Machine Learning Approaches for Job Failure Prediction in HTC Systems

Alessio Arcara

CNAF

Grande centro di calcolo:

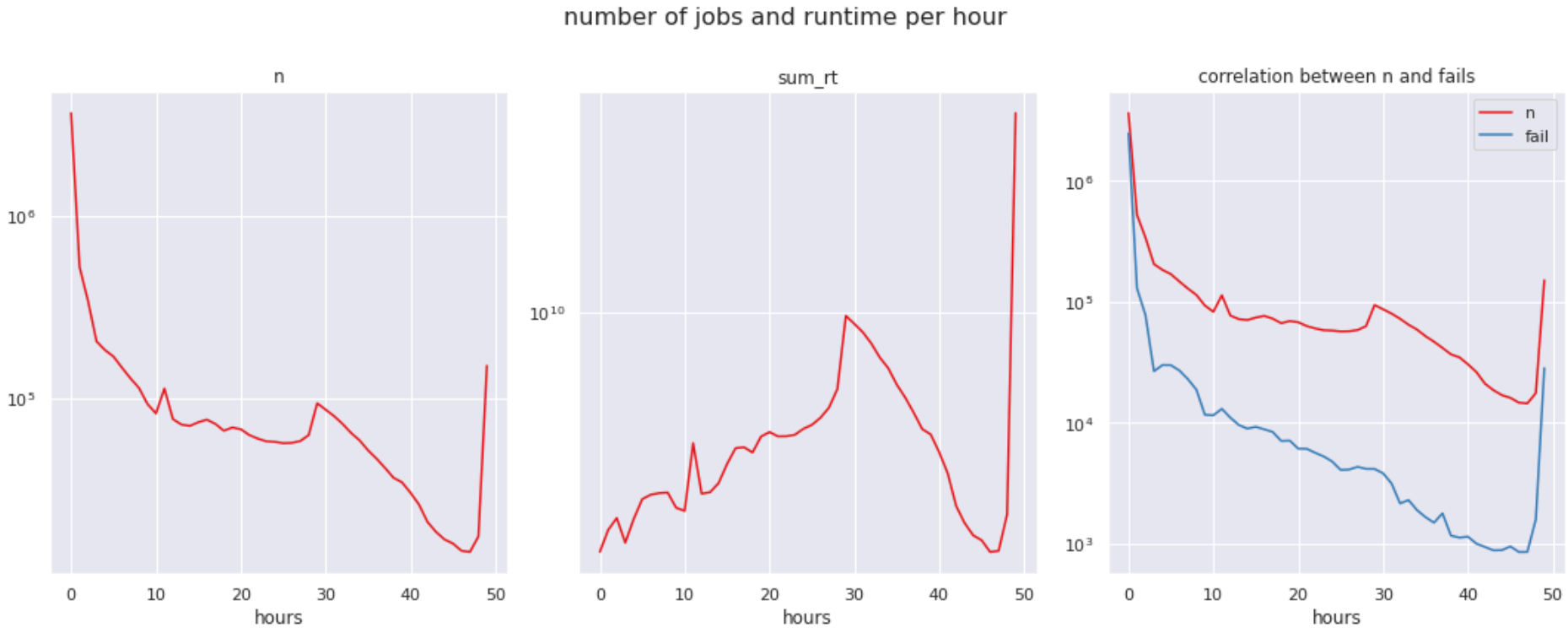
- Circa 59k core distribuiti su O(10^3) host fisici
- O(50) gruppi di utenti
- Jobs provenienti da vari esperimenti scientifici

I sistemi HTC, come questo, seguono un paradigma di calcolo batch:

- Gli utenti inviano i loro jobs che vengono inseriti in una coda
- HTCondor seleziona i jobs dalla coda per assegnarli ai nodi di calcolo per l'esecuzione

High-Throughput Computing

I sistemi HTC sono progettati per gestire un grande numero di job, massimizzando l'utilizzo delle risorse disponibili



prevedere un job che fallisce → consente di ridurre lo spreco di risorse

idea: predire il fallimento di un job di `lunga durata` è sicuramente più importante

Job ‘zombie’ detection

queue	too_much_time	size	perc	time_lost
lhcb	192	262251	0.073212	576
juno	151	10137	1.489593	453
atlas	45	270086	0.016661	135
lhcf	8	1594	0.501882	24
belle	2	42087	0.004752	6

Solo a Marzo 2023: **1194 giorni** di calcolo persi.

