ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

DIPARTIMENTO INFORMATICA – SCIENZA E INGEGNERIA

CORSO DI LAUREA MAGISTALE IN INGEGNERIA INFORMATICA

TESI DI LAUREA

in

Intelligent Systems M

Towards predictive maintenance in High Performance Computing using Deep autoencoder networks: a preliminary study on Marconi100 supercomputer

Candidato:

Silvia Brescia

Relatore:

Prof.ssa **Michela Milano**

Correlatore:

Dott. **Andrea Borghesi**

Anno Accademico 2020-2021

Abstract

I sistemi HPC, sono macchine complesse con componenti eterogenei che possono rompersi o avere dei malfunzionamenti. Queste problematiche possono provocare ingenti danni sia a livello economico che all'infrastruttura stessa. Allo stato dell'arte il rilevamento automatico delle anomalie su questi sistemi è una attività molto complicata e critica. Quello da cui si può trarre vantaggio per questo scopo è l'enorme quantità di dati provenienti da infrastrutture di monitoraggio di cui questi sistemi sono equipaggiati. Infatti la novità introdotta degli algoritmi di Machine Learning è la capacità di analizzare grandi quantitativi di dati e identificare in modo proattivo potenziali problematiche e relative cause. Questo lavoro di tesi ha l'obiettivo di condurre un'analisi preliminare, usando diversi modelli basati su tecniche di Deep Learning, per capire se è possibile fare anomaly detection e manutenzione predittiva su sistemi HPC. In particolare lo studio si svolge sul supercomputer MARCONI100 del Cinceca, di cui per la prima volta vengono analizzati i dati.

INDICE

INTRODUZIONE

I supercomputer sono grandi sistemi che comprendono più sottosistemi ad alte prestazioni che lavorano contemporaneamente e vicini al picco della loro capacità teorica ottimale. Questi sistemi possono avere dei malfunzionamenti causati da varie problematiche come la rottura dell'hardware o una configurazione errata. L'insorgere di queste problematiche può causare una diminuzione delle prestazioni complessive del sistema e non rispondere a requisiti di affidabilità, qualità e disponibilità del servizio. Ovviamente non è desiderabile che questo accada. Attualmente però, si riesce ad rilevare un'anomalia solo dopo che questa è avvenuta e ciò provoca costosi fermi del sistema e perdite anche a livello economico. Da questo deriva la necessità di trovare nuove tecniche capaci di prevedere comportamenti anomali, in modo tale da poter individuare in anticipo le anomalie, prevedere quando si verificherà un guasto e intervenire tempestivamente. Per poter realizzare questo obiettivo è necessario che il sistema sia continuamente monitorato e che quindi si possano raccogliere informazioni sul suo comportamento. La quantità di dati raccolta è molto consistente e sarebbe impossibile per gli amministratori di sistema individuare anomalie su di essa. Per mitigare questo problema, in questo lavoro di tesi viene presentata un'analisi preliminare per il rilevamento delle anomalie nei sistemi HPC attraverso tecniche di Deep Learning, nello specifico andando ad utilizzare delle reti neurali chiamate *autoencoders*. Per poter utilizzare queste tecnologie bisogna avere dati che siano *labeled*, cioè attraverso i quali si possano distinguere rilevamenti avvenuti durante anomalie o in periodi normali. Spesso questo requisito è difficile da ottenere, specialmente per i sistemi HPC, in quando le etichette sui dati sono scarse e spesso accade che sono gli amministratori di sistema stessi che quando rilevano un malfunzionamento assegnano un etichetta ai dati raccolti. Proprio per questo motivo in questa tesi si seguirà un approccio semi-supervisionato che usa dei dati parzialmente etichettati e un approccio non supervisionato. Verrà illustrato come sono stati costruiti i vari modelli di autoencoder in grado di eseguire *Anomaly Detection* per un sistema HCP. Il sistema in analisi è il supercomputer MACRONI100, uno tra i più potenti a livello globale di proprietà del Cineca. Per eseguire questo studio sono stati messi a diposizione i dati raccolti su alcuni nodi dei sistema relativi a circa 9 mesi di utilizzo. I dati di questo sistema, messo in funzione in Aprile 2020, vengono analizzati per la prima volta durante la stesura di questo elaborato. In dettaglio verrà prima eseguita una analisi sui dati raccolti di alcuni nodi nel sistema in modo tale da evidenziare il comportamento delle

variabili osservate in periodi anomali e in periodi normali. Poi si passerà alla costruzione di tre modelli di autoencoder:

- un autoencoder semplice con approccio semi supervisionato allenato con due funzioni di perdita differenti (Mean Standard Error e Mean Absolute Error)
- un autoencoder variazionale con approccio non supervisionato con spazio latente a 2 e 3 dimensioni
- un autoencoder variazionale con approccio semi supervisionato con spazio latente a 2 e 3 dimensioni

Infine verrà eseguito un confronto tra le tecniche utilizzate per valutare quale di esse ottiene i risultati migliori per il sistema in analisi e verranno sottolineati gli aspetti essenziali che servono per arrivare una predizione delle anomalie per i sistemi HCP. Attraverso questo lavoro sarà possibile in futuro per esperti del dominio riuscire a valutare le cause delle anomalie e arrivare anche alla prevenzione dei problemi.

Nel dettaglio la tesi è così suddivisa:

- Nel capitolo 1 vengono descritti i sistemi HPC, il supercomputer MARCONI100 e gli strumenti utilizzati per la realizzazione di questo elaborato;
- Nel capitolo 2 si introduce la tematica dell'Anomaly Detection e della Predictive Maintenance nell'ambito dei sistemi HPC;
- Nel capitolo 3 si espone l'analisi preliminare sui dati relativi ai nodi del supercomputer e il confronto tra i risultati ottenuti;
- Nel capitolo 4 si mostra lo stato dell'arte relativo ai vari algoritmi di Machine Learning e alle loro applicazioni allo scopo di fare anomaly detection;
- Nel capitolo 5 viene riportata la costruzione del modello di Machine Learning bastato su un autoencoder semplice con approccio semi supervisionato;
- Nel capitolo 6 si illustra la costruzione del modello di Machine Learning bastato su un autoencoder VAE con approccio non supervisionato e semi supervisionato;
- Nel capitolo 7 viene effettuato un confronto tra i vari modelli costruiti e vene valutato quale risulti essere più efficace per l'Anomaly Detection sui sistemi HPC;
- Nel capitolo 8 si riepilogano i risultati ottenuti e si evidenzia quali potranno essere gli sviluppi futuri dell'analisi condotta in questo lavoro

CAPITOLO 1

Sistemi HPC e strumenti utilizzati

1.1 I sistemi HPC

L'incredibile aumento del volume di dati che deriva dalla sviluppo delle recenti tecnologie, ha reso i processi tradizionali di analisi molto più complessi e difficili da gestire. In particolare questo coinvolge le applicazioni che richiedono elaborazione tempestiva e big data come ad esempio, immagini satellitari, dati provenienti da sensori, operazioni bancarie, server web, social network, che quindi necessitano di meccanismi efficienti per la raccolta, l'archiviazione, l'elaborazione e analisi di questi dati. [1] Per rispondere a queste richieste computazionali, ci si sta dirigendo verso modelli di tipo ***High Performance Computing*** (HPC). L'espressione High Performance Computing viene usata per identificare l'utilizzo di sistemi di elaborazione di grande potenza in grado di fornire delle prestazioni molto elevate nell'ordine dei PetaFLOPS (10^{15}operazioni in virgola mobile al secondo), costituiti dalla combinazione di un elevato numero di nodi di elaborazione. A seconda dell'architettura scelta, questi nodi possono essere composti singoli processori oppure da veri e propri computer indipendenti e collegati tra loro in reti ad alta velocità. Gli approcci utilizzati tipicamente per la realizzazione di un sistema di HPC sono due [2]:

- ***Cluster Computing***: si combina la capacità elaborativa di un insieme di computer che comunque possono anche operare autonomamente. Un cluster sarà generalmente costituito da workstation o PC ad alte prestazioni interconnessi da una rete ad alta velocità e funzionare come una raccolta integrata di risorse e può avere una singola immagine di sistema [3].
- ***Massive Parallel Processing***: nell'approccio meglio definito di elaborazione parallela massiva (MPP, Massive Parallel Processing) si combina invece la potenza di decine o centinaia di processori che sono fisicamente collocati in un unico sistema fisico. Ogni nodo può avere una varietà di componenti hardware ma generalmente è costituito da una memoria principale e da uno o più processori. Alcuni nodi, possono anche avere periferiche come dischi o sistemi di backup. Ogni nodo esegue una copia separata del sistema operativo [4].

Caratteristica	Cluster	MPP
Numero di nodi	O(100) o meno	O(100) – O(1000)
Complessità nodi	Medium grain	Fine grain o medium
Comunicazione tra i nodi	Scambio di messaggi	Scambio di messaggi/ variabili condivise per una memoria condivisa distribuita
Scheduling delle attività	Code multiple ma coordinate	Coda di esecuzione singola sull'host
Server Side Include Support	Desiderato	Parziale
Copie del nodo OS e tipologie	N OS omogenei o micro-kernel	N micro-kernels monolitici o OS layered
Spazio degli indirizzi	Multiplo o singolo	Multiplo
Sicurezza tra i nodi	Richiesta se viene esposto	Non richiesta
Proprietà	Una o più organizzazioni	Una organizzazione

Un sistema HPC è una combinazione di nodi di elaborazione ad alte prestazioni, un fabric di interconnessione a bassa latenza con larghezza di banda elevata, storage parallelo ad alte prestazioni e software di sistema, che soddisfa i requisiti più impegnativi per HPC e analisi dei dati ad alte prestazioni [5].

Un tipico stack di soluzioni HPC è composto da:

- Infrastruttura
    - Hardware servers ad alte prestazioni
    - Archiviazione ad alta velocità
    - Interconnessione ad alta velocità
- Software di gestione
- Software di pianificazione
- Ambiente di sviluppo software
- Applicazioni
- Soluzione Health Insight

Queste soluzioni possono essere applicate in un'ampia gamma di settori. Ad esempio, automobilistico, aziende aerospaziali ed elettroniche che si concentrano sull'automazione della

progettazione elettronica (EDA), computer-aided engineering (CAE), big data analytics e intelligenza artificiale (AI).

1.2 Il sistema MARCONI100

MARCONI100 è un cluster accelerato basato su architettura IBM Power9 e GPU Volta NVIDIA, acquisito da Cineca nell'ambito dell'iniziativa europea PPI4HPC. È disponibile da aprile 2020 al pubblico italiano e ai ricercatori industriali. La sua capacità di calcolo è di circa 32 PFlops. Il ***Cineca*** è un Consorzio no-profit, composto da 67 Università italiane e 13 Istituzioni. ***SCAI*** (SuperComputing Applications and Innovation) è il dipartimento di High Performance Computing del CINECA, il più grande centro di calcolo in Italia e uno dei più grandi in Europa. La missione di SCAI è quella di accelerare la scoperta scientifica fornendo risorse di calcolo ad alte prestazioni, sistemi e strumenti di gestione e archiviazione dei dati e servizi e competenze HPC in generale, con l'obiettivo di sviluppare e promuovere servizi tecnico-scientifici relativi al calcolo ad alte prestazioni per l'Italia e comunità di ricerca europea [6].

Il progetto Marconi100 si pone al 11° posto come più potente supercomputer a livello globale, e al 2° a livello europeo [7]. Le caratteristiche tecniche sono:

Nodi: 980
Processori: 2x16 cores IBM POWER9 AC922 at 3.1 GHz
Acceleratori: 4 x NVIDIA Volta V100 GPUs, Nvlink 2.0, 16GB
Cores: 32 cores/nodo
RAM: 256 GB/nodo
Peak Performance: ~32 PFlop/s

Figura 1 - MARCONI100

Marconi100 è composto da 980 nodi di calcolo e 8 nodi di accesso, collegati con una rete Mellanox Infiniband EDR organizzata in un'architettura chiamata DragonFly ++. Ogni nodo è costituito da 2 socket Power9, ciascuno con 16 core e 2 GPU Volta (32 core e 4 GPU per nodo). Il multi-threading è attivo con 4 thread per core fisico (128 thread totali – o cpu logiche – per nodo).

1.3 Gli strumenti utilizzati

I nodi del MARCONI100 su cui si è lavorato sono: r205n02, r205n17, r205n20, r206n06, r206n14, r206n17, r206n18, r206n19, r208n02, r208n04, r208n07, r208n12, r210n05, r211n19, r212n06, r215n05, r218n03, r208n02, r224n20, r239n12, r243n11, r249n06, r253n06, r254n01, r256n05.

Per l'analisi dei dati e la realizzazione dei modelli di Machine Learning si è scelto di usare *Jupyter* [8], un'applicazione Web open source che permette di scrivere codice in python e che dispone di una interfaccia semplice ed interattiva. Si sono utilizzate anche alcune librerie come ad esempio *keras*, per la realizzazione dei modelli, *pandas* per la gestione dei dati e *matplotlib* per la gestione dei grafici.

CAPITOLO 2

Anomaly detection e predictive maintenance nei sistemi HPC

2.1 Anomaly detection

Per **Anomaly detection,** si intende il processo di ricerca di pattern in un set di dati il cui comportamento non è normale come previsto. Questi comportamenti imprevisti sono anche definiti come anomalie o valori anomali [9]. Le anomalie nei dati si verificano molto raramente e le loro caratteristiche sono significativamente diverse da quelle delle istanze normali. In genere, i dati anomali sono collegati a qualche tipo di problema o evento raro come hacking, frode bancaria, apparecchiature malfunzionanti, difetti strutturali/guasti infrastrutturali o errori testuali. Per questo motivo, dal punto di vista aziendale, è essenziale identificare le anomalie reali piuttosto che i falsi positivi o il rumore dei dati. La Figura 2 illustra le anomalie in un semplice set di dati bidimensionali. I dati hanno due regioni normali, N1 e N2, poiché la maggior parte delle osservazioni risiede in queste due regioni. Punti sufficientemente lontani dalle regioni, come i punti O1 e O2, e i punti nella regione O3, sono anomalie [10].

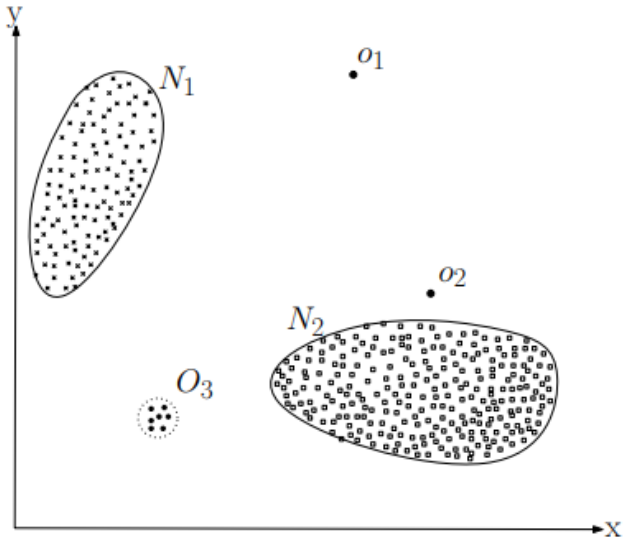

Figura 2 - anomalie in un data-set bidimensionale

2.2 Anomaly detection e predictive maintenance nei sistemi HPC

I sistemi HPC, sono macchine complesse con componenti eterogenei che possono rompersi o funzionare male. Il rilevamento automatico delle anomalie in questi sistemi è un compito impegnativo e critico, poiché questi sistemi dovrebbero funzionare 24 ore su 24, 7 giorni su 7. In realtà, molte cose potrebbero andare storte e causare una diminuzione delle prestazioni complessive di una macchina HPC: hardware può rompersi, funzionare male o essere configurato in modo errato, i programmi software possono contenere bug o raggiungere stati indesiderabili. Questo è un problema importante sia per la comunità informatica scientifica che per i data center e i fornitori di cloud, poiché la loro attività dipende fortemente dalla disponibilità dei loro servizi [11]. Ad esempio, è stato stimato che nel 2016 Amazon avrebbe perso 15 milioni di dollari per un'ora di inattività [12]. Molte moderne macchine HPC e data center sono dotate di un'infrastruttura di monitoraggio che raccoglie informazioni caratterizzanti lo stato del sistema e dei componenti sottostanti, potendo contare su un vasto serie di sensori di misura. Però, data l'enorme quantità di dati di monitoraggio, l'identificazione in tempo reale di problemi e situazioni indesiderate è un compito difficile per gli amministratori di sistema [13]. Per questo motivo sorge la necessità di trovare nuove tecniche capaci di prevedere comportamenti anomali, in modo tale da poter intervenire tempestivamente in caso di situazioni anomale e di ridurre al minimo i danni sia al sistema che quelli economici. Seguendo questa filosofia si sviluppa il concetto di ***predictive maintenance*** che ha come scopo la raccolta e l'analisi in tempo reale dei dati provenienti dai macchinari per individuare in anticipo le anomalie e prevedere quando si verificherà un guasto, in modo da intervenire tempestivamente ed evitare costosi fermi macchina. Grazie alle nuove tecnologie si può osservare ogni aspetto di ogni parte della macchina in tempo reale durante il suo funzionamento. Ciò significa che le prestazioni della macchina vengono costantemente monitorate e l'apprendimento automatico è in grado di analizzare i dati provenienti da ogni macchina per rilevare cambiamenti indesiderati nel funzionamento. Quindi si riesce ad intervenire prima che una situazione anomala/guasto avvenga. Inoltre, osservare questi dati porta anche a poter pianificare in modo ottimale le manutenzioni, migliorando le prestazioni, riducendo i costi e estendendo la vita utile del sistema. Gli sforzi tecnologici sono stati principalmente orientati quindi a passare da processi di manutenzione programmata (intervento prefissato dopo un periodo di tempo definito) a processi di manutenzione preventiva (intervento all'avvicinarsi di una soglia sulle ore di utilizzo della macchina). La novità introdotta dall'intelligenze artificiale è la possibilità di identificare pattern che generano potenziali anomalie in modo più accurato

rispetto a quanto già compreso e codificato dagli operatori umani. Questa, infatti, è la specificità degli algoritmi di machine learning che acquisiscono grandi quantità di dati endogeni sull'impianto ed esogeni, ovvero derivanti dal contesto (dati storici, temperature, campo magnetico o da piattaforme esterne, etc.), li analizzano, identificano in modo proattivo potenziali problematiche e le relative cause. Un modello di manutenzione predittiva permette quindi di intervenire prima della rottura o del malfunzionamento di un singolo componente riuscendo a ***predire*** la probabilità di rottura in base al diverso comportamento (anche minimale) di una serie di proprietà che sono state registrate [14]. La base di partenza per poter fare manutenzione predittiva è riuscire ad estrarre valore dai dati. Lo sviluppo di una soluzione di manutenzione predittiva basata sull'analisi prevede le seguenti fasi:

- Analisi preliminare: definire le feature che possono essere significative da monitorare
- Raccolta dei dati: avviare un sistema di monitoraggio sulle feature scelte e raccogliere informazioni sia su dati normali che anomalie
- Modellazione: creazione di un modello che dovrà essere addestrato con i dati raccolti
- Training e test set: si addestra il modello e si valuta la sua efficacia con i test set
- Ottimizzazione: strutturare il processo di predizione automatizzata in modo dinamico, andando ad ottimizzare progressivamente il modello decisionale sulla base delle predizioni eseguite e degli eventuali cambiamenti del contesto.

La manutenzione predittiva basata sull'analisi dei dati consente quindi di rispondere con agilità e in modo proattivo ai problemi, riducendo la necessità di interventi di assistenza e quindi diminuendo i periodi di inattività non pianificati fino al 30%, aumentando dell'83% la velocità di soluzione dei problemi e riducendo del 75% il tempo necessario di presenza in loco (dati PTC).

In questo lavoro di tesi si andrà a sperimentare in modo pratico quanto descritto in precedenza, infatti l'obiettivo è creare un sistema di predizione delle anomalie sul sistema MARCONI100. In primo momento si analizzeranno i dati raccolti sul sistema MARCONI100 relativi a circa 9 mesi di attività, in particolare è la prima volta che questi dati vengono studiati e quindi si tratterà di un'analisi preliminare che potrà essere ampliata nel futuro prendendo in considerazione altri aspetti, poi si passerà alla strutturazione e al testing di alcuni modelli di apprendimento per valutare quali di questi si dimostra più performante ed efficace per lo scopo.

CAPITOLO 3

Analisi preliminare dei dati

Prima di passare alla realizzazione dei modelli di Machine Learning è stato necessario condurre una analisi preliminare sui dati di monitoraggio del sistema per poter capire in primis come sono strutturati, quale formato hanno e quali sono le informazioni che possono essere utili allo scopo di questa tesi.

3.1 Il formato dei dati

I dati che si hanno a disposizione sul supercomputer MARCONI100 sono stati raccolti tramite un sistema di monitoraggio chiamato *Examon*, un framework per il monitoraggio e la manutenzione dei sistemi HPC, progettato per sistemi informatici su scala molto ampia, come i supercomputer.

Riesce a gestire una grande quantità di dati provenienti da molte fonti eterogenee. I dati da analizzare per questo elaborato sono salvati in formato parquet e per manipolarli è stato necessario usare la libreria *Pandas*.

Attraverso la lettura dei dati relativi ai nodi forniti, si riscontra che per ogni nodo vengono riportate 234 features.

Le informazioni sono organizzate sotto forma di tabelle, dove quindi, ogni colonna indica le metriche rilevate sul nodo e ogni riga rappresenta un particolare istante temporale (timestamp).

Tra le feature di rilievo si individua la metrica ***label,*** che può assumere un valore binario, 0 per indicare uno stato *normale* del nodo e 1 per indicare uno stato *anomalo*.

Il campionamento avviene ogni 5 minuti, ma nel dataset sono riportati i valori ad intervalli di 15 minuti, tranne in alcuni casi (quando si presenta un anomalia in un punto non incluso nell'intervallo temporale dei 15 minuti) in cui vengono riportati anche altre frazioni temporali.

Nei dataset sono presenti dei salti temporali, cioè per alcuni intervalli di tempo non si hanno dati a disposizione da consultare.

