=5):
    X=np.c_[X,np.ones(X.shape[0])]
    for epoch in range(epochs):
      for(x,t) in zip(X,Y):
        o=self.step(np.dot(x,self.w))
        if t!=o:
          error = t-o
          self.w  += self.neta  *  error  *  x
  def predict(self,  X,  addBias  = True):
    X =  np.atleast_2d(X)
    if addBias:
      X=np.c_[X,  np.ones(X.shape[0])]
    return  self.step(np.dot(X,  self.w))
```

```
X = np.array([[0,0], [0,1], [1,0], [1,1]])
y = np.array([[0], [0], [0], [1]])
p_model_and = Perceptron(X.shape[1], neta = 0.01)
p_model_and.fit(X,y,epochs = 50)
```

```
p_model_and.w
```

```
array([-0.71616305,  0.55546309,  0.16290421])
```

```
for (x,  t)  in  zip(X,  y):
  pred  =  p_model_and.predict(x)
  print(f"Data: {x},  Target:   {t},  predicted:    {pred}")
```

```
Data:   [0 0],  Target: [0],  predicted:        1
Data:   [0 1],  Target: [0],  predicted:        1
Data:   [1 0],  Target: [0],  predicted:        0
Data:   [1 1],  Target: [1],  predicted:        1
```

```
X   = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y   = np.array([[0],    [1],    [1],    [1]])
p_model_or  = Perceptron(X.shape[1],  neta  =   0.1)
p_model_or.fit(X,  y,  epochs=  100)
```

```
for (x,  t)  in  zip(X,  y):
  pred  =  p_model_or.predict(x)
  print(f"Data: {x},  Target:   {t},  predicted:    {pred}")
```

```
Data:   [0 0],  Target: [0],  predicted:        0
Data:   [0 1],  Target: [1],  predicted:        1
Data:   [1 0],  Target: [1],  predicted:        1
Data:   [1 1],  Target: [1],  predicted:        1
```

```
X = np.array([[0,   0], [0, 1], [1, 0], [1, 1]])
Y = np.array([[0],  [1],    [1],    [0]])
p_model_xor = Perceptron(X.shape[1],  neta  =   0.1)
p_model_xor.fit(X,  y,  epochs= 50)
```

```
for (x,  t)  in  zip(X,  y):
  pred  =  p_model_xor.predict(x)
  print(f"Data: {x},  Target:   {t},  predicted:    {pred}")
```

```
Data:   [0 0],  Target: [0],  predicted:        0
Data:   [0 1],  Target: [1],  predicted:        1
Data:   [1 0],  Target: [1],  predicted:        1
Data:   [1 1],  Target: [1],  predicted:        1
```

```
from sklearn.linear_model import  Perceptron
from sklearn.datasets  import  load_digits
X,  y   = load_digits(return_X_y    =  True)
p = Perceptron()
```

```
p.fit(X,  y)
print(p.score(X,  y))
```

```
0.9393433500278241
```

```
x   = np.arange(36).reshape(-1,     9)
x
```

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8],
       [ 9, 10, 11, 12, 13, 14, 15, 16, 17],
       [18, 19, 20, 21, 22, 23, 24, 25, 26],
       [27, 28, 29, 30, 31, 32, 33, 34, 35]])
```

```
x[0]
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8])
```

```
x[0].shape
```

```
(9,)
```

```
x.shape
```

```
(4, 9)
```

```
x.shape[0]
```

```
4
```

```
name=["Vivek",  "Yash",  "Aditya","Tejesh"]
roll_no=[   365,    362,    361,    382]
mapped=zip(name,roll_no)
print(set(mapped))
```

```
{('Vivek', 365), ('Tejesh', 382), ('Yash', 362), ('Aditya', 361)}
```

```
in_num  =   10
print   ("Input number in_num",in_num)
out_arr =  np.atleast_2d(in_num)
print   ("output 2d array from input number:",out_arr)
```

```
Input number in_num 10
output 2d array from input number: [[10]]
```