- Jobs che terminano senza rilasciare l’host fisico, causando `leakage` delle risorse fino al timeout.
  - Timeout grid: 3 giorni
  - Timeout local: 7 giorni
- L’addestramento di modelli di ML per tali job è complicato → sono rari!

Uno sguardo al Dataset

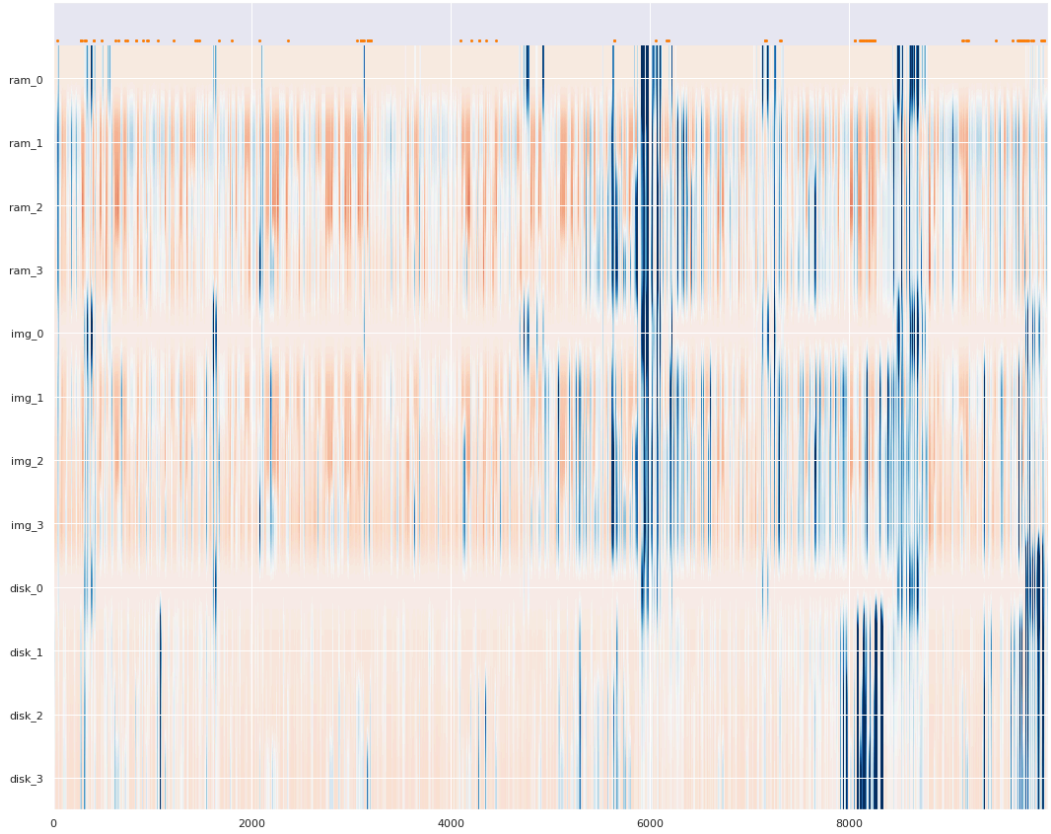
- Stato (RAM, DISK, SWAP) dei jobs campionati ogni 3 minuti
- Ma il batch system aggiorna lo stato ogni 15 minuti!
- **Grandezze monotone non decrescenti**