Le feature che vengono prese in analisi sono:

Nome della feature	Descrizione	Unità di misura
ambient	temperatura all'ingesso del nodo	°C
dimmX_temp	temperatura del modulo DIMM n. X. X = 0…15	°C
fanX_Y	velocità della ventola Y nel modulo X. X = 0..3, Y = 0,1	RPM
fan_disk_power	consumo energetico della ventola del disco	W
gpuX_core_temp	temperatura del core per la GPU id X. X = 0,1,3,4	°C
gpuX_mem_temp	temperatura della memoria per la GPU id X. X = 0,1,3,4	°C
pX_coreY_temp	temperatura del core n. Y nella CPU socket n. X. X=0..1, Y=0..21	°C
pX_io_power	consumo energetico per il sottosistema I/O per la CPU socket n. X. X = 0..1	W
pX_mem_power	consumo di energia per il sottosistema di memoria per la CPU socket n. X. X = 0..1	W
pX_power	consumo di energia per la CPU socket n. X. X = 0..1	W
pX_vdd_temp	temperatura del regolatore di tensione per la CPU socket n. X. X = 0..1	°C
pcie	temperatura al PCIExpress slots	°C
psX_input_power	potenza assorbita all'ingresso dell'alimentatore n. X. X = 0..1	W
psX_input_voltag	tensione all'ingresso dell'alimentatore n. X. X = 0..1	V
psX_output_curre	corrente all'uscita dell'alimentatore n. X. X = 0..1	A
psX_output_volta	tensione all'usicta dell'alimentatore n. X. X = 0..1	V
total_power	consumo di energia totale del nodo	W
boottime	l'ultima volta che il sistema è stato avviato	s
bytes_in	numero di byte in ingresso per secondo	bytes/sec
bytes_out	numero di byte in uscita per secondo	bytes/sec
cpu_aidle	percentuale di tempo dall'avvio della CPU inattiva	%
cpu_idle	percentuale di tempo in cui la CPU o le CPU erano inattive e il sistema non ha ricevuto una richiesta di I/O al disco in sospeso	%
cpu_nice	percentuale di utilizzo della CPU che si è verificato durante l'esecuzione a livello di utente con buona priorità	%
cpu_num		

cpu_speed	velocità della CPU in termini di MHz	MHz
cpu_steal		%
cpu_system	percentuale di utilizzo della CPU che si è verificato durante l'esecuzione a livello di sistema	%
cpu_user	percentuale di utilizzo della CPU che si è verificata durante l'esecuzione a livello di utente	%
cpu_wio	percentuale di tempo in cui la CPU o le CPU erano inattive durante il quale il sistema ha ricevuto una richiesta di I / O del disco in sospeso	%
load_fifteen	carico medio in quindici minuti	
load_five	carico medio in cinque minuti	
load_one	carico medio in un minuto	
mem_buffers	quantità di buffered memory	KB
mem_cached	quantità di cached memory	KB
mem_free	quantità di free memory	KB
mem_shared	quantità di shared memory	KB
mem_total	quantità totale di memoria visualizzata in KB	KB
part_max_used	percentuale massima utilizzata per tutte le partizioni	%
pkts_in	pacchetti in ingresso per secondo	packets/sec
pkts_out	pacchetti in uscita per secondo	packets/sec
proc_run	numero totale di processi in esecuzione	
proc_total	numero totale di processi	
swap_free	quantità di swap memory disponibile	KB
swap_total	Quantità di swap memory totale	KB
state		
label	indica lo stato dell'intero nodo, con un valore pari a 0 indicante uno stato normale e 1 indicante uno stato anomalo del nodo	
timestamp	istante temporale relativo alle letture	DataTime

Per ogni feature (tranne timestramp e label) si hanno due valori riportati nel dataset:

- ***avg***: rappresenta il valore realmente rilevato
- ***var***: variazione percentuale tra il valore rilevato e quello precedente

3.2 Il rack 205

I nodi dei supercomptur sono suddivisi in rack, cioè degli armadi. Per il rack 205 si andranno ad analizzare i nodi 02, 17, 20.

3.2.1 Nodo r205n02

Le letture in questo nodo vengono riportate in una tabella di 8384 righe e 234 colonne.

Il primo campionamento, risale al 2020-07-22 15:45:00 mentre l'ultimo risale al 2021-01-19 09:15:00.

index	timestamp	avg:ambient	avg:dimm0_temp	avg:state	label
14901	2020-07-22 15:45:00	19.106667	26.0	0.666667	1
14904	2020-07-22 16:00:00	18.626667	26.0	0.666667	1
14907	2020-07-22 16:15:00	18.426667	26.0	0.666667	1
14910	2020-07-22 16:30:00	18.973333	26.0	0.666667	1
14913	2020-07-22 16:45:00	19.120000	26.0	0.666667	1
...	...	...	...	...	...
24843	2021-01-19 08:15:00	21.266667	28.0	0.000000	0
24846	2021-01-19 08:30:00	21.000000	28.0	0.000000	0
24849	2021-01-19 08:45:00	21.493333	28.0	0.000000	0
24852	2021-01-19 09:00:00	21.720000	28.0	0.000000	0
24855	2021-01-19 09:15:00	21.773333	28.4	0.000000	0

Figura 3 - dataset nodo r205n02

Si possono individuare circa 6000 letture in cui il nodo risulta in stato *normale* e circa 2000 in cui risulta in stato *anomalo*. Non si sono riscontrati valori NaN. La velocità della CPU in termini di MHz (MHz), viene rilevata come un valore costante di 3800 MHz. La distribuzione delle anomalie su questo nodo è così rappresentata:

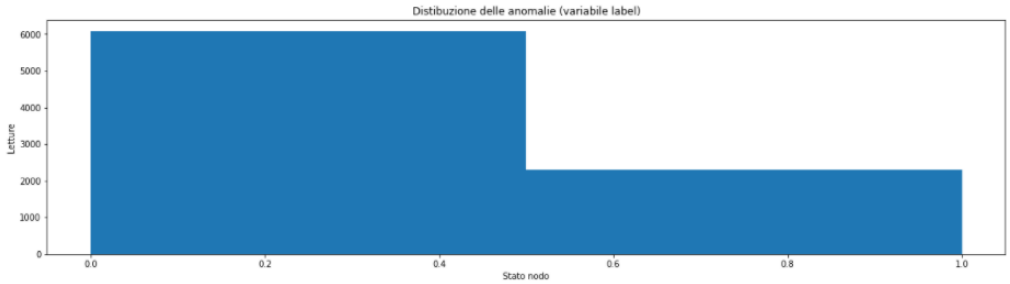

Figura 4 - distribuzione delle anomalie in base alla variabile label nodo r205n02

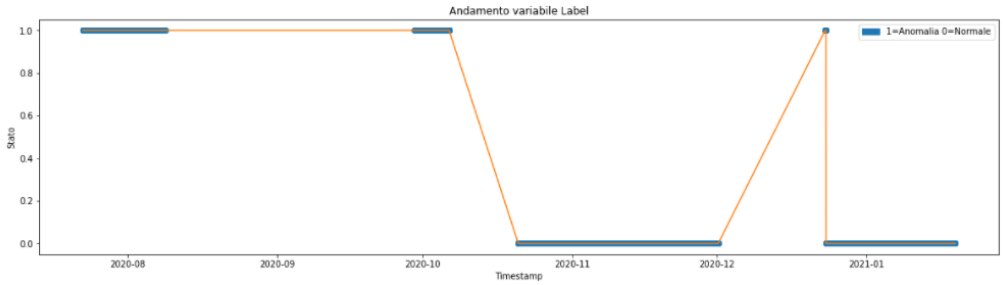

Figura 5 - andamento della variabile label nodo r205n02

Se si va ad osservare la feature *cpu_idle* si può notare che nei periodi di anomalia la CPU è sempre idle e questo è giustificato dal fatto che quando un sys admin riceve segnalazioni che un nodo si sta comportando male mette il nodo in stato DOWN+DRAIN usando un tool chiamato Nagios e in questa fase il nodo è sottoposto ad ulteriori controlli da parte del sys admin, e nuovi job non possono esservi eseguiti finché non viene rimesso in stato UP.

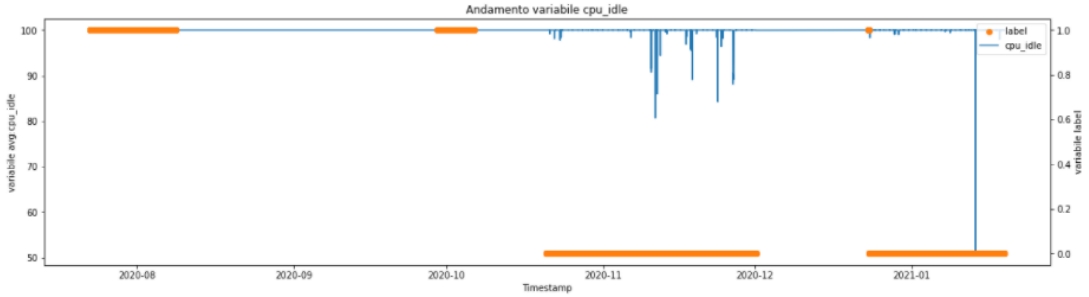

Figura 6 - andamento variabile cpu_idle nodo r205n02

Inoltre se si considera la percentuale di utilizzo della CPU che si è verificato durante l'esecuzione a livello di sistema o di utente, nel momento di anomalie questa risulta essere zero.

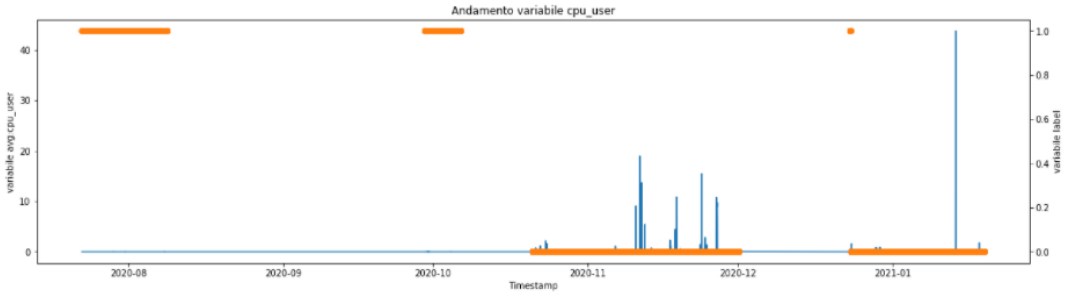

Figura 7 - andamento variabile cpu_user nodo r205n02

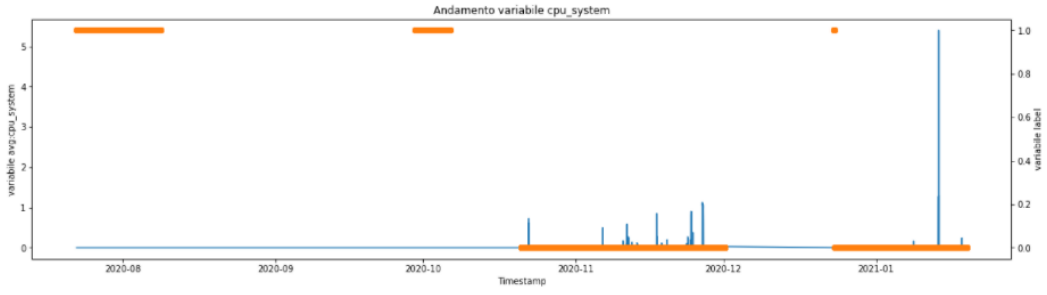

Figura 8 - andamento variabile cpu_system nodo r205n02

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un valore superiore a 0 e man mano che il sistema torna alla normalità il valore assunto si abbassa.

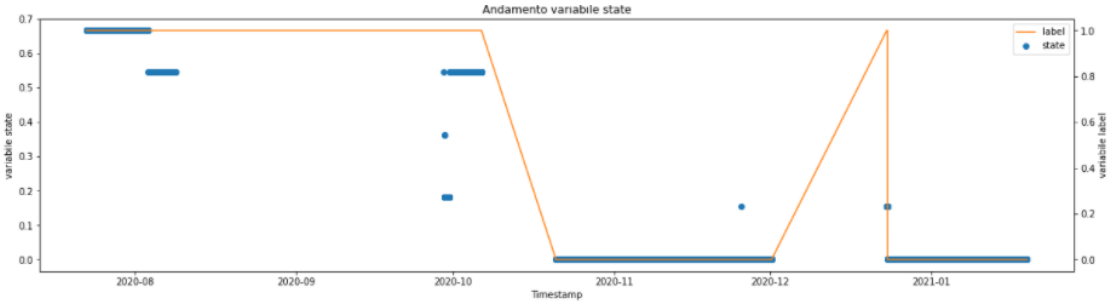

Figura 9 - andamento variabile state nodo r205n02

3.2.2 Nodo r205n17

Le letture in questo nodo vengono riportate in una tabella di 5330 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:15:00 mentre l'ultimo risale al 2021-02-18 14:45:00. Si possono individuare 5208 letture in cui il nodo risulta in stato *normale* e 122 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

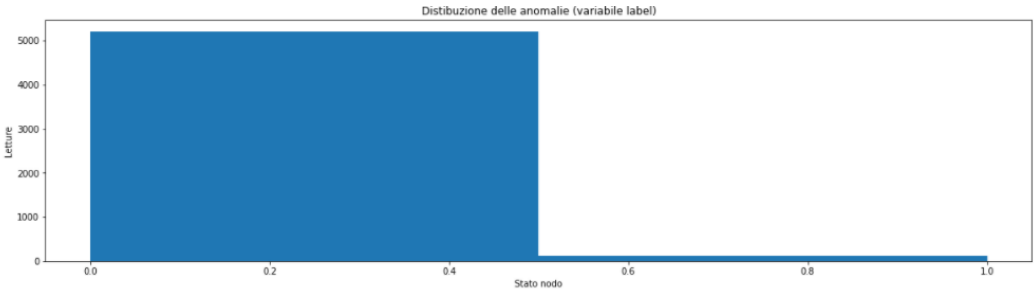

Figura 10 - distribuzione delle anomalie in base alla variabile label nodo r205n17

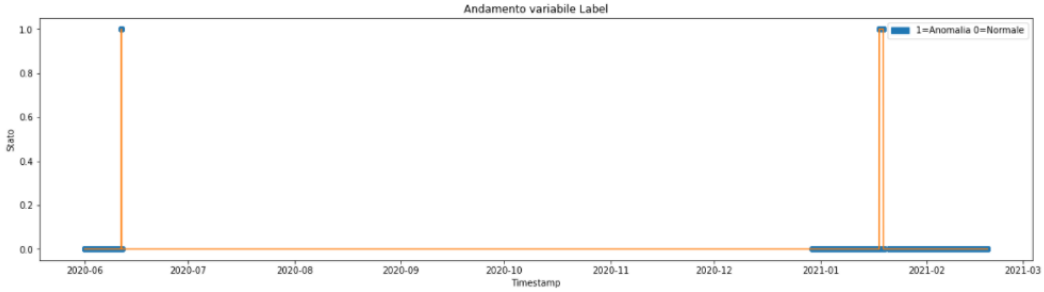

Figura 11 - andamento della variabile label nodo r205n17

Anche in questo caso, la variabile state ha valore 0 se il sistema funziona, altrimenti assume un valore superiore a 0 e man mano che il sistema torna alla normalità il suo valore si abbassa.

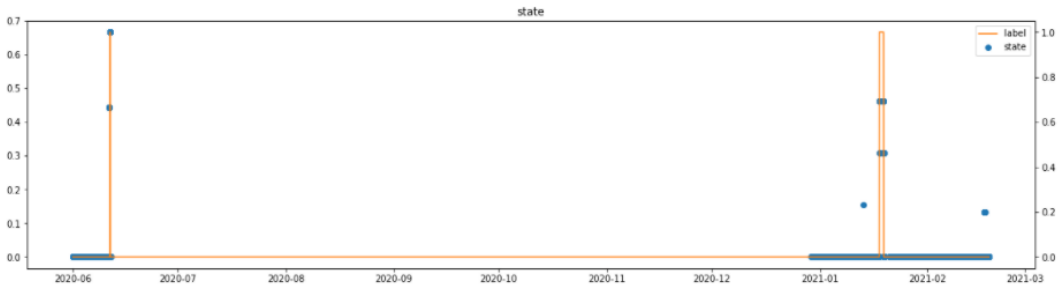

Figura 12 - andamento variabile state nodo n2015n17

3.2.3 Nodo r205n20

Le letture in questo nodo vengono riportate in una tabella di 14230 righe e 234 colonne. Le colonne rappresentano le feature rilevate. Il primo campionamento, risale al 2020-07-02 08:30:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 11556 letture in cui il nodo risulta in stato *normale* e 2674 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

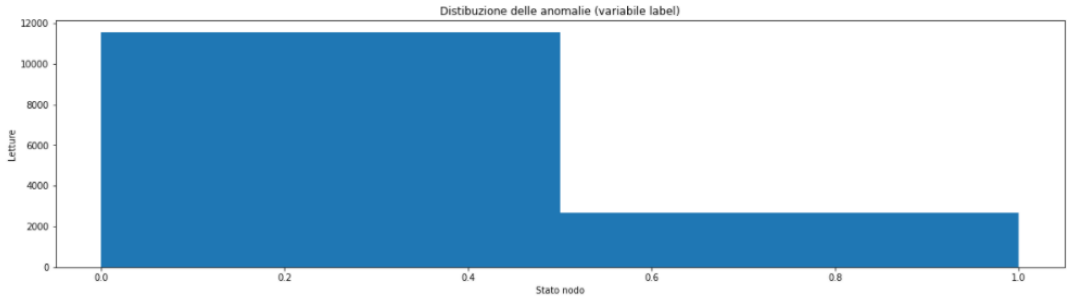

Figura 13 - distribuzione delle anomalie in base alla variabile label nodo r205n20

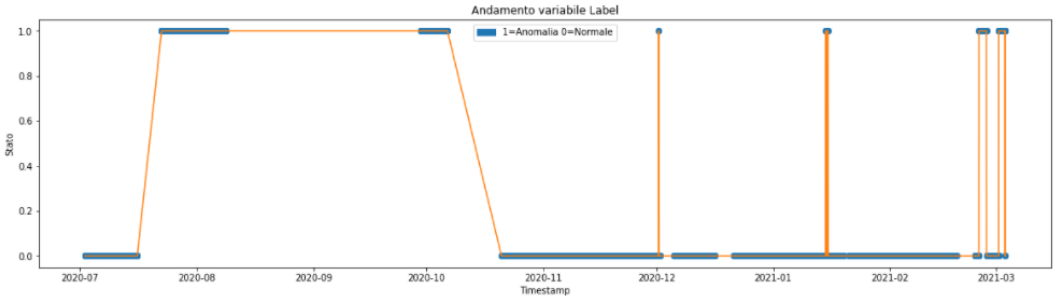

Figura 14 - andamento della variabile label nodo r205n20

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

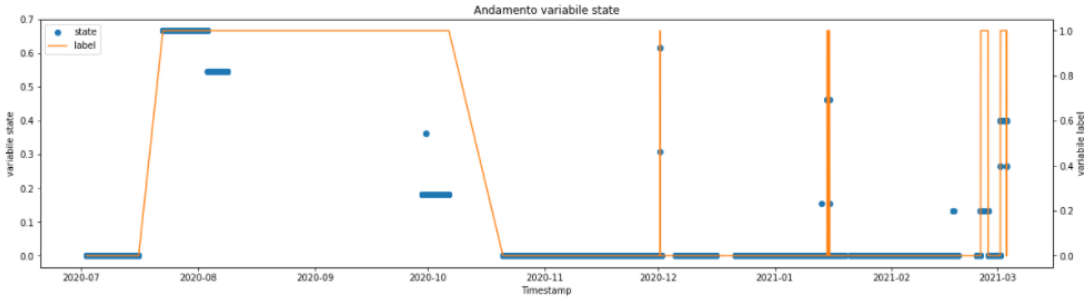

Figura 15 - andamento variabile state nodo n2015n20

Possiamo inoltre notare, facendo il plot dei pacchetti di input e output per secondo, che in prossimità di una anomalia il numero dei pacchetti di input supera di molto quello dei pacchetti di output, come si può vedere dalle rilevazioni in prossimità del 10/2020.

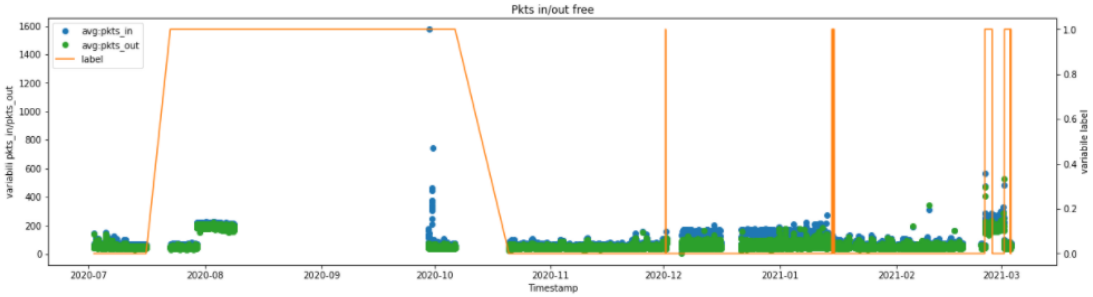

Figura 16 - pacchetti di input e output nodo r205n20

Se si va ad effettuare il plot sul carico medio a un minuto, a cinque minuti a quindici minuti, vedremo che si possono riscontrare dei picchi proprio in corrispondenza delle anomalie.

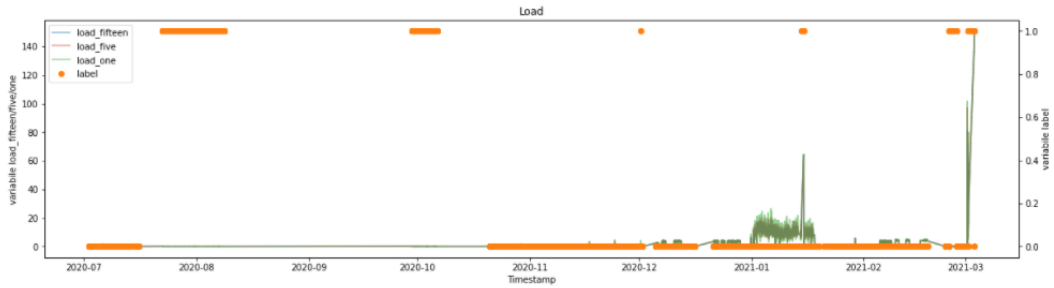

Figura 17 - carico medio nodo r205n20

Come valori costanti vengono rilevati cpu_nice = 0, cpu_speed=3800, cpu_steal=0.

3.2.4 Confronto tra i nodi del rack 205

	Shape	Primo camp.	Ultimo camp.	Stato anomalo	Salti	Numero anomalie	Media anomalie
n02	(8384,234)	2020-07-22 15:45:00	2021-01-19 09:15:00	27%	Si	7	86 h e 30 min
n17	(5330,234)	2020-05-31 22:15:00	2021-02-18 14:45:00	2%	Si	2	15 h e 15 min
n20	(14230, 234)	2020-07-02 08:30:00	2021-03-03 13:45:00	18%	Si	6	114 h e 30 min

Per poter capire se i nodi falliscono insieme o in momenti distinti, si è pensato di visualizzare lo stato della variabile label per tutti i nodi del rack in un'unica figura.

