Esercitazione

- Scaricare da ‘*Virtuale*’ la risorsa Esercitazione:
  - Dentro sarà possibile trovarvi lo script python «*metrics.py*».

- Il contenuto sarà il seguente:

metrics.py > ...

```python
import sys
import torch
import numpy as np

from sklearn.metrics import ConfusionMatrixDisplay

▶ Launch TensorBoard Session
from torch.utils.tensorboard import SummaryWriter

torch.manual_seed(42)
np.random.seed(42)


class Metrics():

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

    # Calcola la matrice di confusione fra le classi.
    def compute_confusion_matrix(self) -> None: ···

    # Calcola l'accuracy sulla confusion matrix.
    def accuracy(self) -> float: ···

    # Calcola la recall per la classe indicata.
    def recall(self, class_id: int) -> float: ···

    # Calcola la precision per la classe indicata.
    def precision(self, class_id: int) -> float: ···

    # Calcola l'f1-score per la classe indicata.
    def f1_score(self, class_id: int) -> float: ···

    # Calcola il numero di campioni veri per la classe indicata.
    def support(self, class_id: int) -> int: ···

    # Mostra un report di tutte le informazioni.
    def report(self) -> None: ···

    # Verifica se l'indice classe e' valido.
    def __valid_class_id(self, class_id: int) -> bool: ···


if __name__ == '__main__':

    num_data = 20
    classes = ['circle', 'square', 'triangle']

    real_y = np.random.randint(0, len(classes), num_data)
    pred_y = np.random.randint(0, len(classes), num_data)

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

Esercitazione

A.A. 23/24

UNIEC

Esercitazione

- Aggiungere alla classe *Metrics* il metodo *get_confusion_matrix_figure* il quale dovrà:
  - Verificare se la confusion matrix è già stata calcolata.
  - Se non è stata calcolata, calcolarla.
  - Utilizzare *ConfusionMatrixDisplay* della libreria *scikit-learn* per ottenere un display della matrice.
  - Ottenere dal display un plot e assegnarlo ad una variabile.
  - Restituire la ‘figure’ del plot.

- Completato il metodo, aggiornare *__main__* nel seguente modo:

```
140  if __name__ == '__main__':
141
142      num_data = 20
143      classes = ['circle', 'square', 'triangle']
144
145      real_y = np.random.randint(0, len(classes), num_data)
146      pred_y = np.random.randint(0, len(classes), num_data)
147
148      mt = Metrics(classes, real_y, pred_y)
149      mt.report()
150
151      writer = SummaryWriter()
152
153      [illegible]
154      [illegible]
155
156      writer.flush()
157      writer.close()
158
```

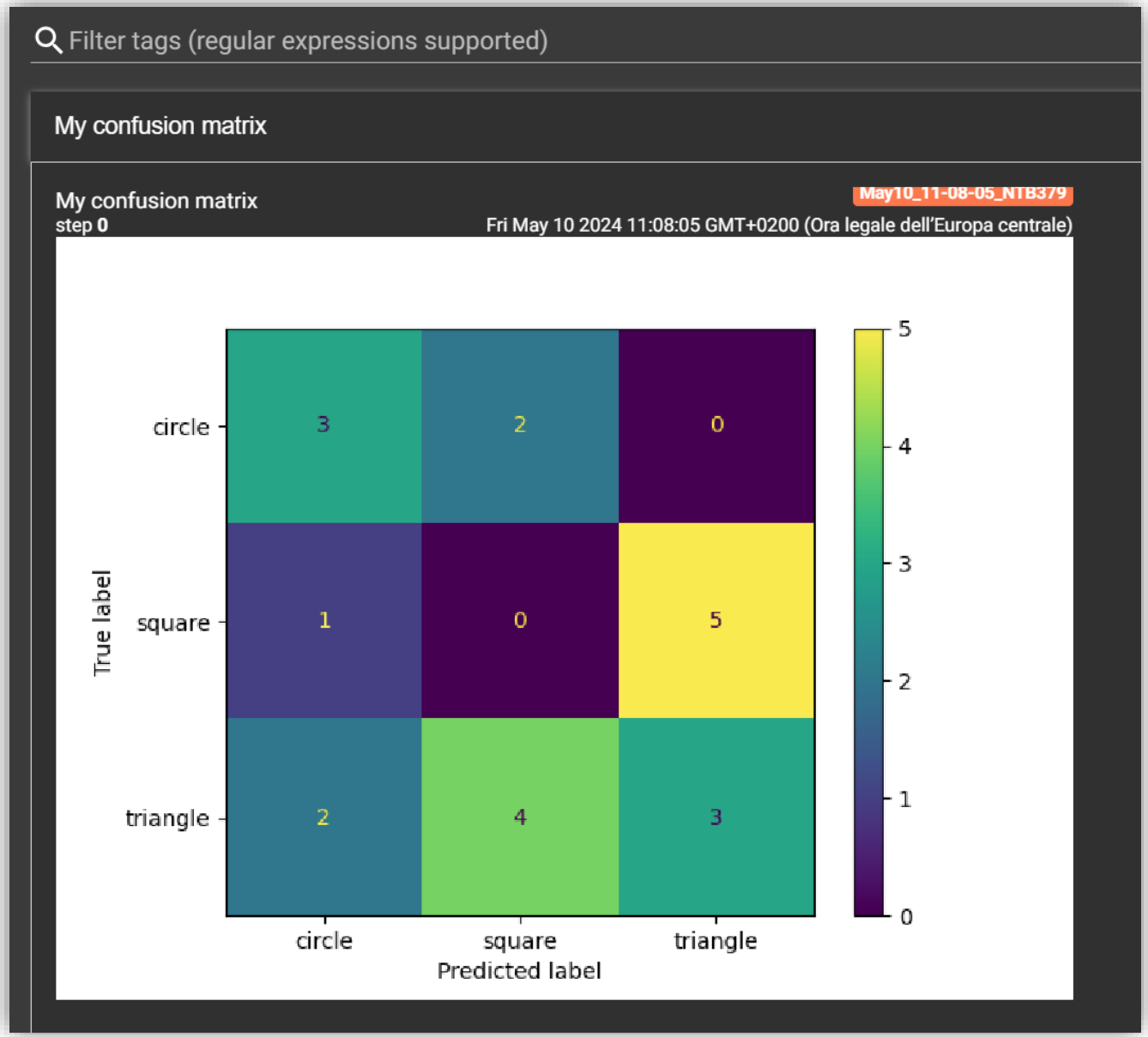
Esercitazione

Esercitazione

- Completare il contenuto ‘sfocato’ al fine di:
  - Ottenere la *figure* della confusion matrix.
  - Loggare la *figure* nella TensorBoard.

- Al termine:
  - Eseguire lo script.
  - Aprire una sessione TensorBoard.
  - Verificare il caricamento della *figure*.

Esercitazione



Esercitazione