Stato dei jobs | Risorse utilizzate

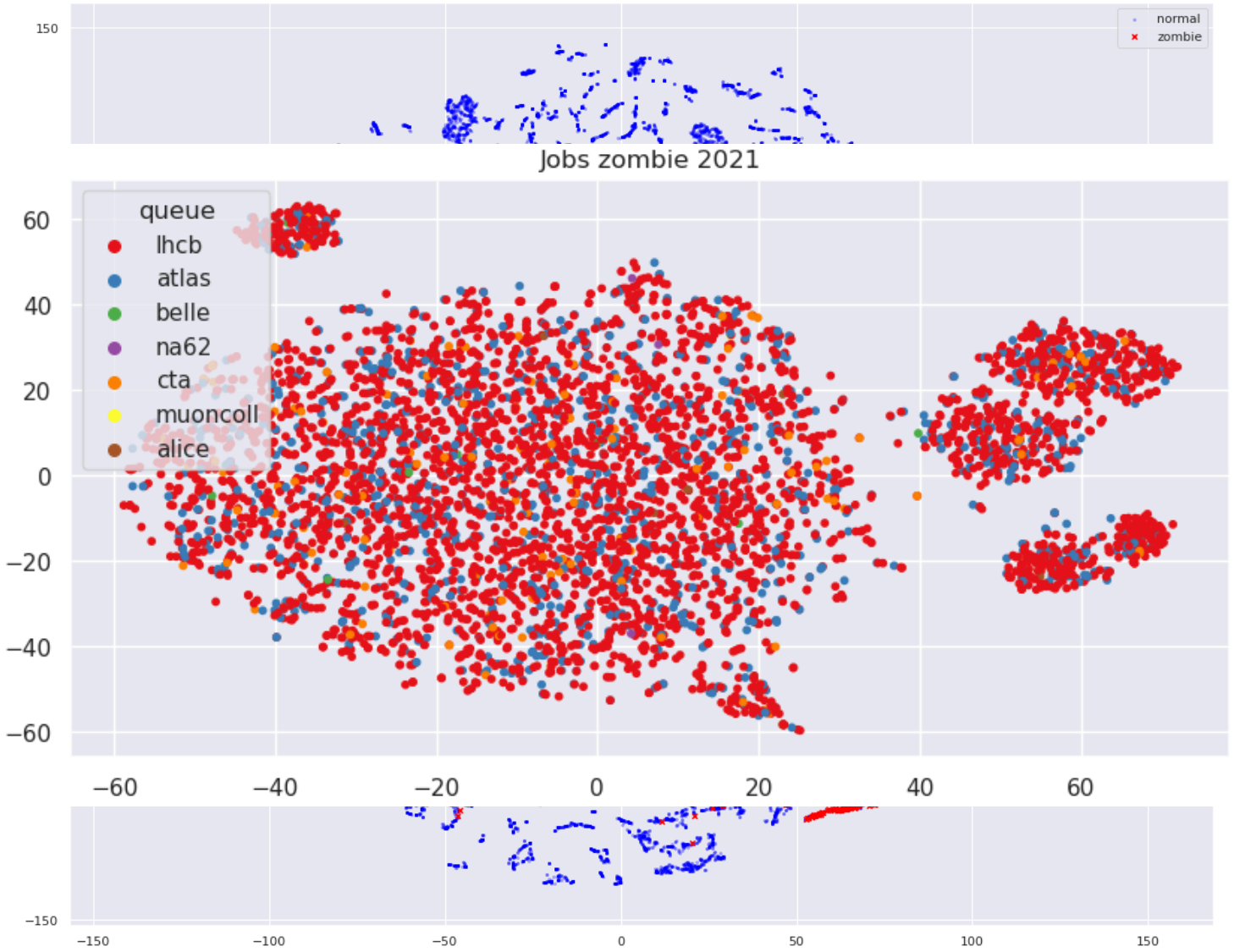
	job	queue	fail	mint	maxt	t	ram	swap	disk	job_work_type	job_type	days
0	10392841.0_ce03-htc	lhcb	0	1678697293	1678817707	0	0.000000	0.000007	0.000007	lhc	grid	2
1	10392841.0_ce03-htc	lhcb	0	1678697293	1678817707	1	0.877140	2.816248	0.000827	lhc	grid	2
2	10392841.0_ce03-htc	lhcb	0	1678697293	1678817707	2	0.986448	2.940984	0.008437	lhc	grid	2
3	10392841.0_ce03-htc	lhcb	0	1678697293	1678817707	3	1.014304	2.974192	0.015939	lhc	grid	2
4	10392841.0_ce03-htc	lhcb	0	1678697293	1678817707	4	1.028876	2.988056	0.022468	lhc	grid	2
5	10392841.0_ce03-htc	lhcb	0	1678697293	1678817707	5	1.063332	3.205468	0.028817	lhc	grid	2
6	10392841.0_ce03-htc	lhcb	0	1678697293	1678817707	6	1.065080	3.205468	0.037459	lhc	grid	2
7	10392841.0_ce03-htc	lhcb	0	1678697293	1678817707	7	1.066992	3.213924	0.045447	lhc	grid	2
8	10392841.0_ce03-htc	lhcb	0	1678697293	1678817707	8	1.147368	3.336116	0.051517	lhc	grid	2
9	10392841.0_ce03-htc	lhcb	0	1678697293	1678817707	9	1.158024	3.336116	0.060507	lhc	grid	2