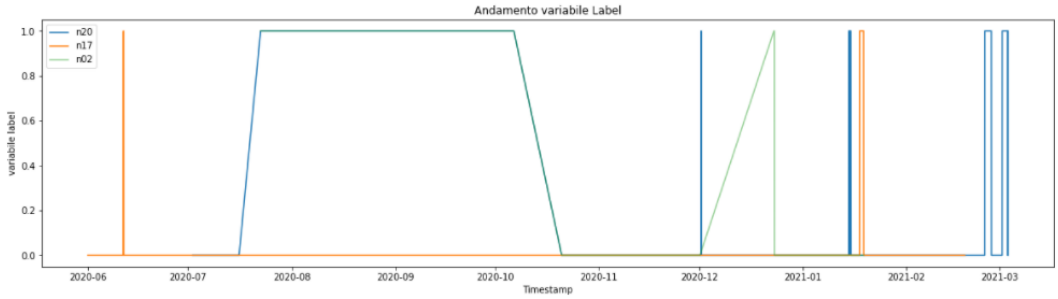

Figura 18 - andamento della variabile label per i nodi del rack 205

Come è possibile vedere dalla figura precedente, i vari nodi che appartengono a questo rack hanno, per la maggior parte delle letture, comportamenti diversi per quanto riguarda le anomalie. Solo il nodo n02, e il nodo n20 falliscono insieme, precisamente nelle letture intorno al 08/2020 e al 10/2020. Di seguito viene riportato lo zoom in questo intervallo temporale.

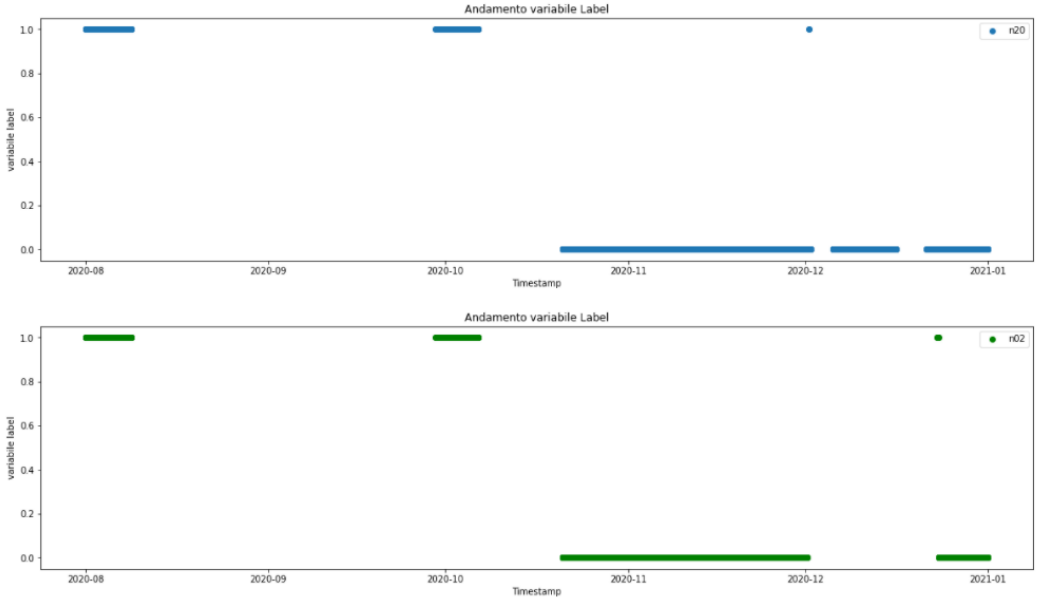

Figura 19 - zoom intervallo temporale 08/2020 - 10/2020 per i nodi 02 e 20

3.3 Il rack 206

Per il rack 206 si andranno ad analizzare i nodi 02, 14, 17, 18, 19.

3.3.1 Nodo r206n02

Le letture in questo nodo vengono riportate in una tabella di 10781 righe e 234 colonne. Il primo campionamento, risale al 2020-07-06 12:30:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 9886 letture in cui il nodo risulta in stato *normale* e 895 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

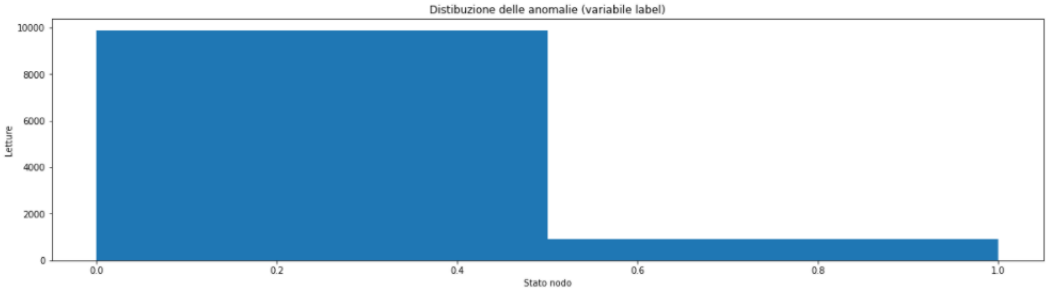

Figura 20 - distribuzione delle anomalie in base alla variabile label nodo r206n02

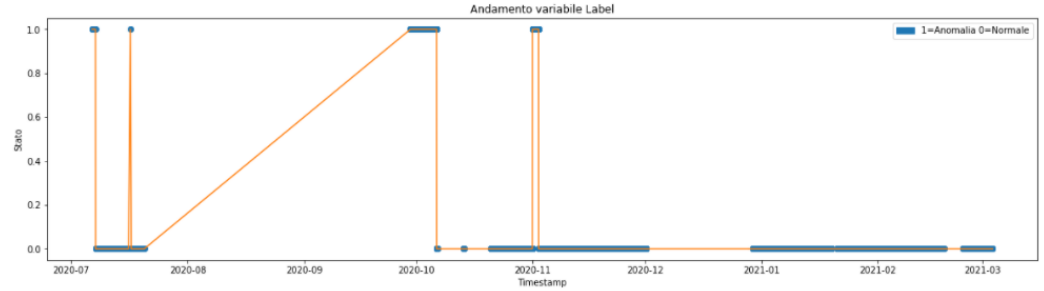

Figura 21 - andamento della variabile label nodo r206n02

Effettuando il plot relativo alle temperature dei moduli DIMM, si può notare che i loro valori variano tutti in modo simile e che non ci sono particolari correlazioni con le anomalie.

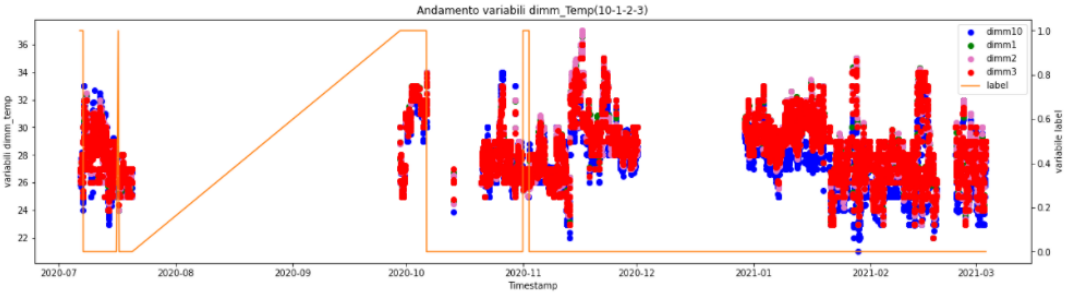

Figura 22 - andamento variabili dimm_temp nodo r206n02

Osservando l'andamento delle temperature delle GPU, si può notare che anche queste variano in modo simile tra loro.

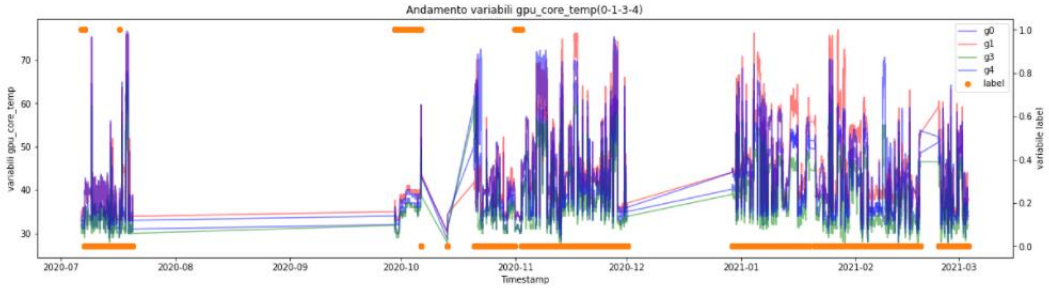

Figura 23 - andamento variabili gpu_core_temp nodo r206n02

Si è provato ad effettuare uno zoom nell'intervallo 2020/10 e 2020/12, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al suo valore solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, i risultati saranno simili.

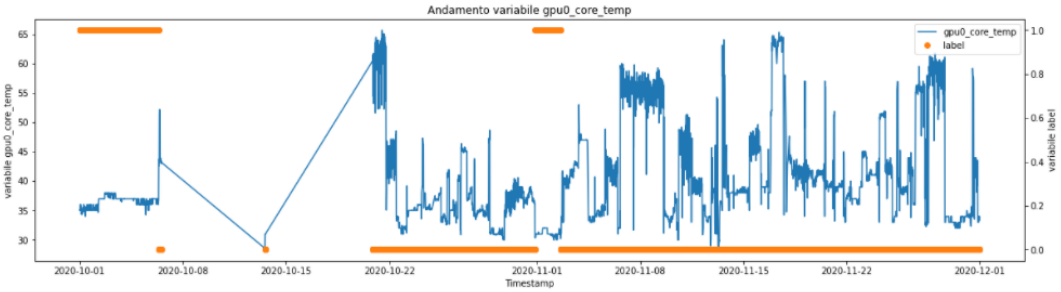

Figura 24 - andamento variabile gpu0_core_temp nodo r206n02

Se si va ad osservare la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

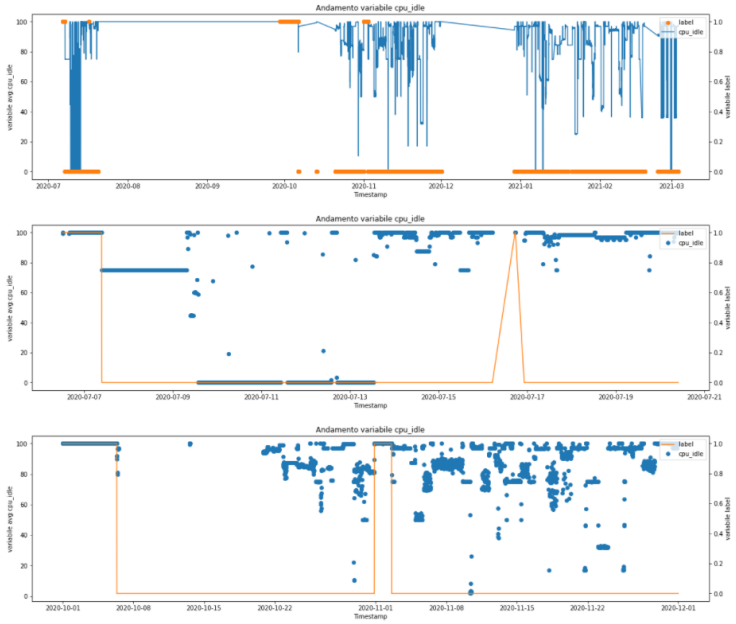

Figura 25 - andamento variabile cpu_idle nodo r206n02

Inoltre se si considera la percentuale di utilizzo della CPU che si è verificato durante l'esecuzione a livello di sistema o di utente, nel momento di anomalie questa risulta essere zero.

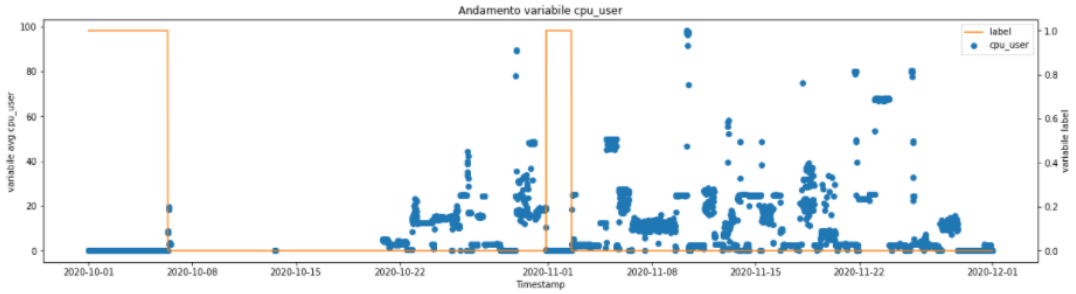

Figura 26 - andamento variabile cpu_user nodo r206n02

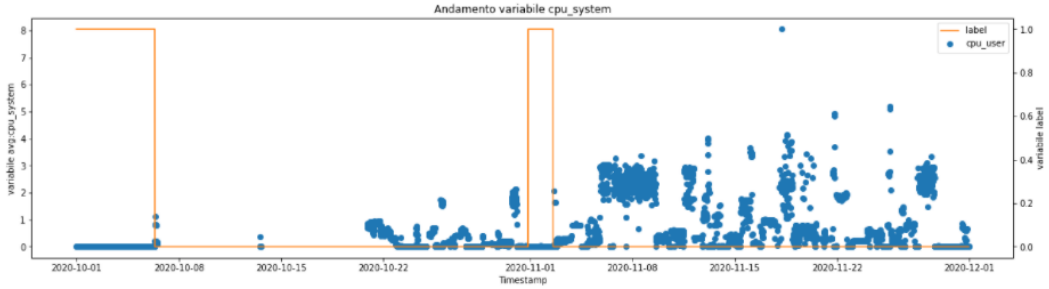

Figura 27 - andamento variabile cpu_system nodo r206n02

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

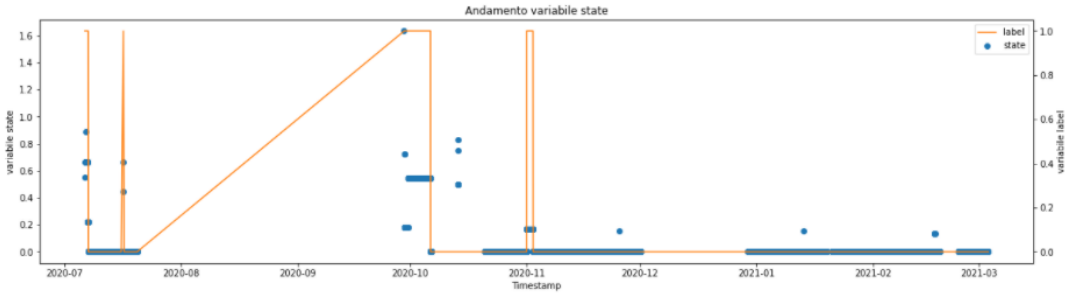

Figura 28 - andamento variabile state nodo r206n02

3.3.2 Nodo r206n14

Le letture in questo nodo vengono riportate in una tabella di 7845 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:15:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 7779 letture in cui il nodo risulta in stato *normale* e 66 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

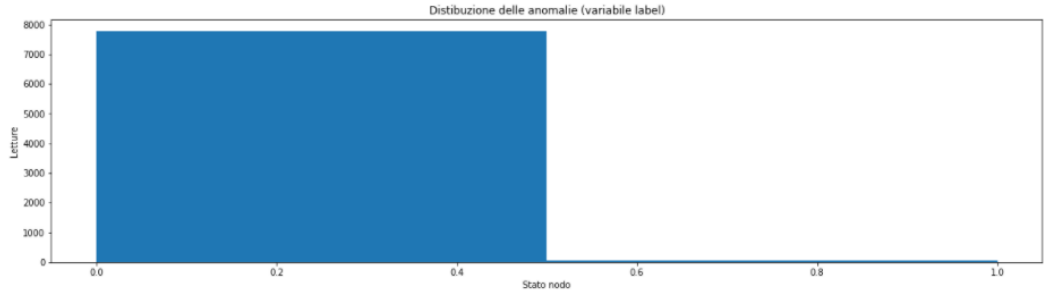

Figura 29 - distribuzione delle anomalie in base alla variabile label nodo r206n14

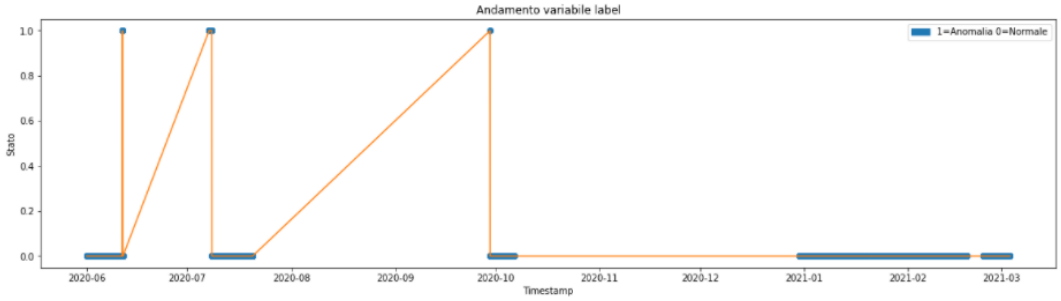

Figura 30 - andamento della variabile label nodo r206n02

Osservando l'andamento delle temperature delle GPU, si può notare che queste variano in modo simile.

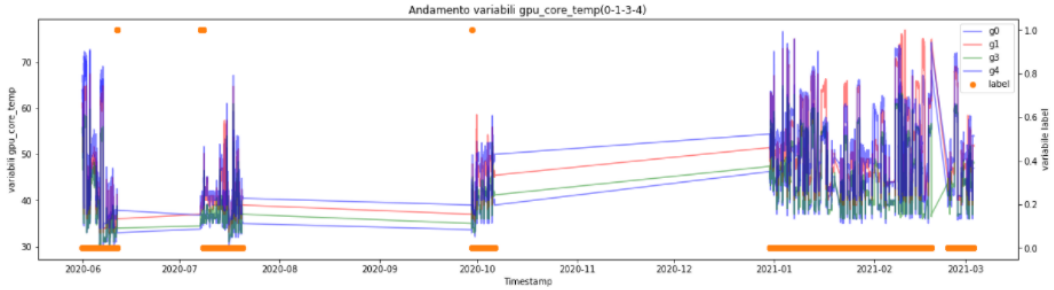

Figura 31 - andamento variabili gpu_core_temp nodo r206n14

Si è provato ad effettuare uno zoom nell'intervallo 2020/06 e 2020/08, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al suo valore solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, i risultati saranno simili.

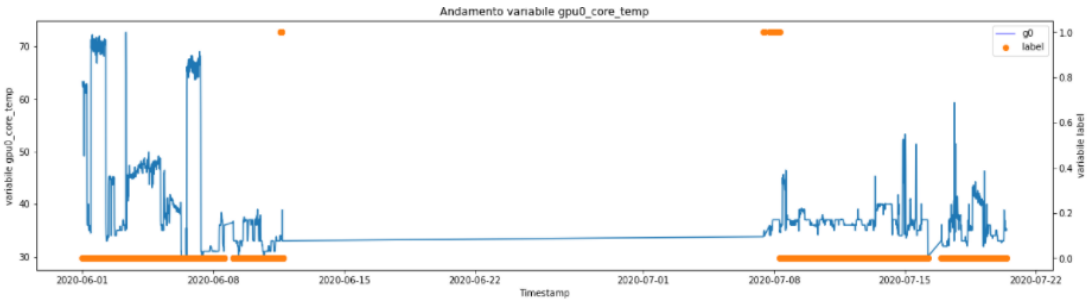

Figura 32 - andamento variabile gpu0_core_temp nodo r206n14

Se si va ad osservare la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

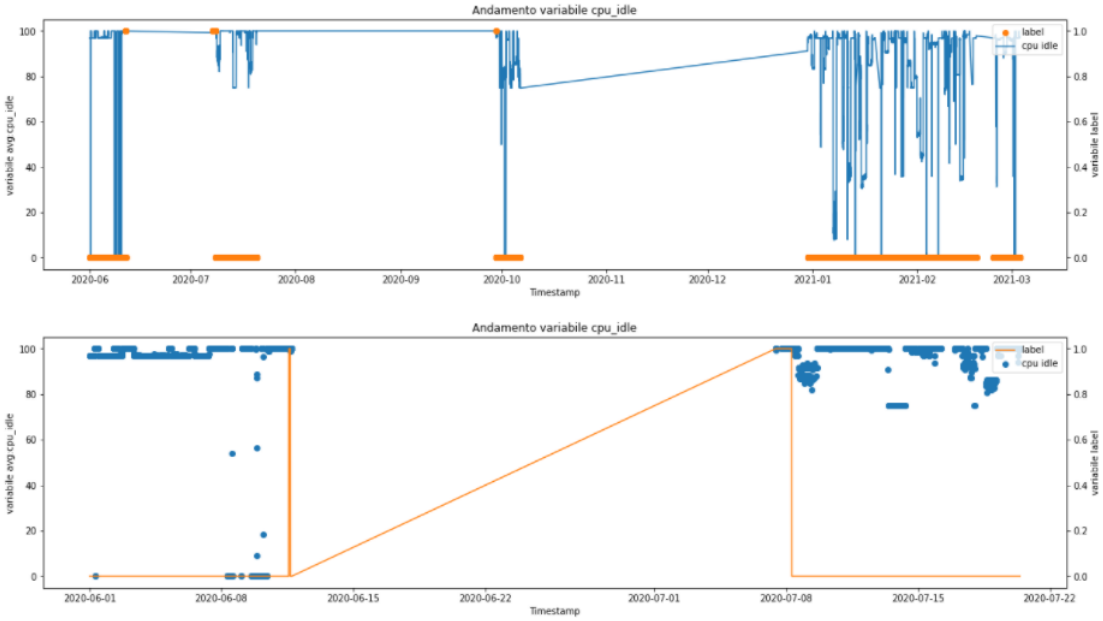

Figura 33 - andamento variabile cpu_idle nodo r206n14

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

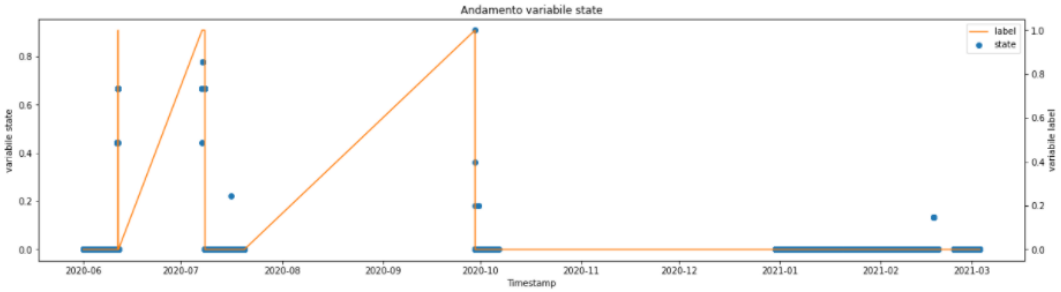

Figura 34 - andamento variabile state nodo r206n14

3.3.3 Nodo r206n17

Le letture in questo nodo vengono riportate in una tabella di 11664 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:15:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 11594 letture in cui il nodo risulta in stato *normale* e 70 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

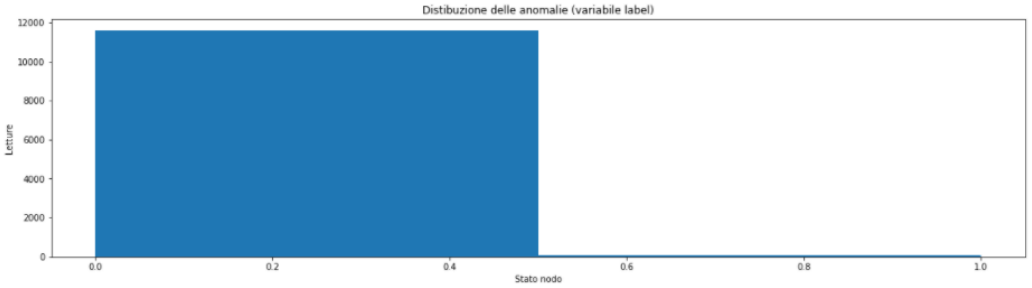

Figura 35 - distribuzione delle anomalie in base alla variabile label nodo r206n17

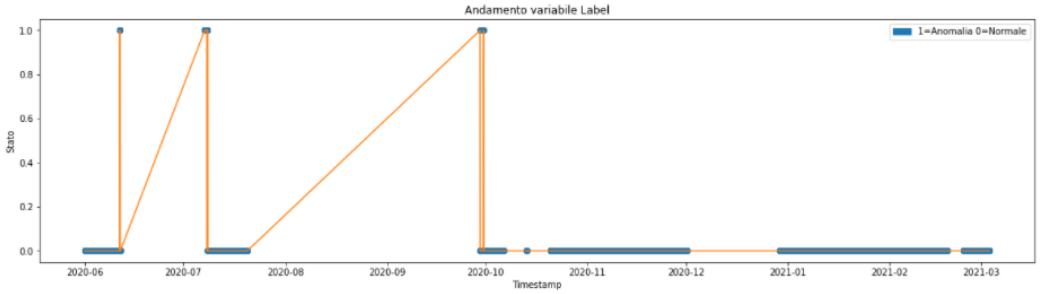

Figura 36 - andamento della variabile label nodo r206n17

Osservando l'andamento delle temperature delle GPU, si può notare che queste variano in modo simile.

