- Modificare il metodo *__init__* dello script «*metrics.py*» dell'esercitazione A.

Esercitazione

Esercitazione

- Il nuovo metodo avrà la seguente firma:

```
class Metrics():

    def __init__(self, classes : list[str], real_y: np.array, pred_y: np.array) -> None: ...
```

- Al metodo:
  - Non sarà più fornito *num_data*.
  - Saranno forniti direttamente *real_y* e *pred_y*.
  - Questi ultimi saranno assegnati alle proprie variabili di classe.

Esercitazione

- Modificare poi, il metodo *test* del codice di **classificazione** fornito a lezione e presente nello script «*net_runner.py*».

Esercitazione

- Nel metodo:
  - Rimuovere tutto il codice legato al calcolo dell'accuratezza totale e per classe.
  - Riempire due liste, *real_y* e *pred_y* con etichette e predizioni di test.

Esercitazione

- Fatto questo:
  - Creare un oggetto ***Metrics***.
  - Fornirgli: le classi, *real_y* e *pred_y*.
  - Mostrare un *report* della classificazione.
  - Restituire *accuracy* dal metodo *test*.

Esercitazione

- Il nuovo metodo somiglierà a:

```
    def test(self, testloader : torch.utils.data.DataLoader, use_current_net: bool = False, preview : bool = False):

        if use_current_net: ···
        else: ···

        # La rete entra in modalita' inferenza.
        net.eval()

        real_y = []
        pred_y = []

        # Non e' necessario calcolare i gradienti al passaggio dei dati in rete.
        with torch.no_grad(): ···

        mt = Metrics(self.classes, real_y, pred_y)
        mt.report()

        return mt.accuracy()
```

Esercitazione

- Al completamento, si sarà riusciti ad integrare l’oggetto di calcolo delle metriche con la rete di classificazione:
  - Eseguendo un addestramento o un test, sarà possibile vedere un report continuo delle metriche.
  - Testare entrambi i casi.

Esercitazione