Uno sguardo al Dataset

Possiamo utilizzare una `heatmap` per visualizzare i jobs zombie in relazione a DISK, SWAP e RAM



Spazio latente di un autoencoder

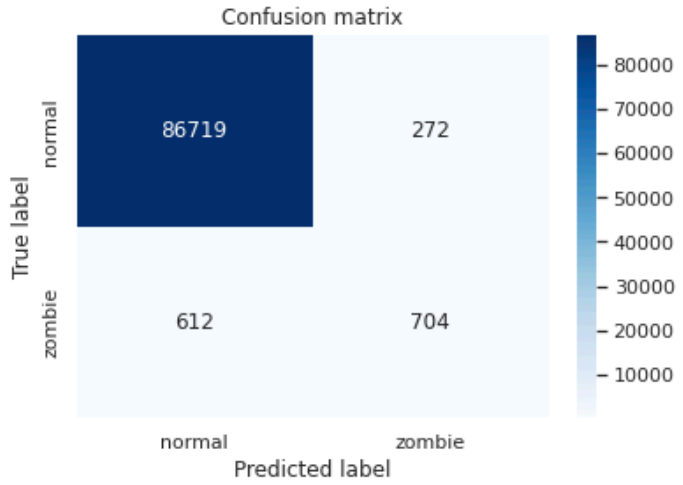


Addestramento supervisionato modelli (un’ora)

- Prima ora di vita di un job su logs **fine 2021**
- **Input**: i valori di utilizzo di DISK, SWAP e RAM nella prima ora delle serie storiche sono state trasformati in *features* (ad esempio: DISK_0, DISK_1, DISK_2, DISK_3)
- Sulla prima metà di settembre 2021, sono stati addestrati diversi modelli di ML e, tra essi, il modello risultato vincente è stato XGBoost