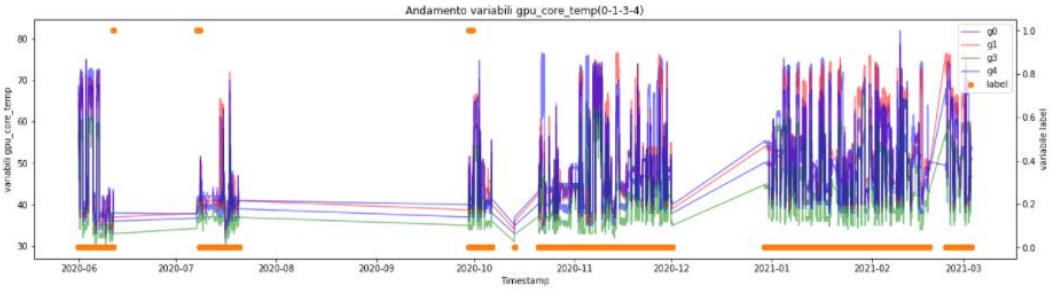

Figura 37 - andamento variabili gpu_core_temp nodo r206n17

Si è provato ad effettuare uno zoom nell'intervallo 2020/07 e 2020/08, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al suo valore solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, i risultati saranno simili.

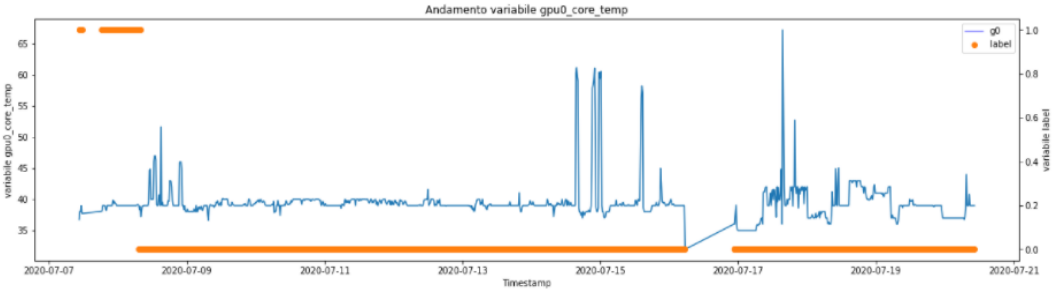

Figura 38 - andamento variabile gpu0_core_temp nodo r206n17

Se si va ad osservare la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

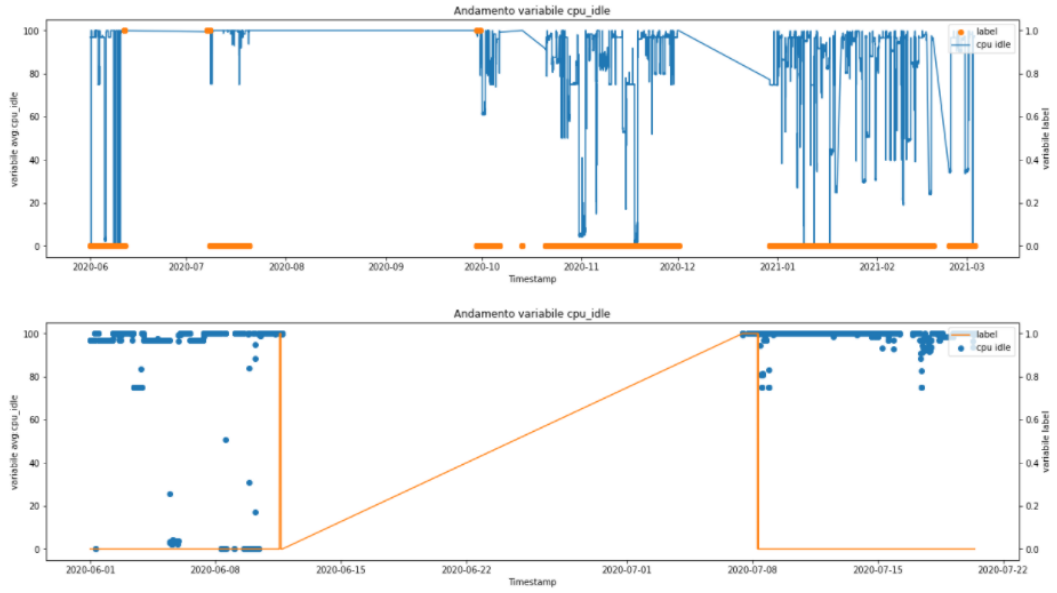

Figura 39 - andamento variabile cpu_idle nodo r206n17

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

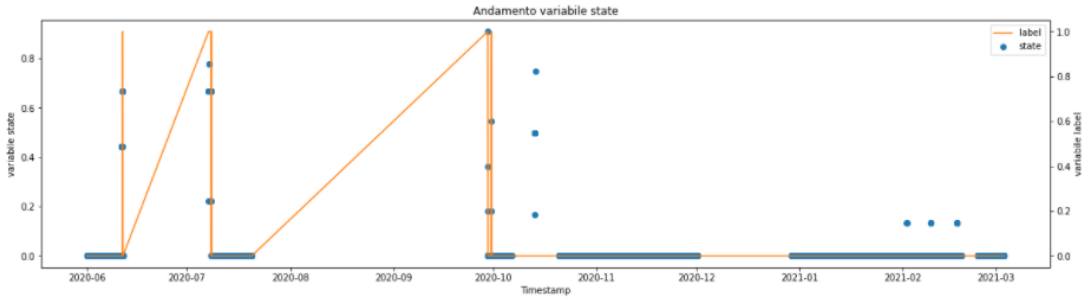

Figura 40 - andamento variabile state nodo r206n17

3.3.4 Nodo r206n18

Le letture in questo nodo vengono riportate in una tabella di 11681 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:00:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 11617 letture in cui il nodo risulta in stato *normale* e 64 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

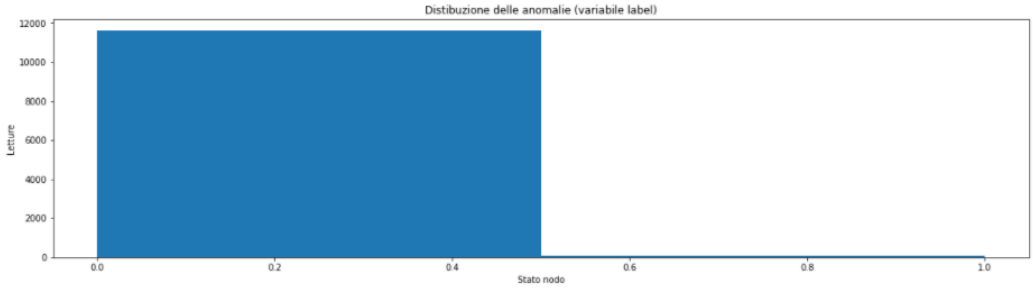

Figura 41 - distribuzione delle anomalie in base alla variabile label nodo r206n18

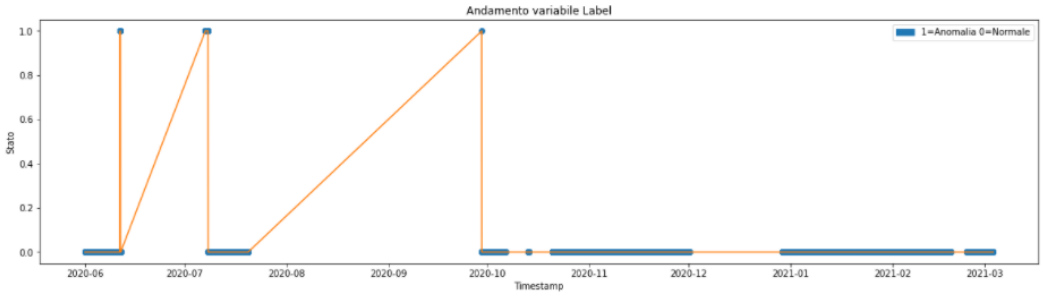

Figura 42 - andamento della variabile label nodo r206n18

Osservando l'andamento delle temperature delle GPU, si può notare che queste variano in modo simile.

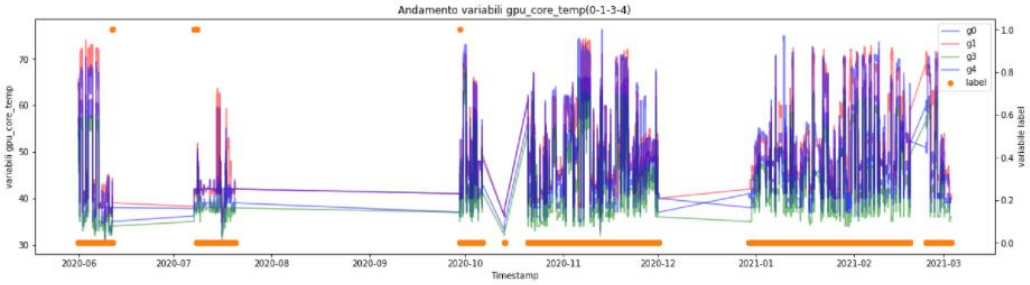

Figura 43 - andamento variabili gpu_core_temp nodo r206n18

Si è provato ad effettuare uno zoom nell'intervallo 2020/06 e 2020/08, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al suo valore solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, i risultati saranno simili.

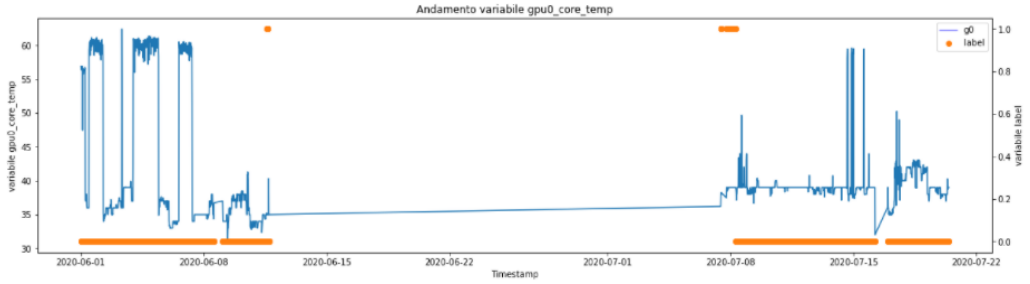

Figura 44 - andamento variabile gpu0_core_temp nodo r206n18

Se si va ad osservare la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

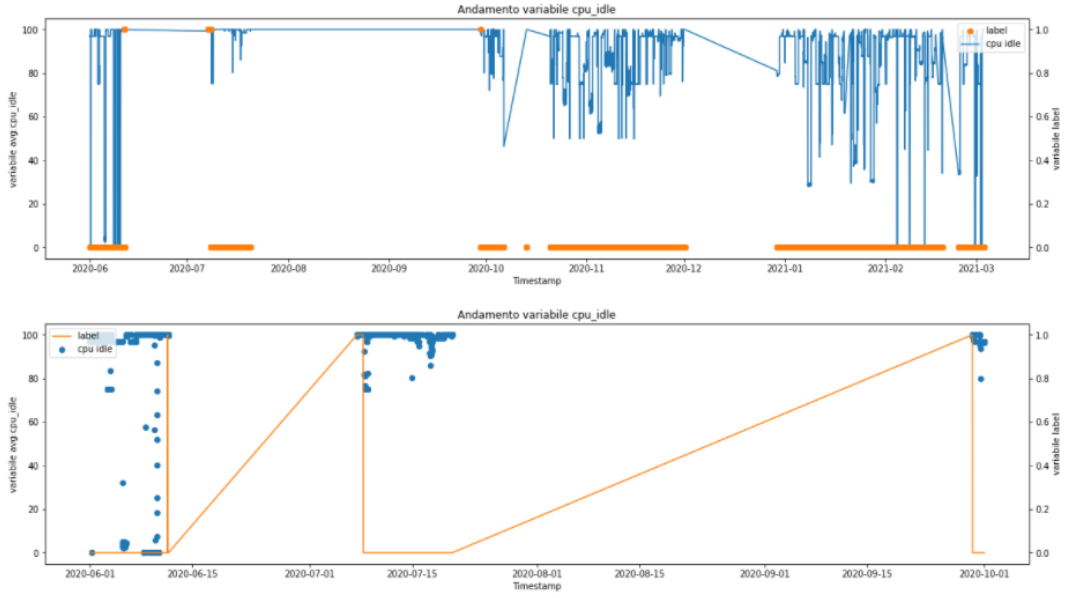

Figura 45 - andamento variabile cpu_idle nodo r206n18

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

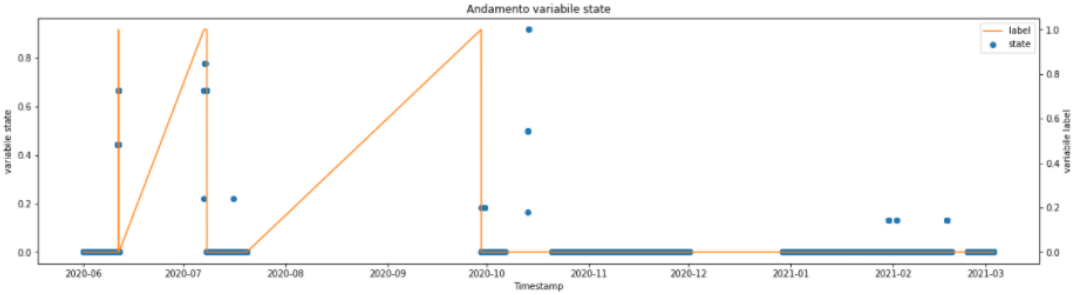

Figura 46 - andamento variabile state nodo r206n18

3.3.5 Nodo r206n19

Le letture in questo nodo vengono riportate in una tabella di 11703 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:00:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 11635 letture in cui il nodo risulta in stato *normale* e 68 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

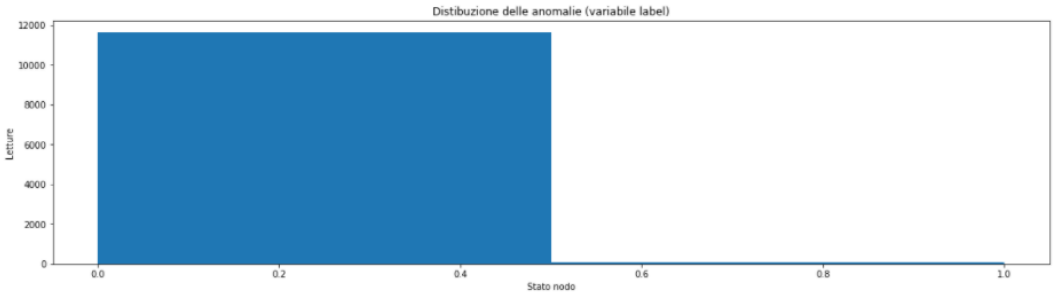

Figura 47 - distribuzione delle anomalie in base alla variabile label nodo r206n19

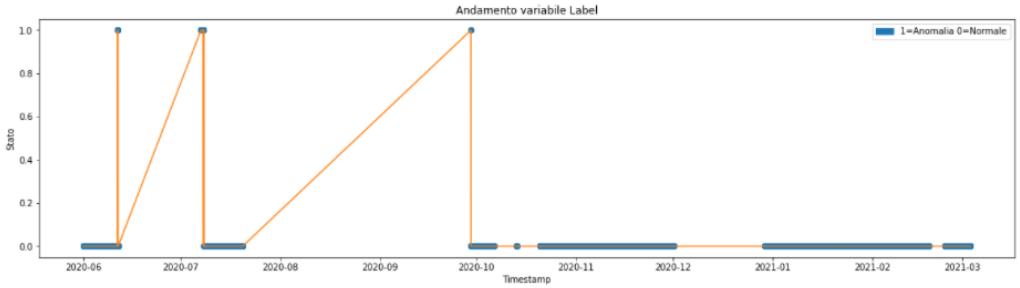

Figura 48 - andamento della variabile label nodo r206n19

Osservando l'andamento delle temperature delle GPU, si può notare che queste variano in modo simile.

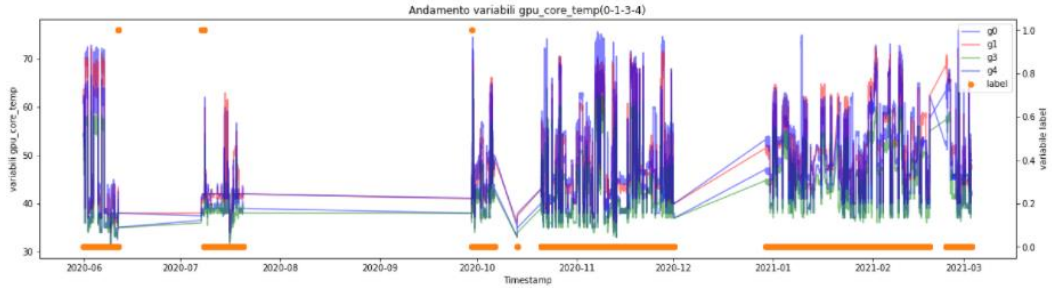

Figura 49 - andamento variabili gpu_core_temp nodo r206n19

Si è provato ad effettuare uno zoom nell'intervallo 2020/06 e 2020/08, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al suo valore solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, i risultati saranno simili.

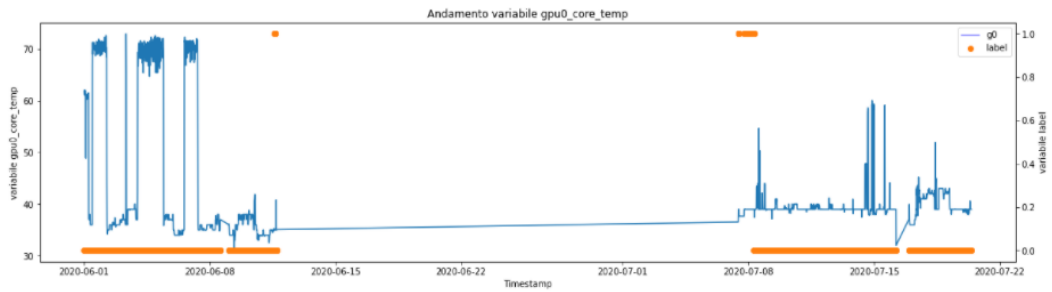

Figura 50 - andamento variabile gpu0_core_temp nodo r206n19

Se si va ad osservare la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

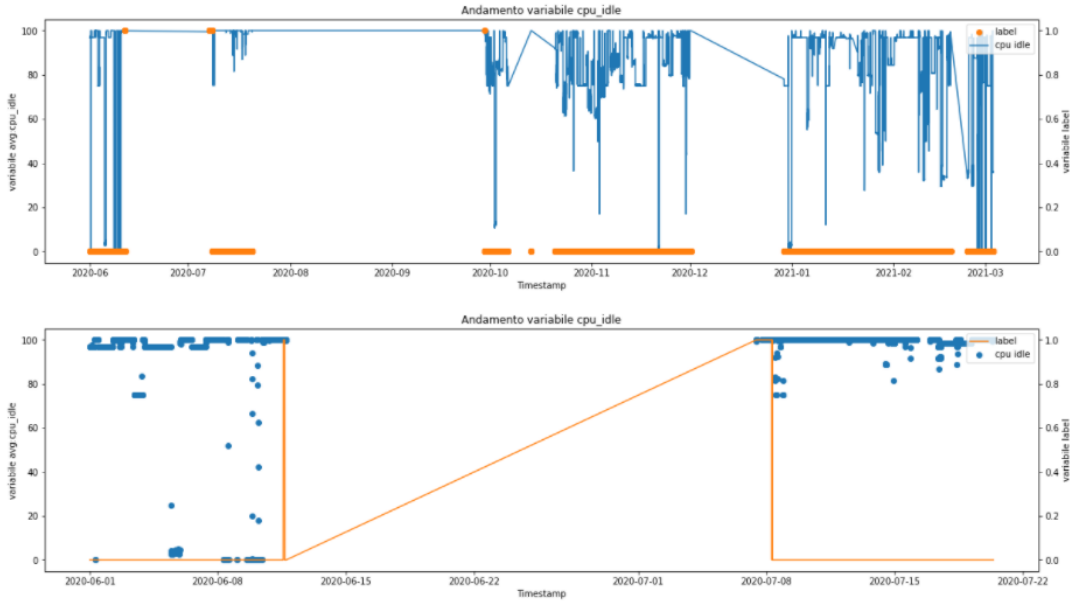

Figura 51 - andamento variabile cpu_idle nodo r206n19

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

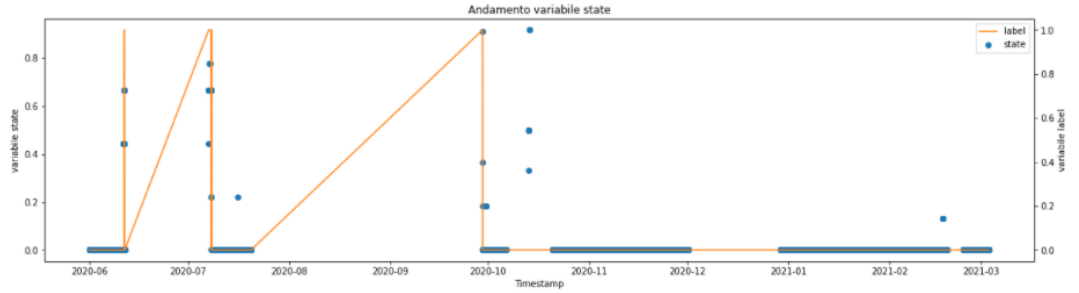

Figura 52 - andamento variabile state nodo r206n19

3.3.6 Confronto tra i nodi del rack 206

	Shape	Primo camp.	Ultimo camp.	Stato anomalo	Salti	Numero anomalie	Media anomalie
n02	(10781, 234)	2020-07-06 12:30:00	2021-03-03 13:45:00	8%	Si	4	57 h e 30 min
n14	(7845, 234)	2020-05-31 22:15:00	2021-03-03 13:45:00	0.8%	Si	3	7 h 25min
n17	(11664, 234)	2020-05-31 22:15:00	2021-03-03 13:45:00	0.6%	Si	4	6 h
n18	(11681, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	0.5%	Si	4	3 h 45 min
n19	(11703, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	0.6%	Si	4	4 h

Per poter capire se i nodi falliscono insieme o in momenti distinti, si è pensato di visualizzare lo stato della variabile label per tutti i nodi del rack in un'unica figura.

