

Ecco l'elenco delle funzioni di perdita con un piccolo script di esempio per ognuna.

Mean Squared Error (MSE)

La **Mean Squared Error (MSE)** è una funzione di perdita comunemente usata nei problemi di regressione. Calcola la media del quadrato delle differenze tra i valori predetti e i valori reali. In altre parole, misura quanto sono diverse le previsioni del modello rispetto ai valori effettivi. L'obiettivo è minimizzare questa differenza durante l'addestramento del modello.

```
main.py ×
main.py > ...
1  import torch
2
3  y_true = torch.tensor([1.0, 2.0, 3.0]) # Valori reali
4  y_pred = torch.tensor([1.5, 2.5, 3.5]) # Valori predetti
5
6  mse_loss = torch.nn.MSELoss()
7  loss = mse_loss(y_pred, y_true)
8
9  print("MSE Loss:", loss.item())
10 |
```

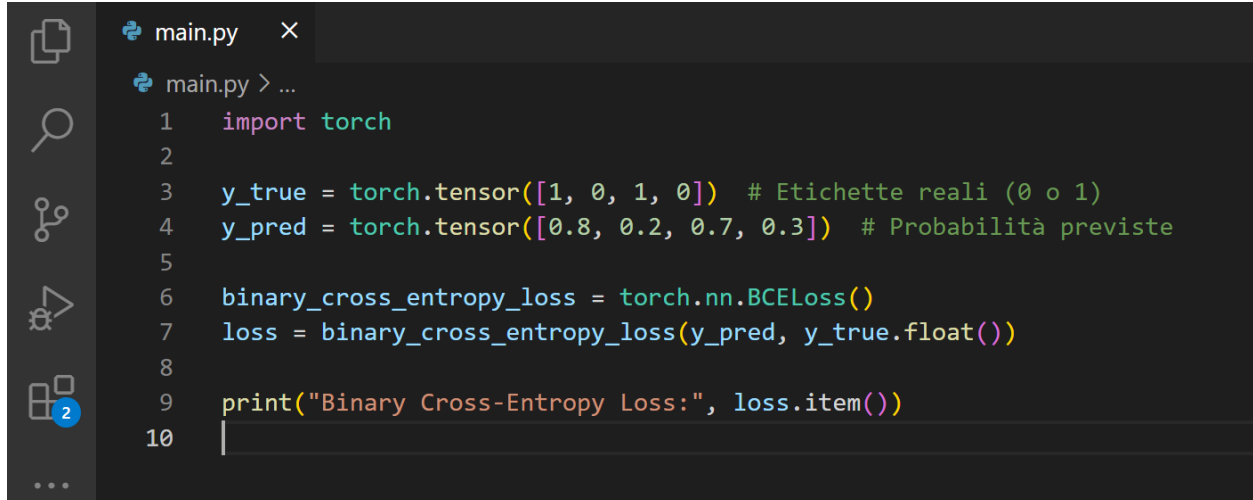
Cross-Entropy Loss

La **Cross-Entropy Loss** è una funzione di perdita ampiamente utilizzata nei problemi di classificazione. Misura la differenza tra le probabilità previste dal modello e le etichette reali. In un problema di classificazione, il modello prevede la probabilità di appartenenza a ciascuna classe per ogni esempio. La Cross-Entropy Loss confronta queste probabilità con le etichette reali e calcola una penalità per le previsioni errate. L'obiettivo è minimizzare questa penalità durante l'addestramento del modello.

```
main.py ×
main.py > ...
1  import torch
2
3  y_true = torch.tensor([2, 0, 1]) # Indici delle classi reali
4  y_pred = torch.tensor([[0.1, 0.2, 0.7],
5                        [0.8, 0.1, 0.1],
6                        [0.2, 0.6, 0.2]]) # Probabilità previste per ciascuna classe
7
8  cross_entropy_loss = torch.nn.CrossEntropyLoss()
9  loss = cross_entropy_loss(y_pred, y_true)
10
11 print("Cross-Entropy Loss:", loss.item())
12
```

Binary Cross-Entropy Loss

La **Binary Cross-Entropy Loss** è una variante della Cross-Entropy Loss utilizzata nei problemi di classificazione binaria, dove ci sono solo due classi possibili (etichetta 0 o 1). Misura la differenza tra le probabilità previste dal modello e le etichette reali per ciascun esempio. La Binary Cross-Entropy Loss è particolarmente adatta quando l'output del modello è una probabilità compressa tra 0 e 1.



```
main.py ×  
main.py > ...  
1  import torch  
2  
3  y_true = torch.tensor([1, 0, 1, 0]) # Etichette reali (0 o 1)  
4  y_pred = torch.tensor([0.8, 0.2, 0.7, 0.3]) # Probabilità previste  
5  
6  binary_cross_entropy_loss = torch.nn.BCELoss()  
7  loss = binary_cross_entropy_loss(y_pred, y_true.float())  
8  
9  print("Binary Cross-Entropy Loss:", loss.item())  
10
```