Addestramento supervisionato modelli (un’ora)

```
*** Confusion matrix ***
```

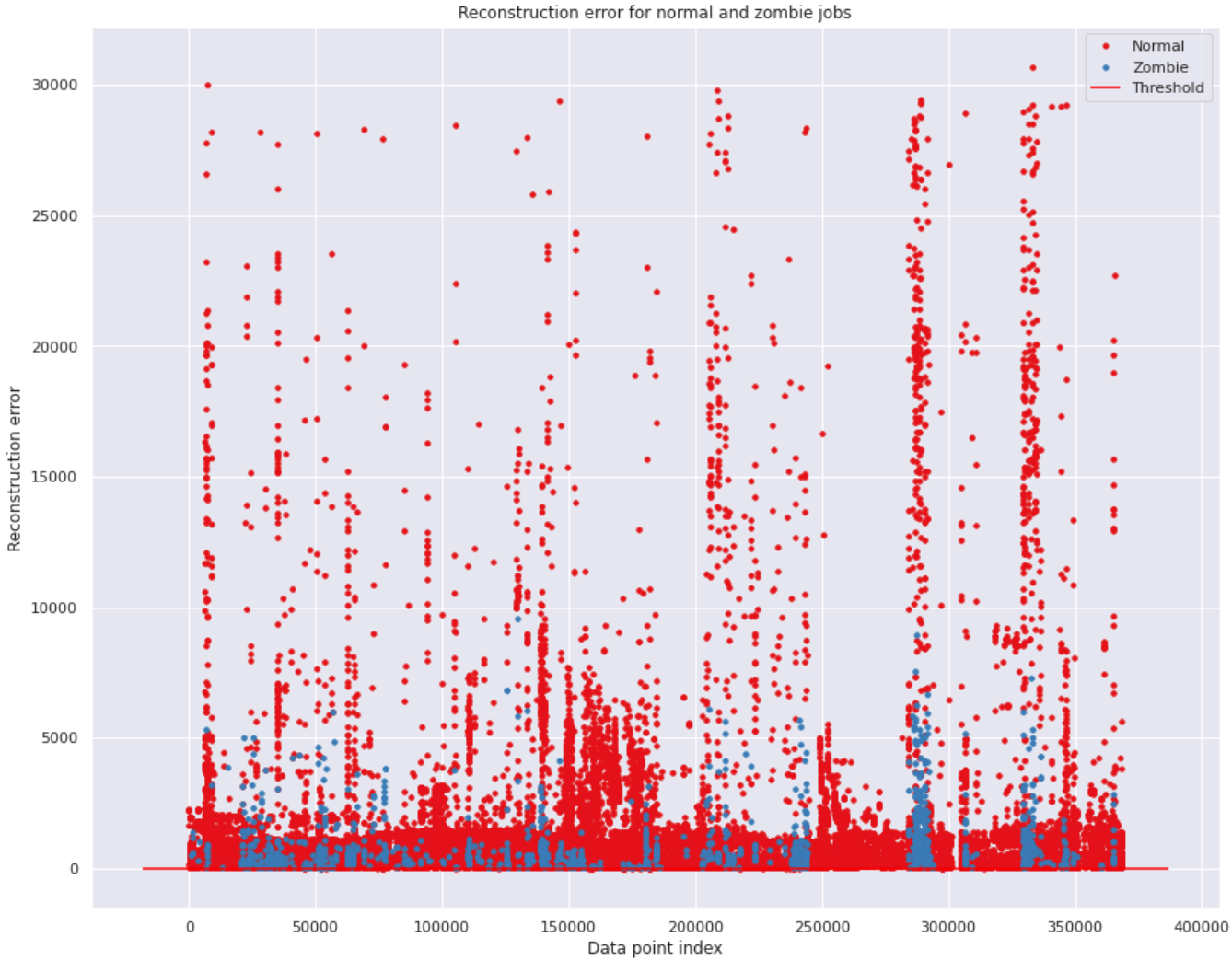



```
*** Precision, Recall, F1-measure per classe e media ***

              normal    zombie       all
precision   0.992992  0.721311  0.857152
recall      0.996873  0.534954  0.765914
f1_measure  0.994929  0.614311  0.804620

*** Calcolo intervallo di confidenza con Confidenza=0.99 con N=88307 per accuracy e f1-measure ***

accuracy: (0.9899894685585514), intervallo confidenza: (0.9890889980011854, 0.9908163144833613)
f1-measure: (0.804619785178991), intervallo confidenza: (0.8011601452485392, 0.8080336536772138)
```