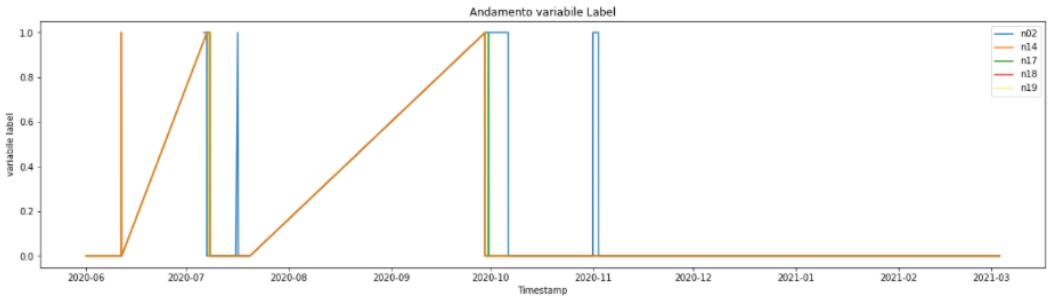

Figura 53 - andamento della variabile label per i nodi del rack 206

Come è possibile vedere dalla figura precedente, possiamo riscontrare che la maggior parte delle anomalie avviene nello stesso intervallo di tempo o in un intorno di esso. Solo il nodo n02 si distacca leggermente dal comportamento degli altri nodi. Di seguito viene riportato lo zoom per i nodi 14,17,18,19 per evidenziare questo comportamento.

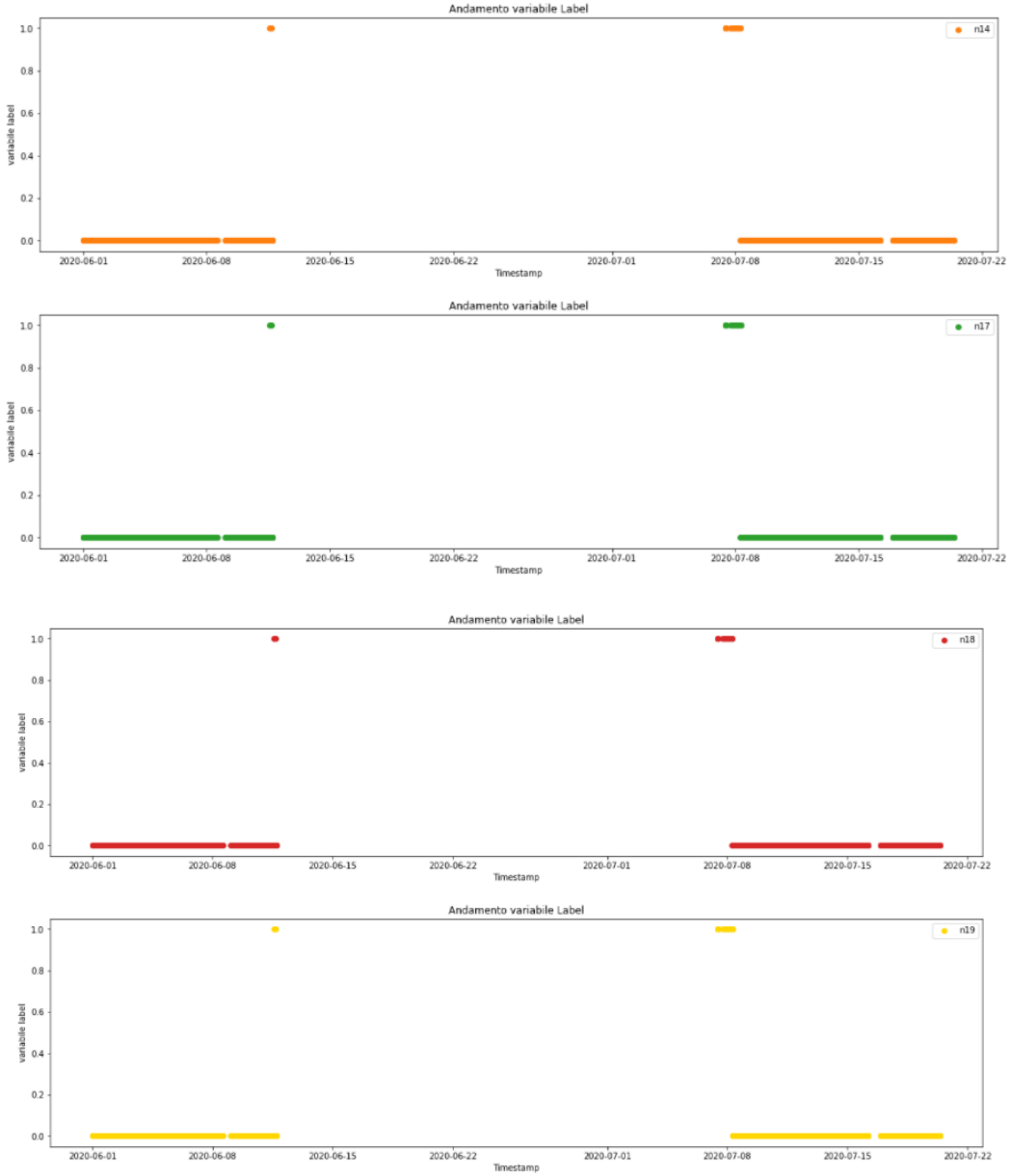

Figura 54 - zoom andamento variabile label per i nodi 14,17,18,19

3.4 Il rack 208

Per il rack 208 si andranno ad analizzare i nodi 02, 04, 07, 12.

3.4.1 Nodo r208n02

Le letture in questo nodo vengono riportate in una tabella di 11382 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:00:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 11365 letture in cui il nodo risulta in stato *normale* e 17 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

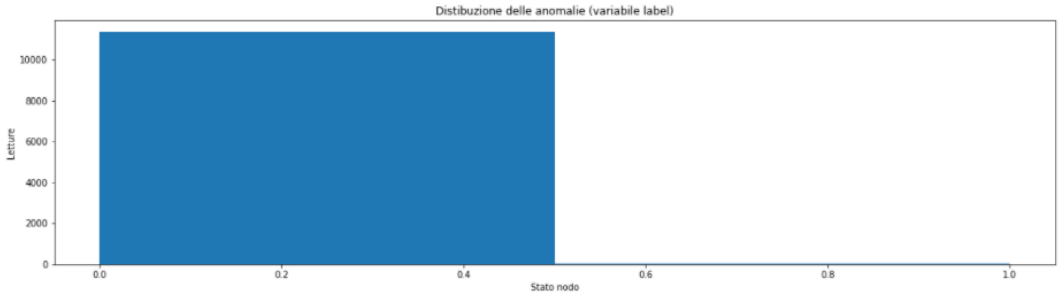

Figura 55 - distribuzione delle anomalie in base alla variabile label nodo r208n02

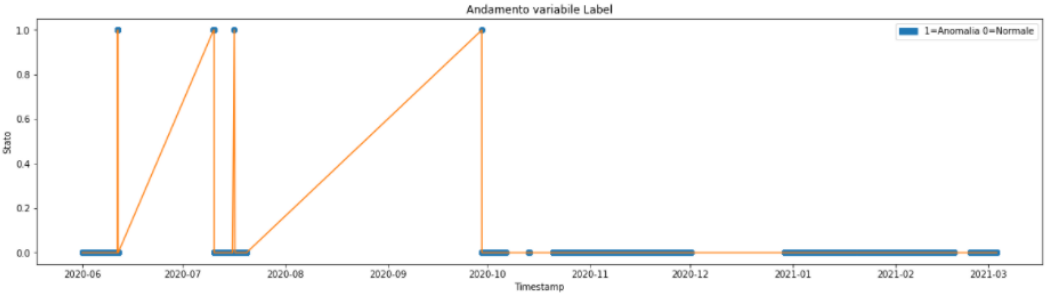

Figura 56 - andamento della variabile label nodo r208n02

Effettuando il plot relativo alle temperature dei moduli DIMM, si può notare che i loro valori variano tutti in modo simile e che non ci sono particolari correlazioni con le anomalie.

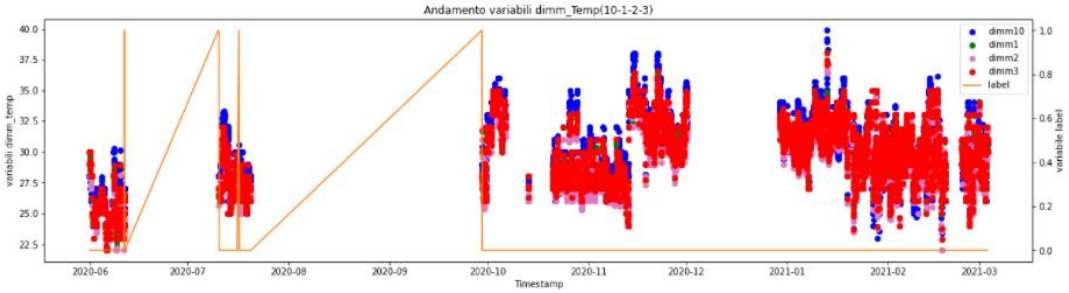

Figura 57 - andamento variabili dimm_temp nodo r208n02

Osservando l'andamento delle temperature delle GPU, si può notare che queste variano in modo simile.

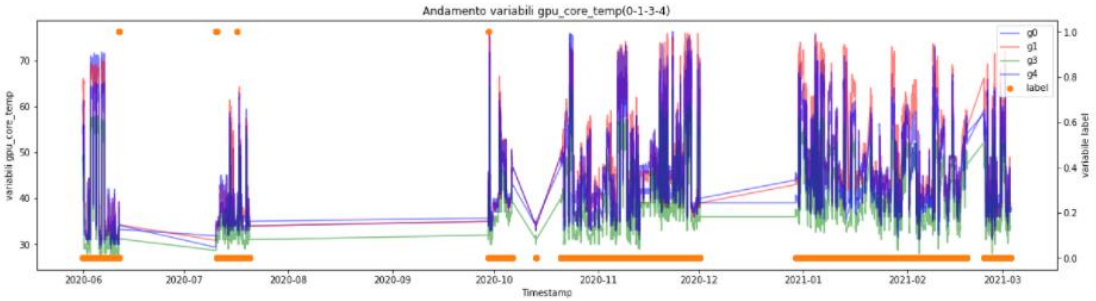

Figura 58 - andamento variabili gpu_core_temp nodo r208n02

Si è provato ad effettuare uno zoom nell'intervallo 2020/06 e 2020/08, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al suo valore solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, i risultati saranno simili.

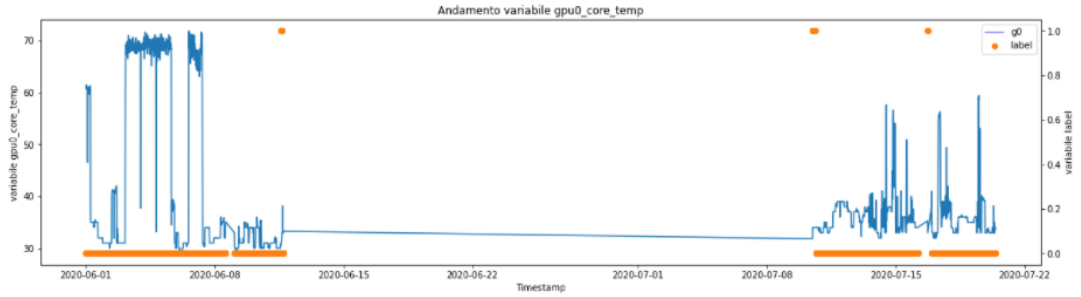

Figura 59 - andamento variabile gpu0_core_temp nodo r208n02

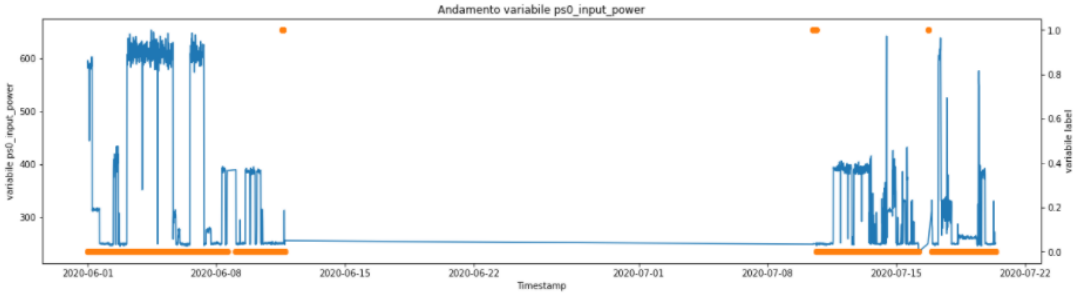

Figura 60 - andamento variabile ps0_input_power nodo r208n02

Se si va ad osservare la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

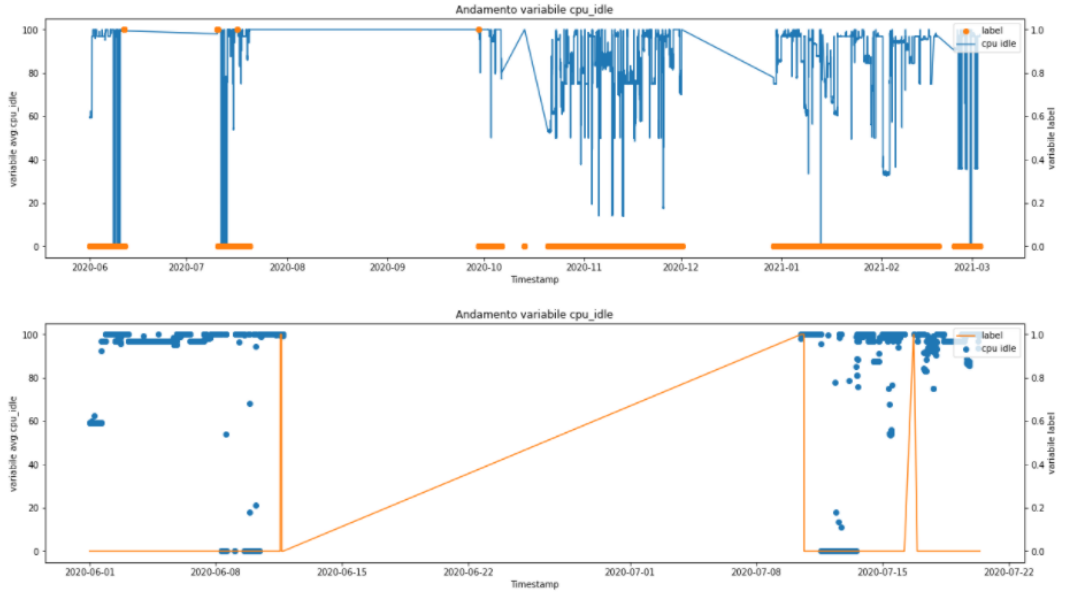

Figura 61 - andamento variabile cpu_idle nodo r208n02

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

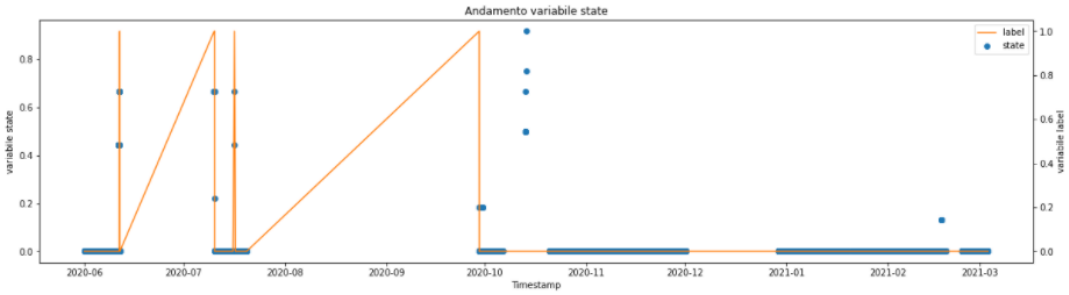

Figura 62 - andamento variabile state nodo r208n02

3.4.2 Nodo r208n04

Le letture in questo nodo vengono riportate in una tabella di 6114 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:00:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 6107 letture in cui il nodo risulta in stato *normale* e 7 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

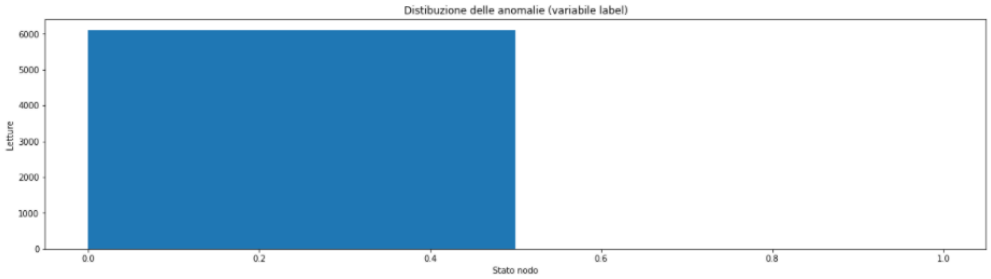

Figura 63 - distribuzione delle anomalie in base alla variabile label nodo r208n04

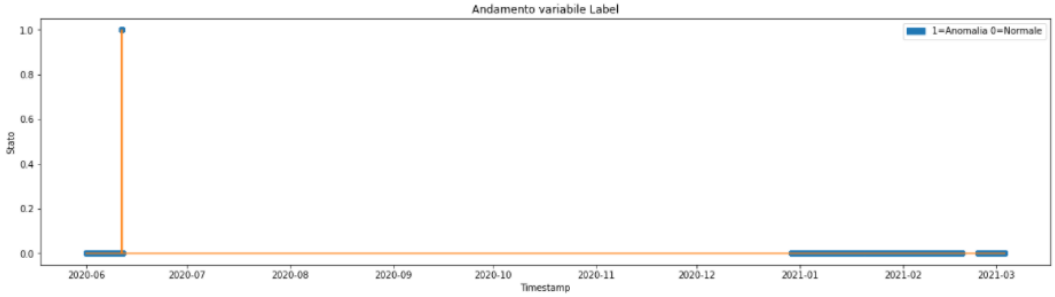

Figura 64 - andamento della variabile label nodo r208n04

Se si va ad osservare la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

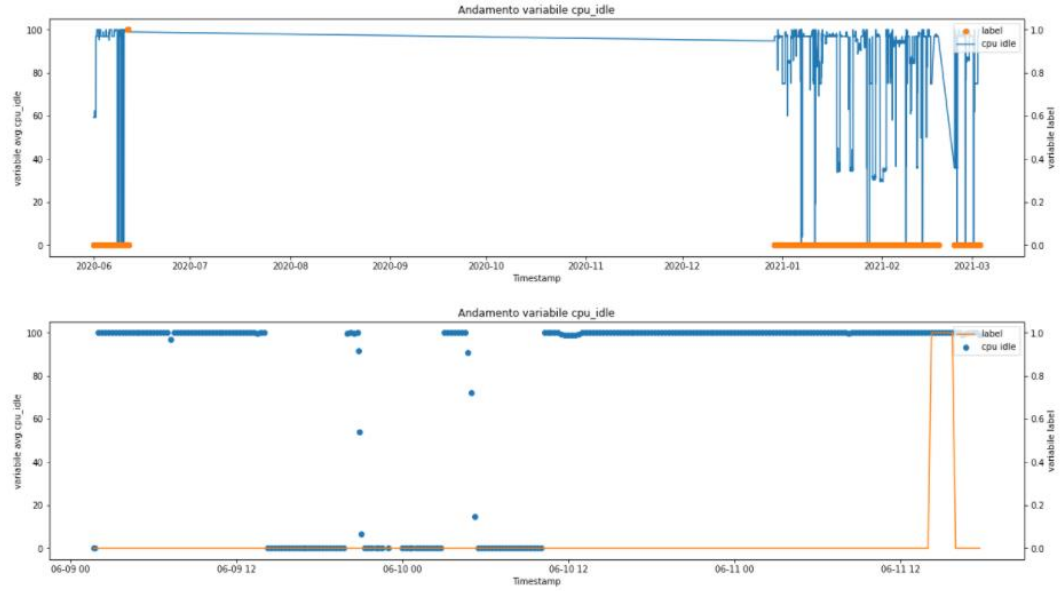

Figura 65 - andamento variabile cpu_idle nodo r208n04

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

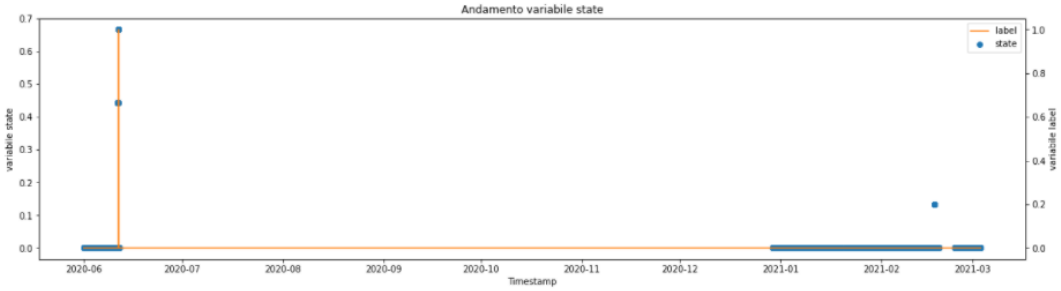

Figura 66 - andamento variabile state nodo r208n04

3.4.3 Nodo r208n07

Le letture in questo nodo vengono riportate in una tabella di 10405 righe e 234 colonne. Il primo campionamento, risale al 2020-07-10 11:30:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 10393 letture in cui il nodo risulta in stato *normale* e 12 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

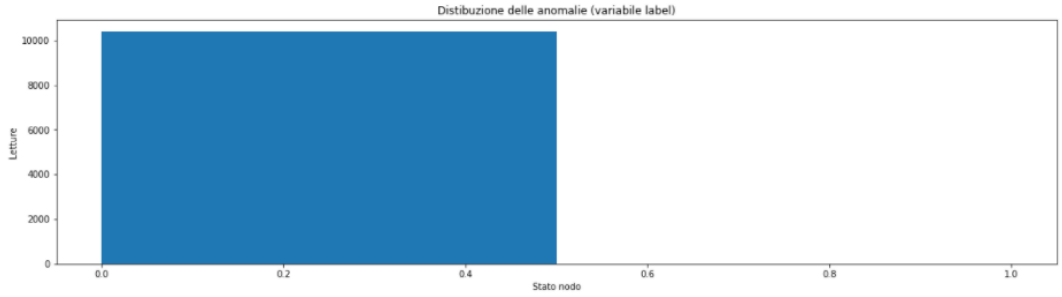

Figura 67 - distribuzione delle anomalie in base alla variabile label nodo r208n07

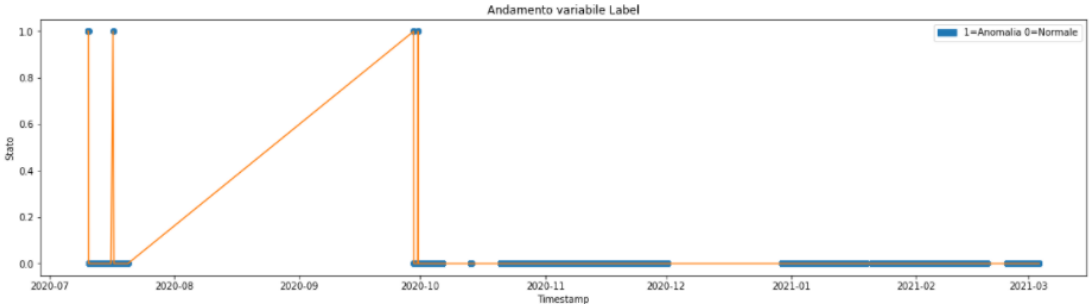

Figura 68 - andamento della variabile label nodo r208n07

Osservando l'andamento delle temperature delle GPU, si può notare che queste variano in modo simile.

