PROBLEMA DI CLASSIFICAZIONE

Generazione del dataset

```
import numpy as np
from sklearn import datasets
X, t = datasets.make_blobs(n_samples=150,n_features=2,
                           centers=2,cluster_std=1.5,
                           random_state=6)
```

2 Array:
X -> features
t –> target

n_samples -> numero di sample
n_features -> numero di feature
centers -> numero di cluster / classi
cluster_std -> deviazione standard dei cluster
random_state -> inizializzazione della generazione di numeri pseudo-casuali

PROBLEMA DI CLASSIFICAZIONE

Visualizzazione del dataset

```python
import matplotlib.pyplot as plt

def plot_data(a,b):
  #Plotting
  fig = plt.figure(figsize=(6,4))
  plt.plot(a[:, 0][b == 0], a[:, 1][b == 0], 'r^')
  plt.plot(a[:, 0][b == 1], a[:, 1][b == 1], 'bs')
  plt.xlabel("feature 1")
  plt.ylabel("feature 2")
  plt.title('Random Classification Data with 2 classes')
  plt.show()
```

CLASSE PERCEPTRON
Metodi

Aggiungere il bias all'input

```
#Add bias to the input
def add_bias(self, x):
  # input -> input for the ANN
  if x.ndim > 1:
    inp = np.insert(x,self.n,1,axis=1)
  else:
    inp = np.insert(x,2,1)
  return inp
# END of function
```

CLASSE PERCEPTRON
Metodi

Attivazione del neurone

```
#Calculates the activation of the neuron
def activation(self,x):
  net_input = np.dot(x,self.weights)
  output = self.step_func(net_input)
  return output
```

CLASSE PERCEPTRON
Metodi

Funzione di attivazione

```
#defining the activation function
def step_func(self, x):
  if (x > 0):
    return 1.0
  else:
    return 0.0
#END of function
```

CLASSE PERCEPTRON
Metodi

Metodo per l'addestramento della rete neurale

```
# Training the ANN
def train(self):

  for epoch in range(self.epochs):

      # variable to store misclassified example
      n_miss = 0

      # looping for every example.
      for idx, i in enumerate(self.inputANN):

          # reshape the array
          i = i.reshape(1,self.n+1)
          # Calculating prediction - outpurt of the ANN
          output = self.activation(i)

          if (t[idx] - output) != 0:
              # Updating if the example is misclassified
              n_miss += 1
              #delta rule
              self.weights += self.lr*((t[idx] - output)*i.T)
          # end of if

      print(epoch, n_miss)  #end of epoch

  return self.weights
  #END of function
```

CLASSE PERCEPTRON
Metodi

Metodo per attivare la rete neurale per una sola predizione

```
#Activate the ANN
def predict(self, x):
  x = self.add_bias(x)
  output = self.activation(x)
  return output
```

- Aggiungiamo il bias al patter di input
- Attiviamo la rete neurale

CLASSE PERCEPTRON

Utilizziamo la classe Percetron per fare apprendere il pattern dei dati che abbiamo generato

```
#create the instance of the class Perceptron

learning_rate = 0.5
epochs = 10

perc = Perceptron(X, t, learning_rate, epochs)

perc.train()

t_pred = np.empty(0)
for x in X:
  val = perc.predict(x)
  t_pred = np.append(t_pred, val)
```