Addestramento supervisionato modelli (un giorno)

- Prime 24 ore di vita di un job su logs **inizio 2023** → *padding* e *truncate*
- **Input**: tensore 3D (batch_size, time_steps, features)
- Architetture di **reti neurali** utilizzate:
  - CNN → *feature extraction*
  - ResNet-like (3 Residual Blocks)
  - LSTM → *long term dependence*
  - CNN + LSTM
  - Transformer (Encoder)
- le reti neurali hanno mostrato performance inferiori alle aspettative
- il precedente modello XGBoost, addestrato solo sulla prima ora, è statisticamente migliore

Addestramento non supervisionato



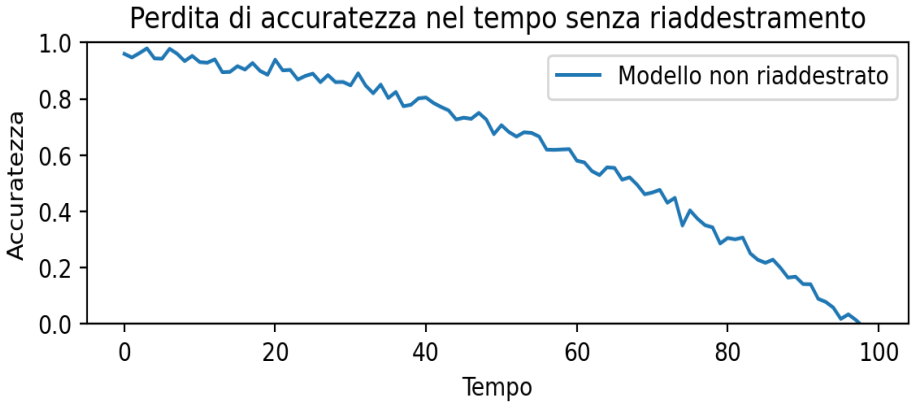
Tecniche per trattamento sbilanciamento dei dati

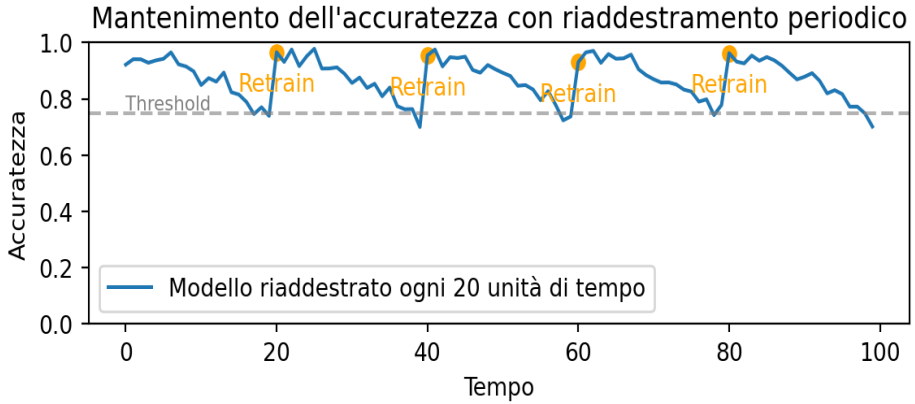
- **sottocampionamento** dei jobs `normali` → scelti casualmente
- **sovracampionamento** dei jobs `zombie` → generati da *variational autoencoder*
- **cost sensitive learning** → via *class_weight*
- **metriche** → F0.5 score

Conclusioni

Problemi:

- *Data drift*: i dati cambiano nel tempo e il modello perde accuratezza





- *Data quality*: valori monotoni non decrescenti e poche istanze relative alla classe meno rappresentata

Possibili sviluppi:

- ottenere features più significative (ad esempio: uso ram, disk e swap puntuali)

Grazie per l’attenzione!