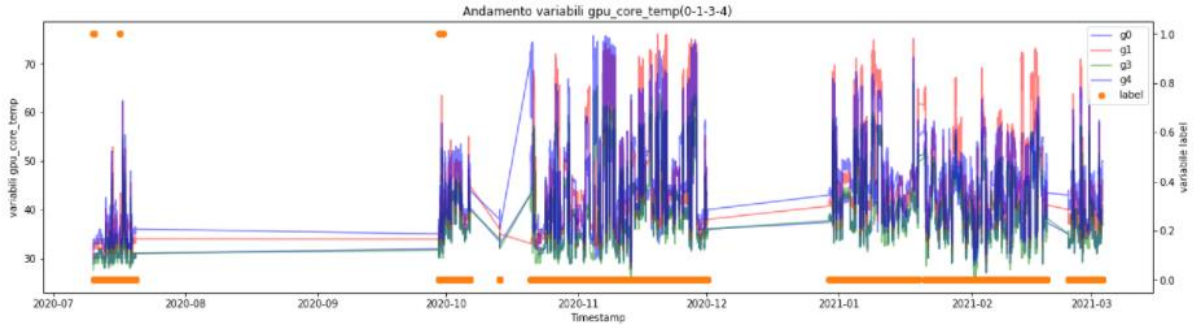

Figura 69 - andamento variabili gpu_core_temp nodo r208n07

Si è provato ad effettuare uno zoom nell'intervallo 2020/07 e 2020/08, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al suo valore solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, i risultati saranno simili.

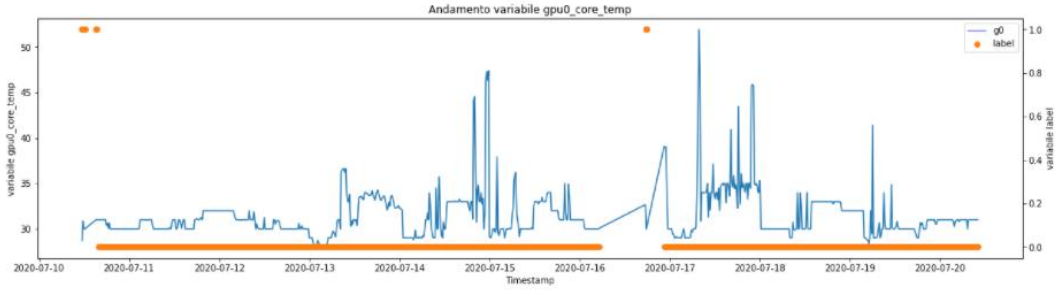

Figura 70 - andamento variabile gpu0_core_temp nodo r208n07

Se si va ad osservare la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

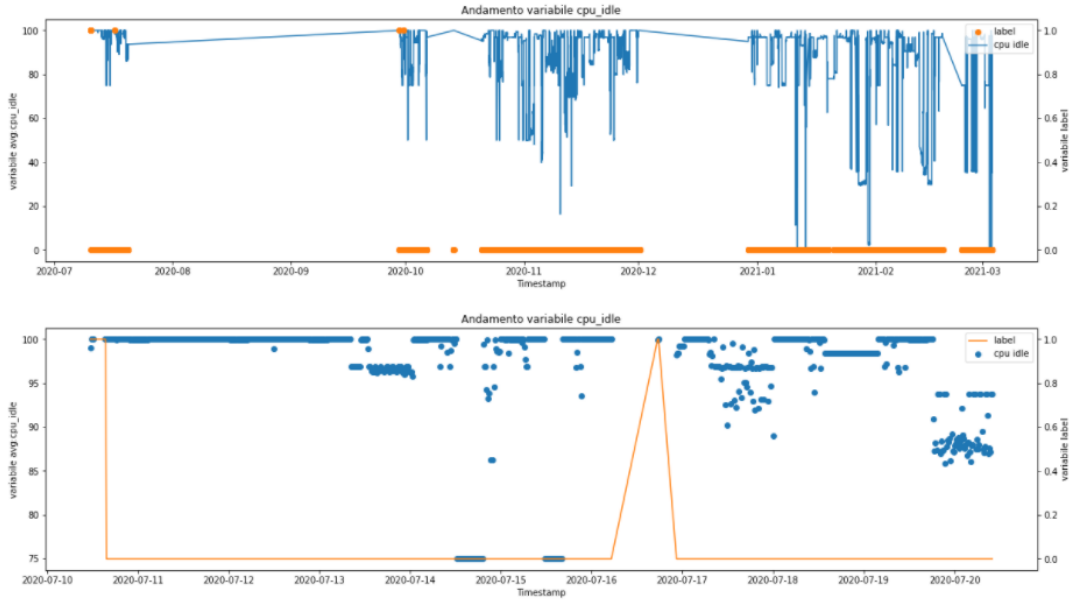

Figura 71 - andamento variabile cpu_idle nodo r208n07

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

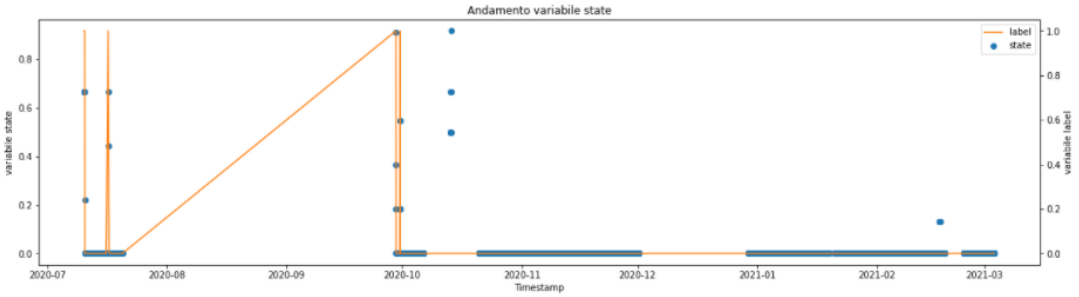

Figura 72 - andamento variabile state nodo r208n07

3.4.4 Nodo r208n12

Le letture in questo nodo vengono riportate in una tabella di 13546 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:15:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 13520 letture in cui il nodo risulta in stato *normale* e 26 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

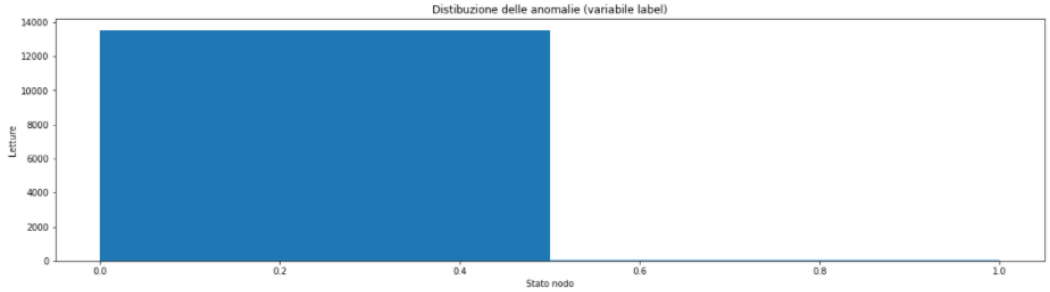

Figura 73 - distribuzione delle anomalie in base alla variabile label nodo r208n14

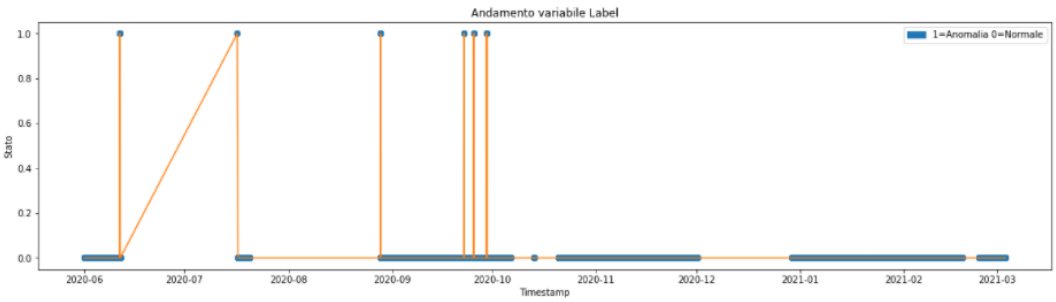

Figura 74 - andamento della variabile label nodo r208n14

Osservando l'andamento delle temperature delle GPU, si può notare che queste variano in modo simile.

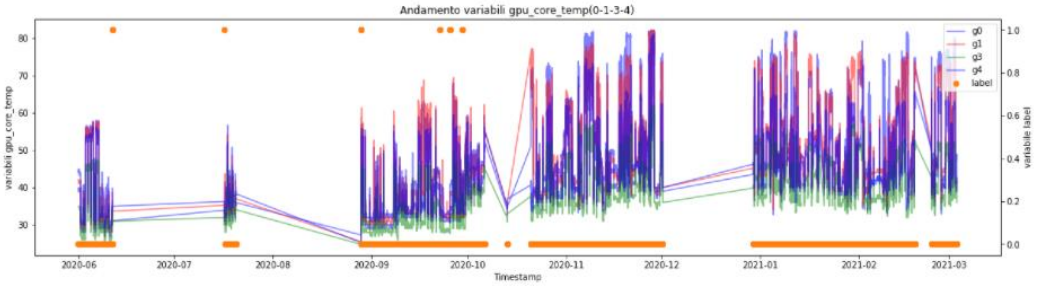

Figura 75 - andamento variabili gpu_core_temp nodo r208n14

Si è provato ad effettuare uno zoom nell'intervallo 2020/09 e 2020/10, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al suo valore solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, i risultati saranno simili.

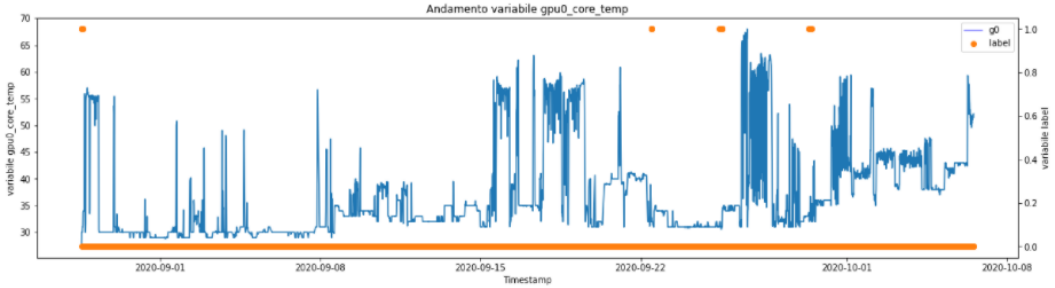

Figura 76 - andamento variabile gpu0_core_temp nodo r208n14

Se si va ad osservare la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

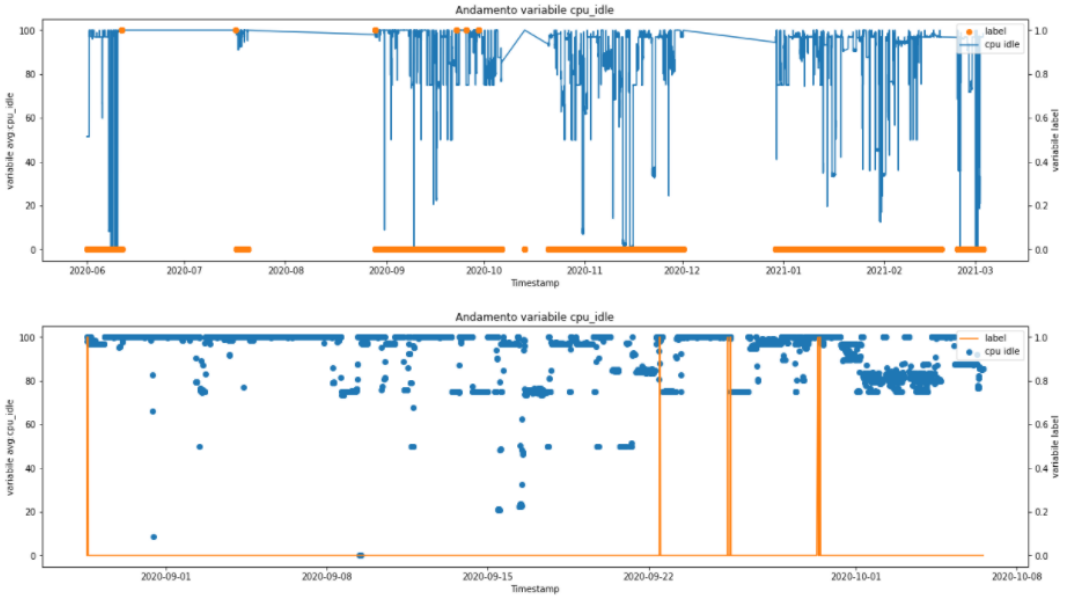

Figura 77 - andamento variabile cpu_idle nodo r208n14

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

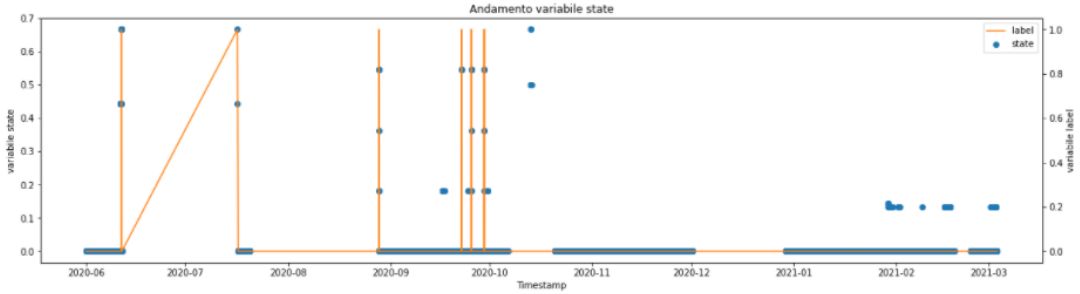

Figura 78 - andamento della variabile state nodo r208n14

3.3.5 Confronto tra i nodi del rack 208

	Shape	Primo camp.	Ultimo camp.	Stato anomalo	Salti	Numero anomalie	Media anomalie
n02	(11382, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	0.1%	Si	5	36 min
n04	(6114, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	0.1%	Si	1	1h 30 min
n07	(10405, 234)	2020-07-10 11:30:00	2021-03-03 13:45:00	0.1%	Si	5	21 min
n12	(13546, 234)	2020-05-31 22:15:00	2021-03-03 13:45:00	0.2%	Si	8	34 min

Per poter capire se i nodi falliscono insieme o in momenti distinti, si è pensato di visualizzare lo stato della variabile label per tutti i nodi del rack in un'unica figura.

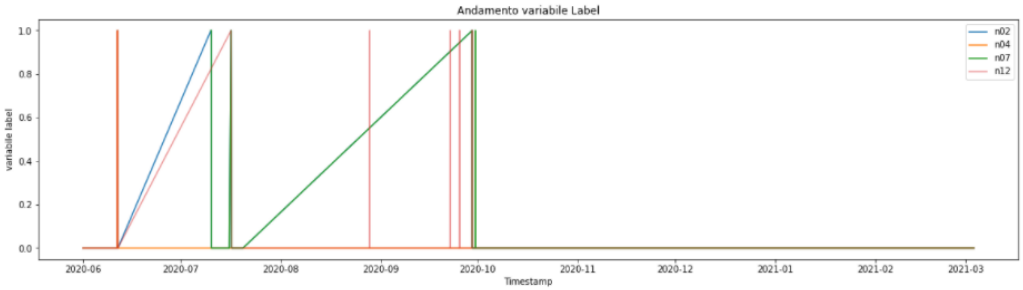

Figura 79 - andamento della variabile label per i nodi del rack 208

Come è possibile vedere dalla figura precedente, possiamo riscontrare che la maggior parte delle anomalie avviene nello stesso intervallo di tempo o in un intorno di esso. Solo il nodo n12 aggiunge altri stati anomali. Di seguito viene riportato lo zoom dell'andamento della variabile label per tutti i nodi, per evidenziare questo comportamento.

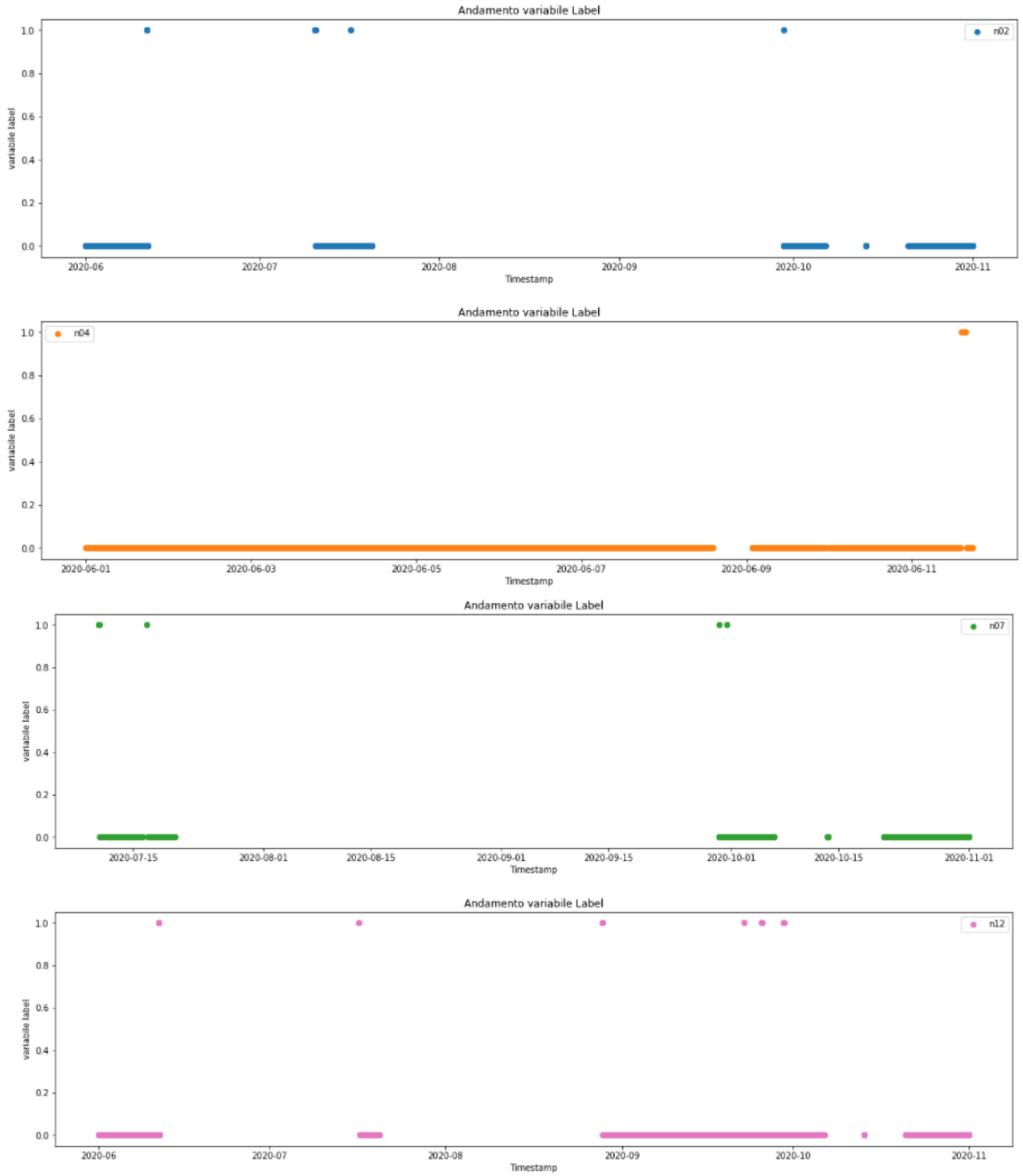

Figura 80 - zoom andamento variabile label per il rack 208

3.5 Nodo r210n05

Le letture in questo nodo vengono riportate in una tabella di 13667 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:15:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 8302 letture in cui il nodo risulta in stato *normale* e 5365 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

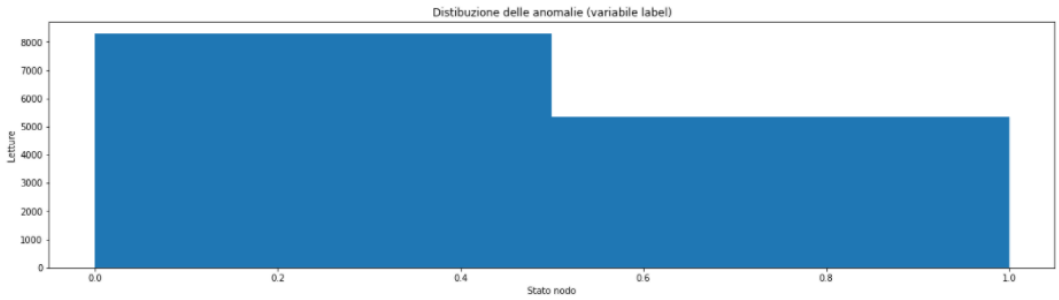

Figura 81 - distribuzione delle anomalie in base alla variabile label nodo r210n05

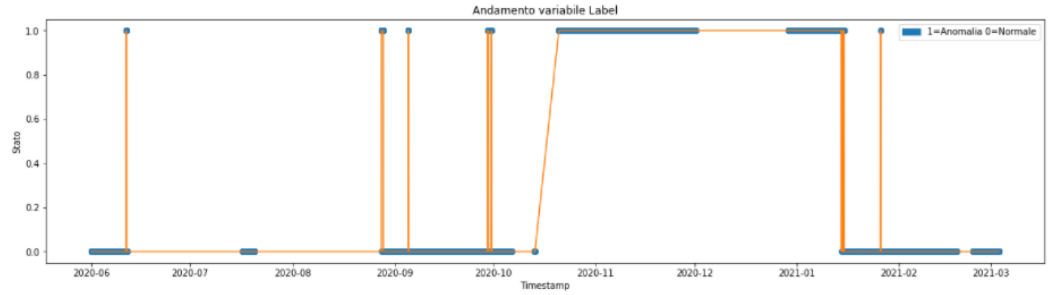

Figura 82 - andamento della variabile label nodo r210n05

Osservando l'andamento delle temperature delle GPU, si può notare che queste variano in modo simile.

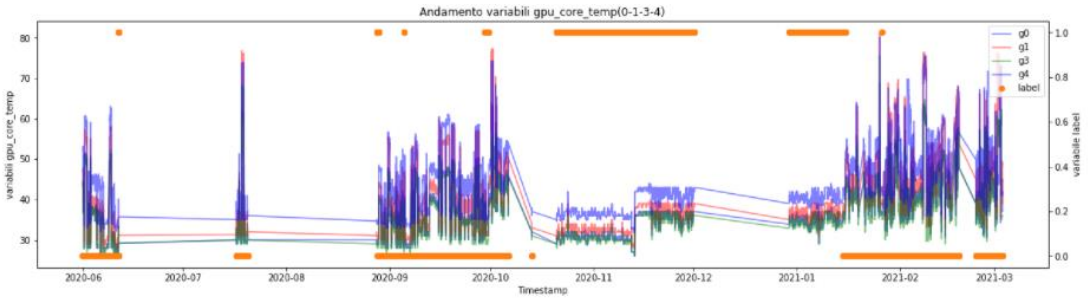

Figura 83 -andamento variabili gpu_core_temp nodo r210n05

Si è provato ad effettuare uno zoom nell'intervallo 2020/10 e 2021/02, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al suo valore solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, i risultati saranno simili.

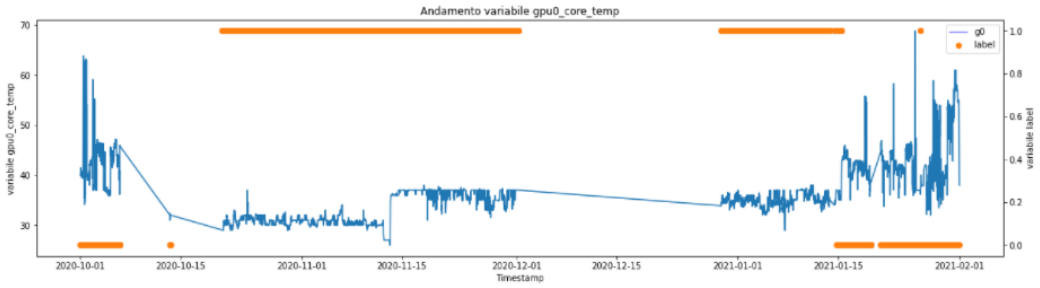

Figura 84 - andamento variabile gpu0_core_temp nodo r210n05

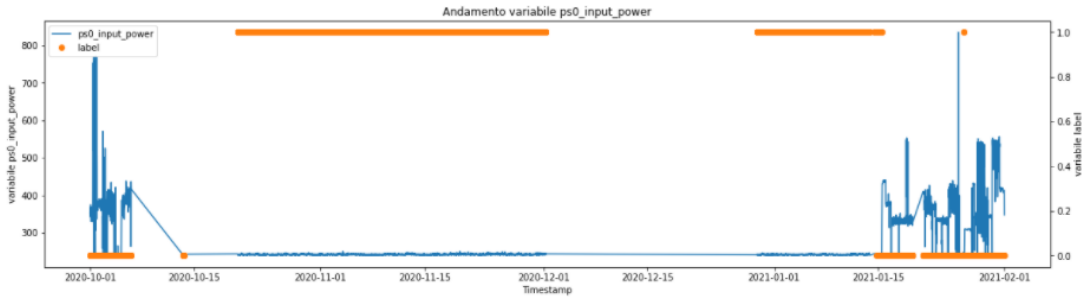

Figura 85 - andamento variabile ps0_input_power nodo r210n05

Osservando la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

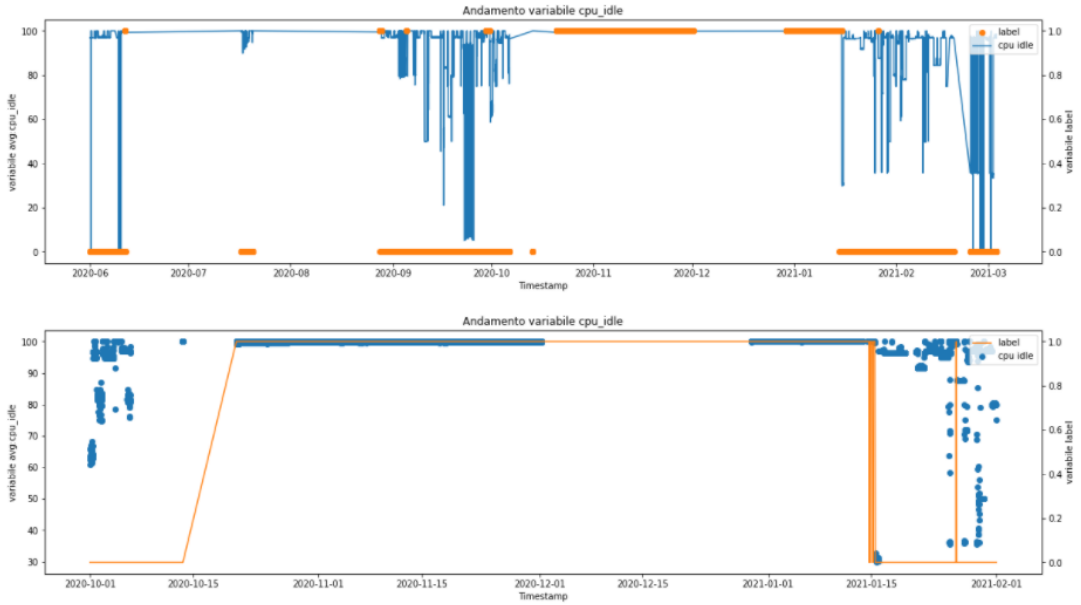

Figura 86 - andamento variabile cpu_idle nodo r210n05

Facendo il plot della metrica fan0_0, si può notare che nei periodi anomali, il valore resta più o meno costante e relativamente basso.

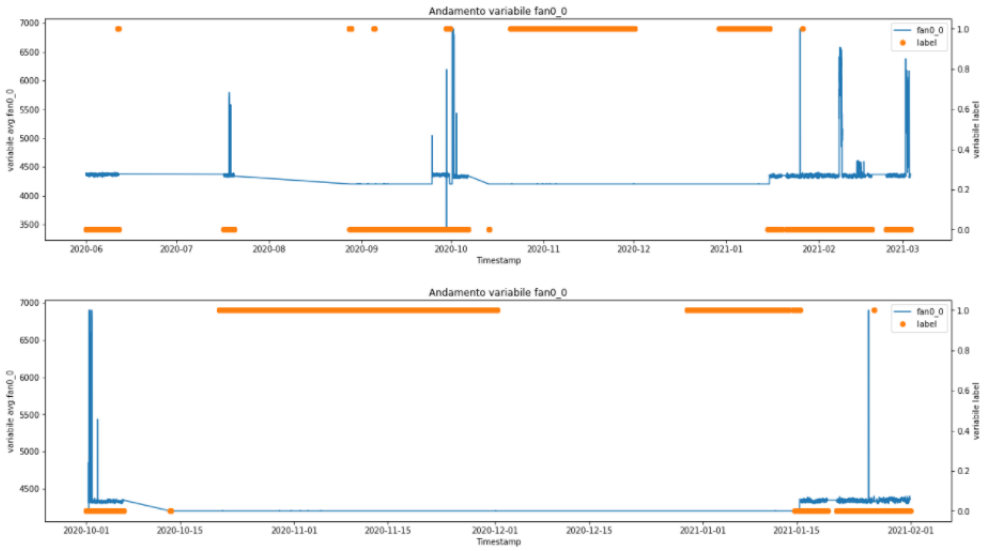

Figura 87 - andamento variabile fan0_0 nodo r210n05

Anche il consumo di energia, nei periodi anomali, ha valori che restano più o meno costanti e relativamente bassi.

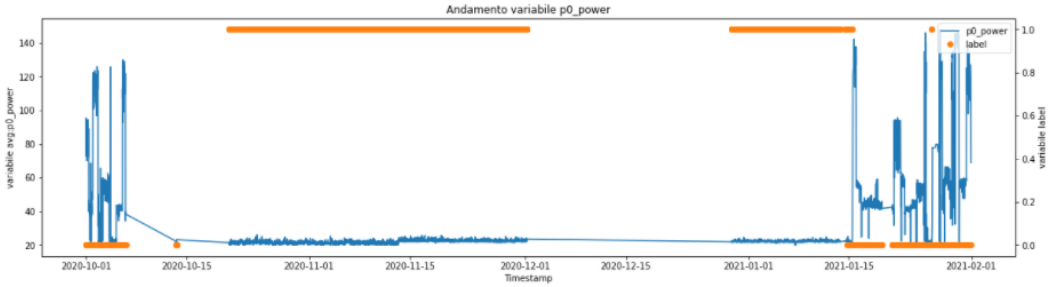

Figura 88 - andamento variabile p0_power nodo r210n05

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

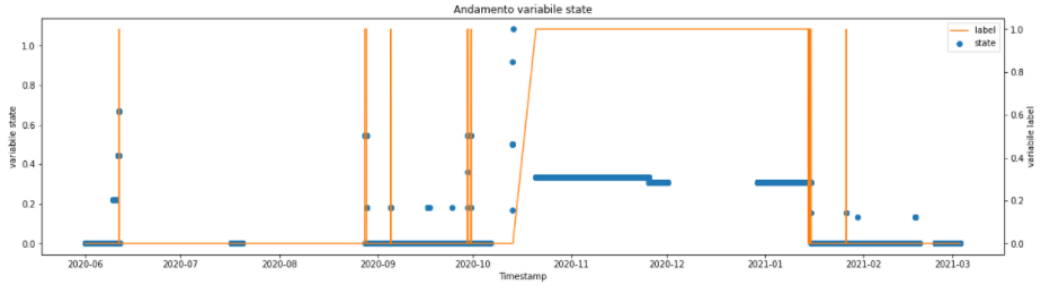

Figura 89 - Andamento variabile state nodo r210n05

Nella tabella seguente è possibile osservare le informazioni del nodo più di rilievo.

	Shape	Primo camp.	Ultimo camp.	Stato anomalo	Salti	Numero anomalie	Media anomalie
r210n05	(13667, 234)	2020-05-31 22:15:00	2021-03-03 13:45:00	39%	Si	27	51 h

3.6 Nodo r211n19

Le letture in questo nodo vengono riportate in una tabella di 6670 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:15:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 5588 letture in cui il nodo risulta in stato *normale* e 1082 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

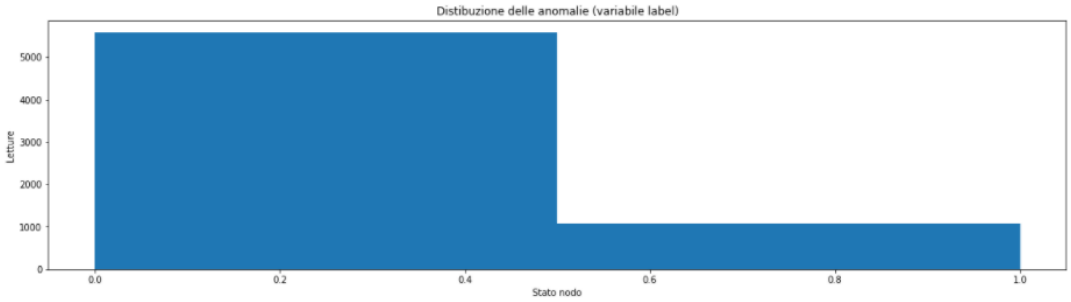

Figura 90 - distribuzione delle anomalie in base alla variabile label nodo r211n19

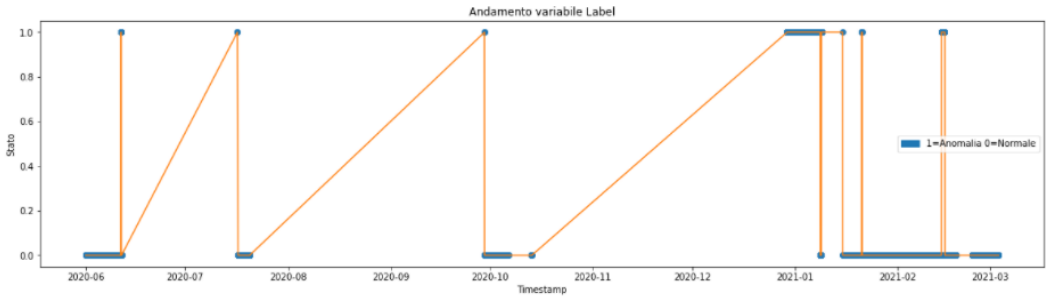

Figura 91 - andamento della variabile label nodo r211n19

Osservando l'andamento delle temperature delle GPU, si può notare che queste variano in modo simile.

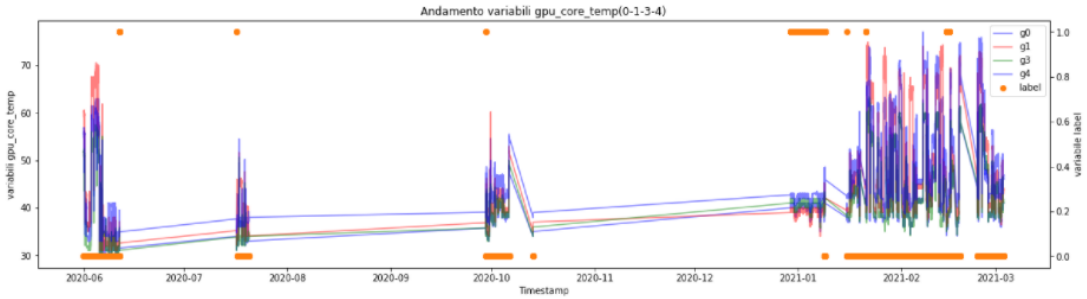

Figura 92 - andamento variabili gpu_core_temp nodo r211n19

Si è provato ad effettuare uno zoom nell'intervallo 2020/10 e 2021/02, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al suo valore solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le

temperature e la potenza dei core nelle CPU socket, velocità dei fan e energia consumata i risultati saranno simili.

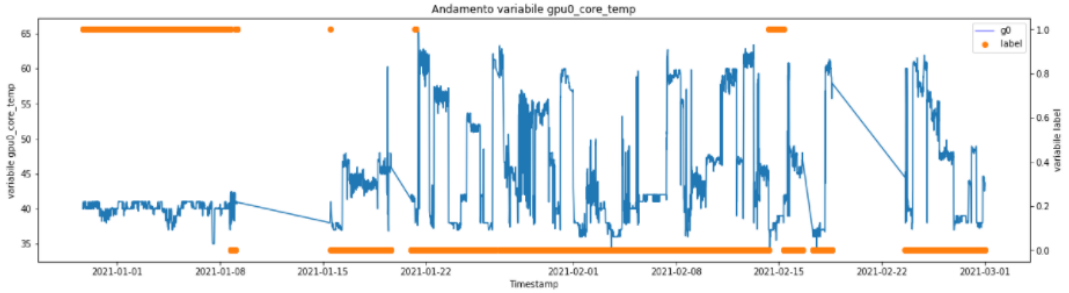

Figura 93 - andamento variabile gpu0_core_temp nodo r211n19

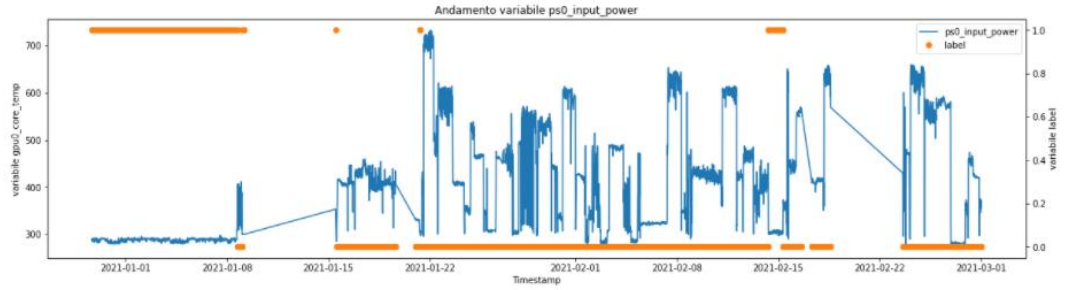

Figura 94 - andamento variabile ps0_input_power nodo r211n19

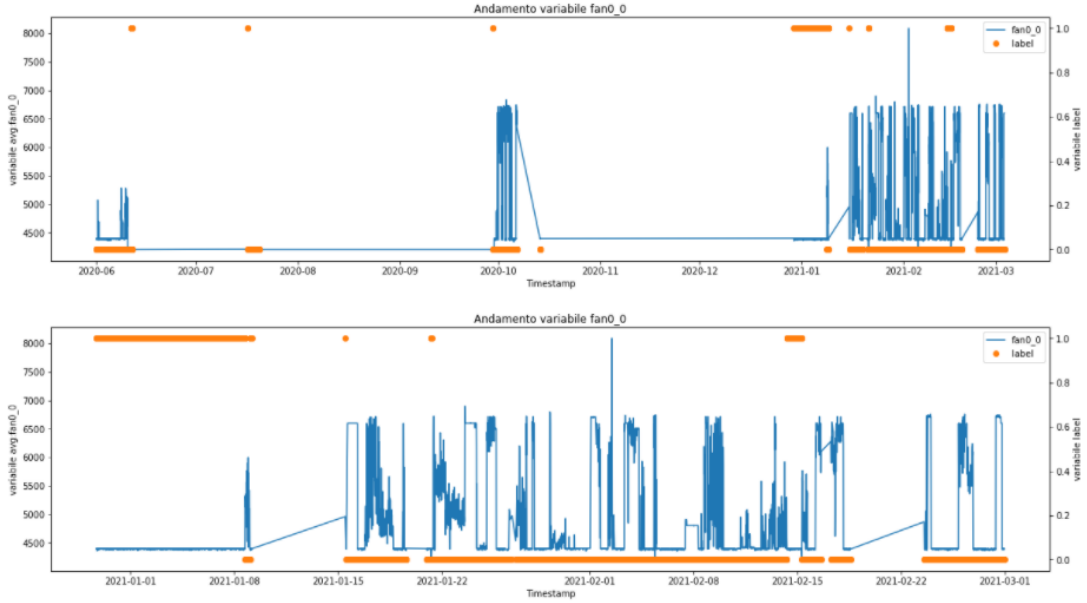

Figura 95 - andamento variabile fan0_0 nodo r211n19

Osservando la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

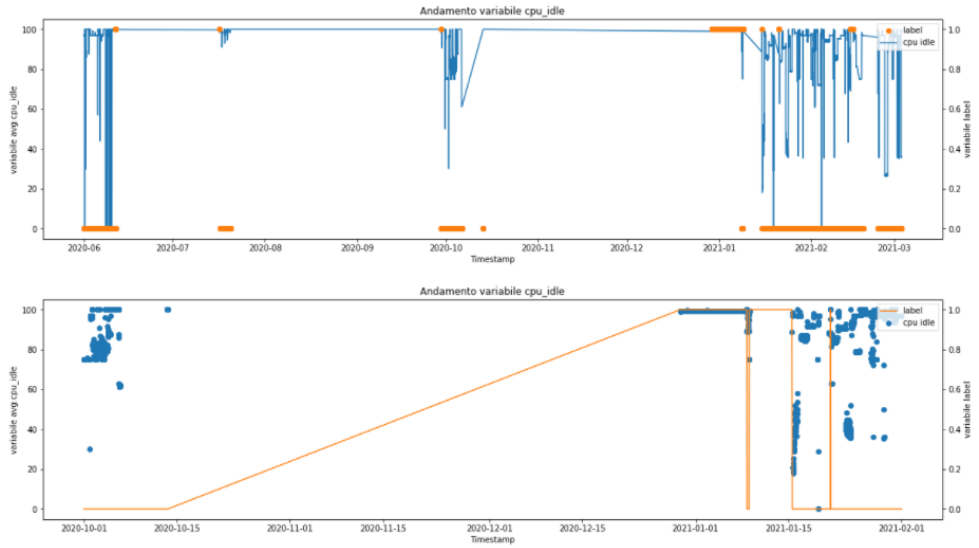

Figura 96 - andamento variabile cpu_idle r211n19

Andando ad effettuare il plot sul carico medio a un minuto, a cinque minuti a quindici minuti, vedremo che si possono riscontrare dei picchi proprio in corrispondenza delle anomalie.

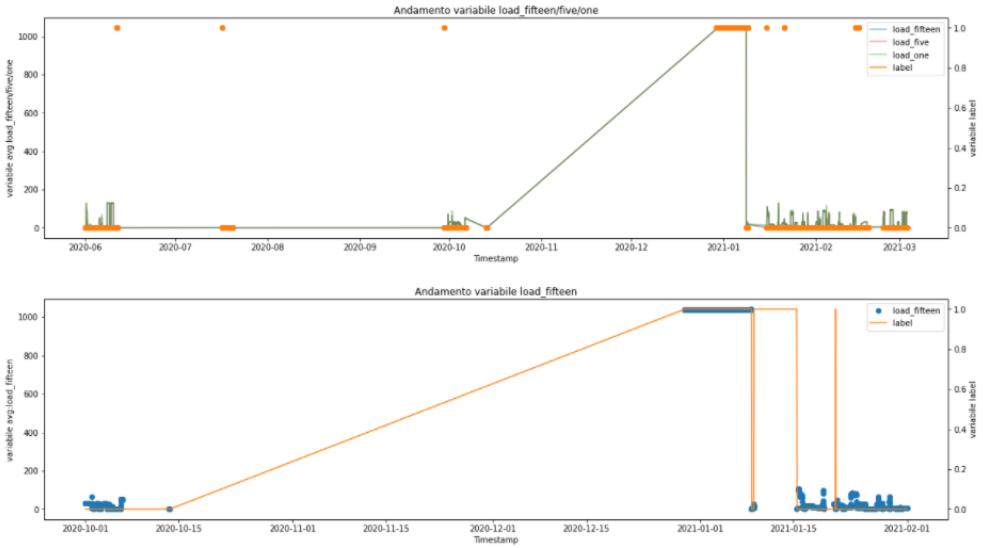

Figura 97 - carico medio nodo r211n19

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

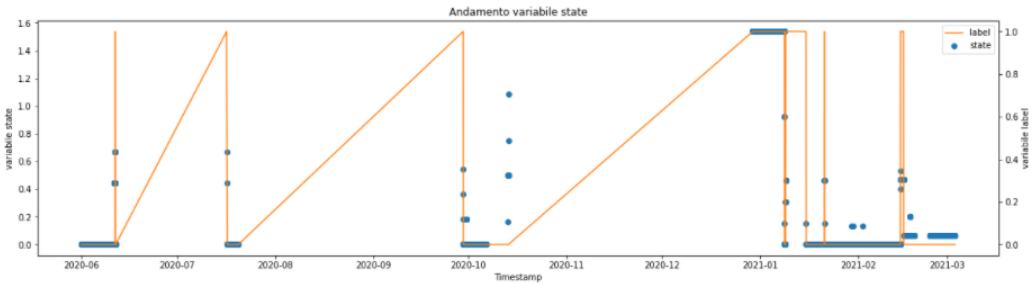

Figura 98 - andamento variabile state nodo r211n19

Nella tabella seguente è possibile osservare le informazioni del nodo più di rilievo.

	Shape	Primo camp.	Ultimo camp.	Stato anomalo	Salti	Numero anomalie	Media anomalie
r211n19	(6670, 234)	2020-05-31 22:15:00	2021-03-03 13:45:00	16%	Si	9	31 h

3.7 Nodo r212n06

Le letture in questo nodo vengono riportate in una tabella di 10244 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:00:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 10195 letture in cui il nodo risulta in stato *normale* e 49 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

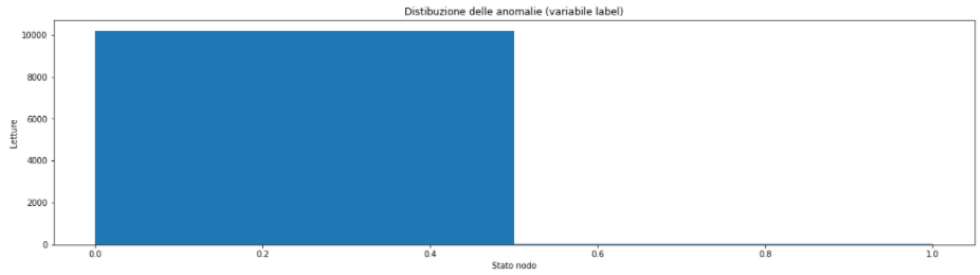

Figura 99 - distribuzione delle anomalie in base alla variabile label nodo r212n06

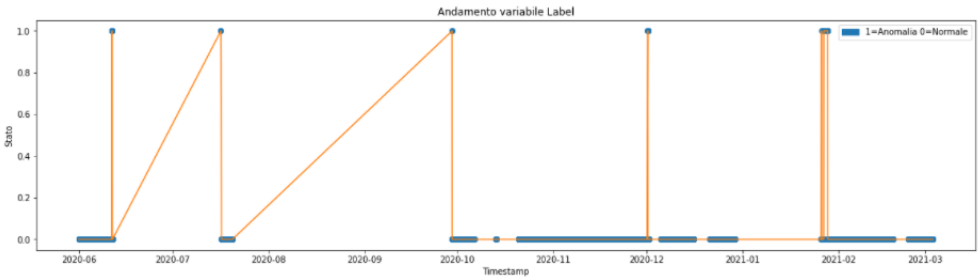

Figura 100 - andamento variabile label nodo r212n06

Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, velocità dei fan e energia consumata, i risultati saranno simili.

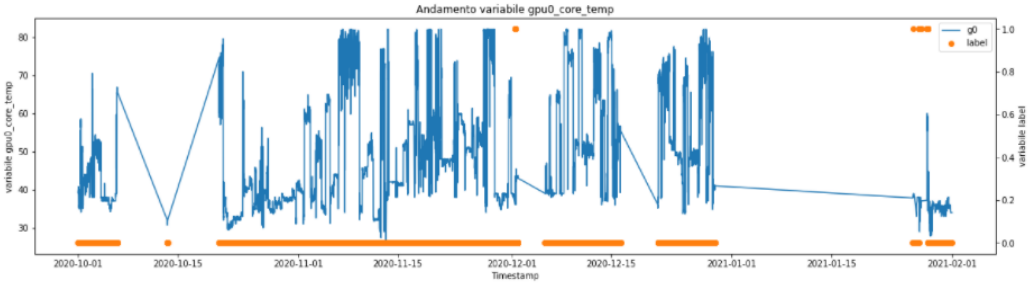

Figura 101 - andamento variabile gpu0_core_temp nodo r212n06

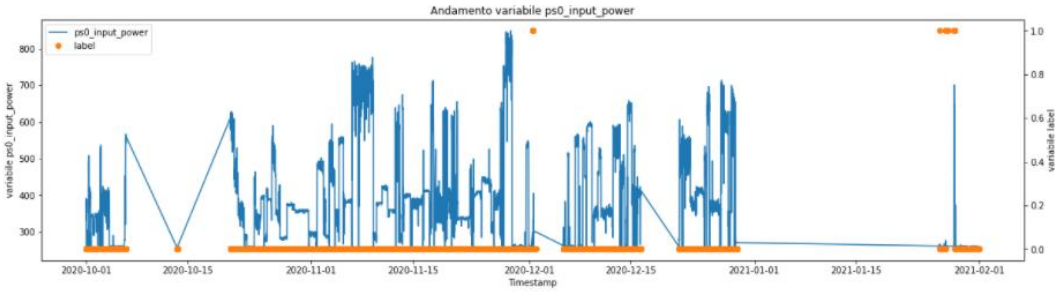

Figura 102 - andamento variabile ps0_input_power nodo r212n06

Osservando la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

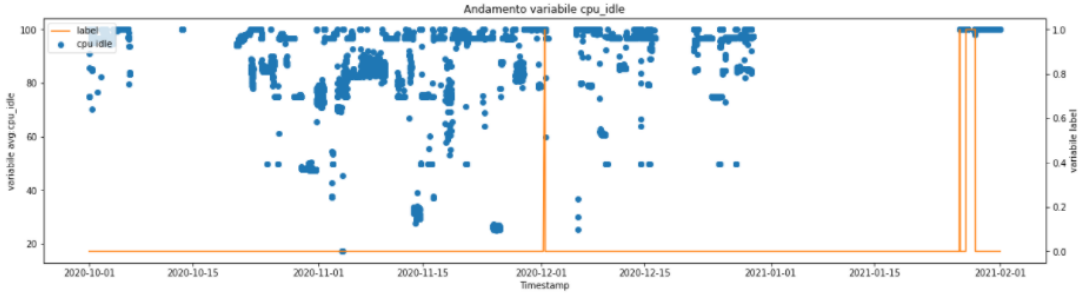

Figura 103 - andamento variabile cpu_idle nodo r212n06

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

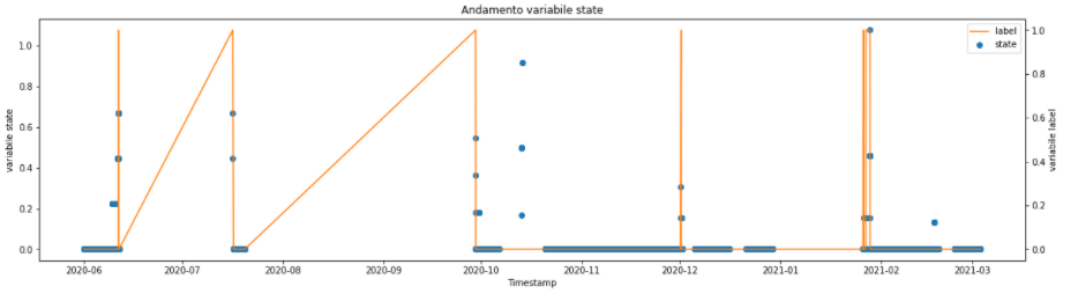

Figura 104 - andamento variabile state nodo r212n06

Nella tabella seguente è possibile osservare le informazioni del nodo più di rilievo.

	Shape	Primo camp.	Ultimo camp.	Stato anomalo	Salti	Numero anomalie	Media anomalie
r212n06	(10244, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	0.4%	Si	14	38 min

3.8 Nodo r215n05

Le letture in questo nodo vengono riportate in una tabella di 17576 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:00:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 12388 letture in cui il nodo risulta in stato *normale* e 5188 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

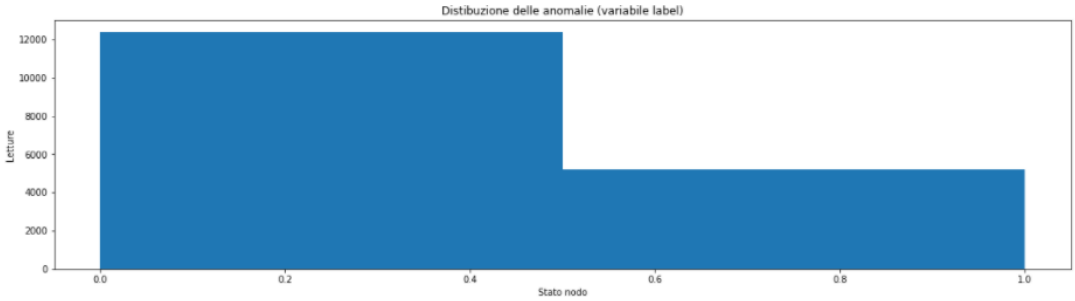

Figura 105 - distribuzione delle anomalie in base alla variabile label nodo r215n05

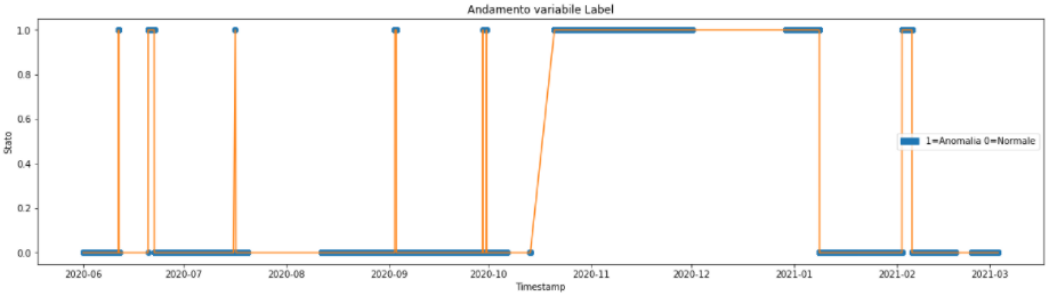

Figura 106 - andamento variabile label nodo r215n05

Si è provato ad effettuare uno zoom nell'intervallo 2020/10 e 2021/02, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, velocità dei fan e energia consumata, i risultati saranno simili.

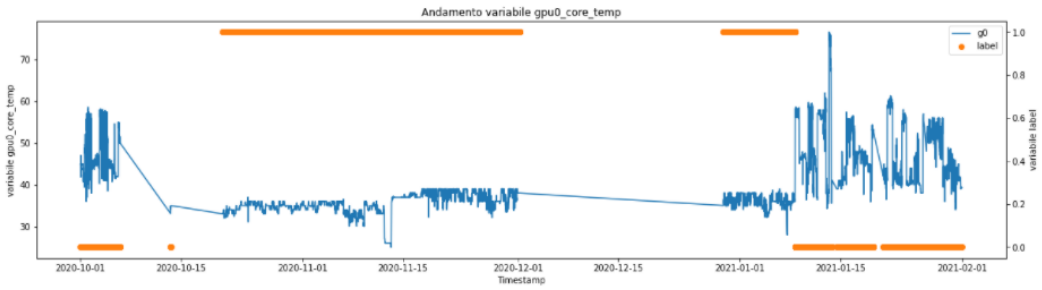

Figura 107 - andamento variabile gpu0_core_temp

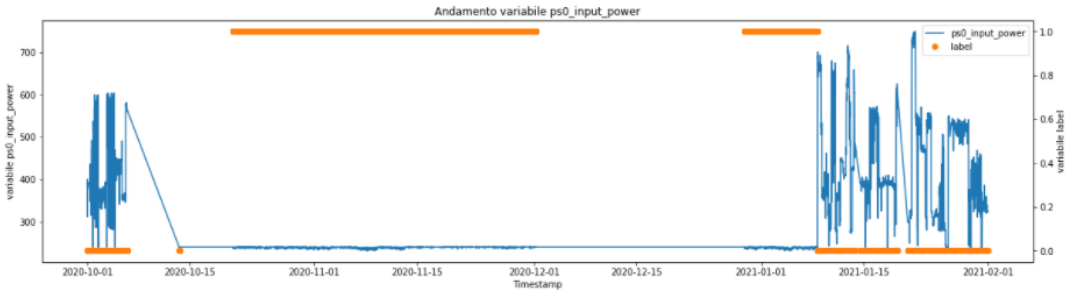

Figura 108 - andamento variabile ps0_input_power nodo r215n05

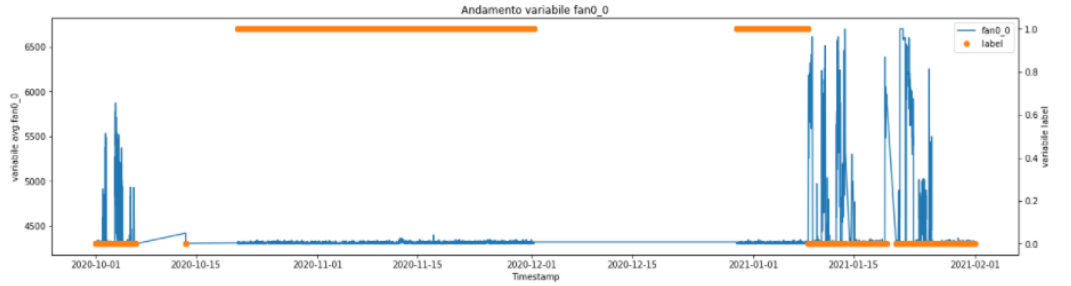

Figura 109 - andamento variabile fan0_0 nodo r215n05

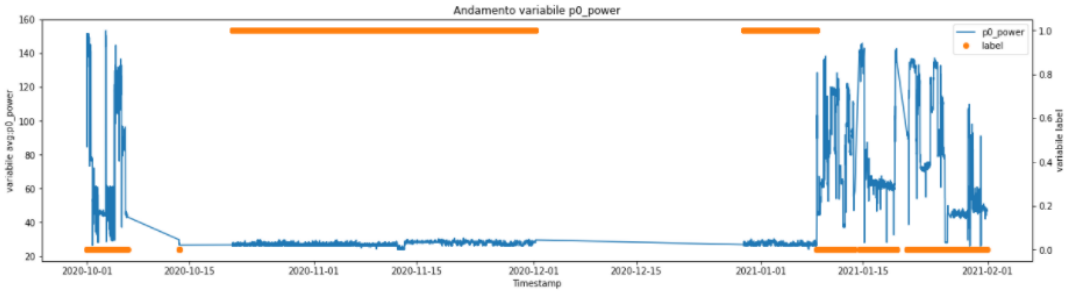

Figura 110 - andamento variabile p0_power

Osservando la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

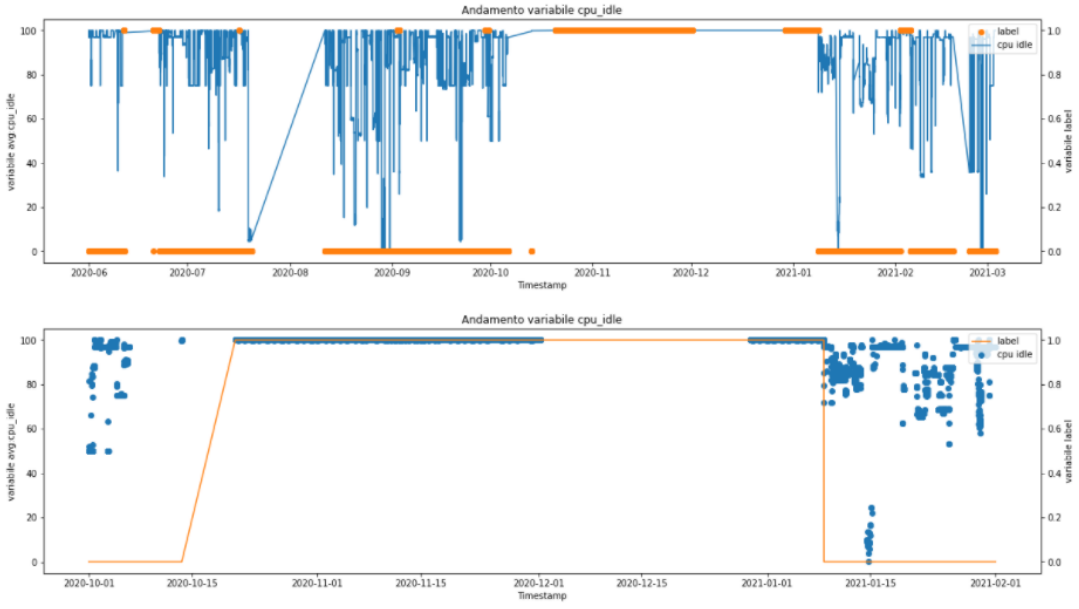

Figura 111 - andamento variabile cpu_idle nodo r215n05

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

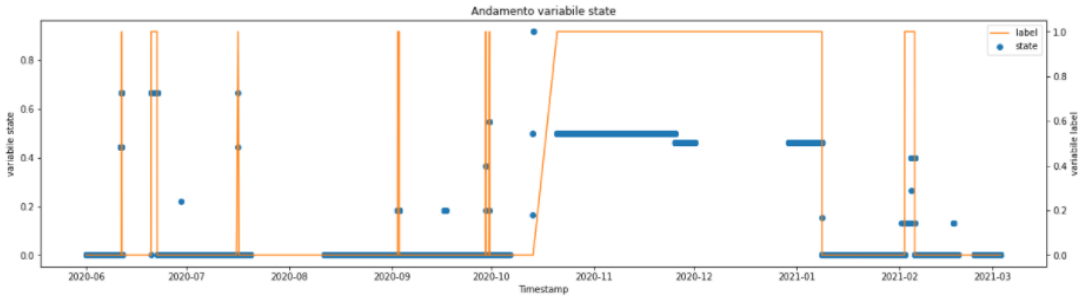

Figura 112 - andamento variabile state nodo r215n05

Nella tabella seguente è possibile osservare le informazioni del nodo più di rilievo.

	Shape	Primo camp.	Ultimo camp.	Stato anomalo	Salti	Numero anomalie	Media anomalie
r215n05	(17576, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	29%	Si	13	104 h 45 min

3.9 Nodo r218n03

Le letture in questo nodo vengono riportate in una tabella di 8300 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:00:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 7332 letture in cui il nodo risulta in stato *normale* e 968 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

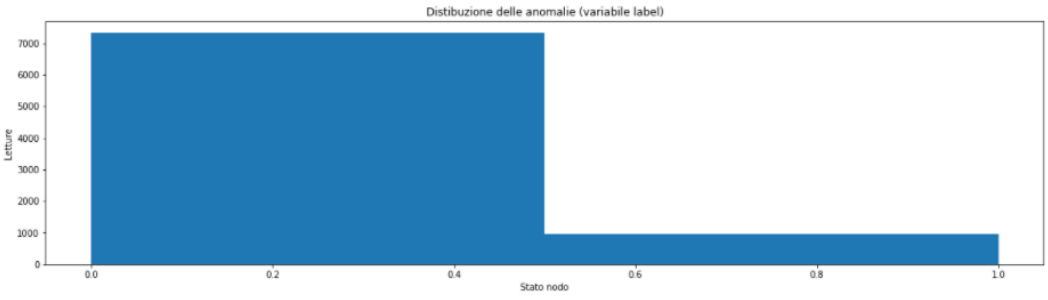

Figura 113 - distribuzione delle anomalie in base alla variabile label nodo r218n03

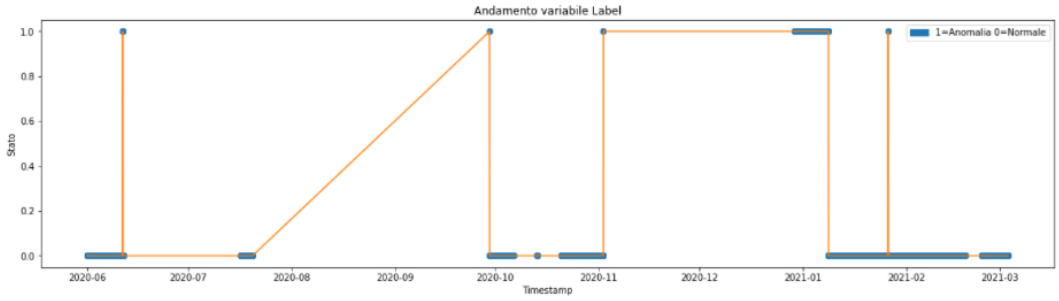

Figura 114 - andamento variabile label nodo r218n03

Si è provato ad effettuare uno zoom nell'intervallo 2020/10 e 2021/02, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, velocità dei fan e energia consumata, i risultati saranno simili.

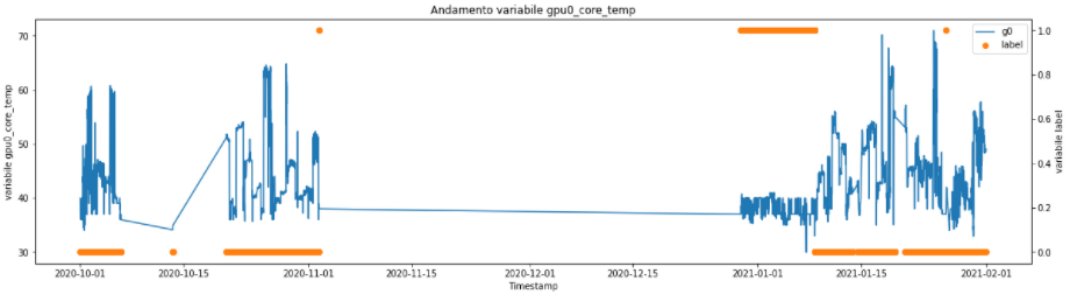

Figura 115 - andamento variabile gpu0_core_temp nodo r218n03

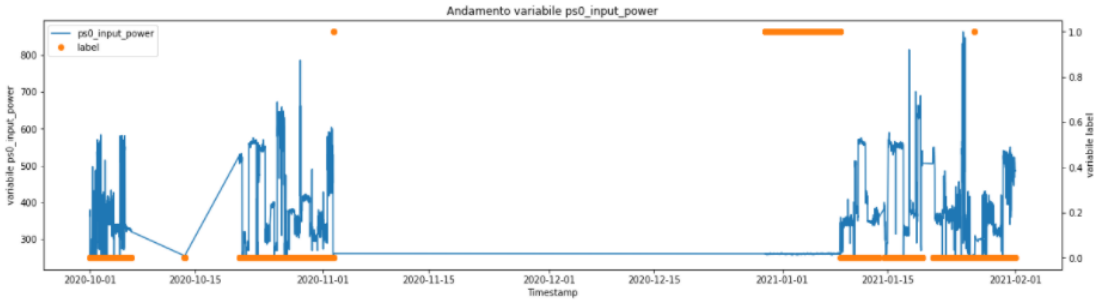

Figura 116 - andamento variabile ps0_input_power nodo r218n03

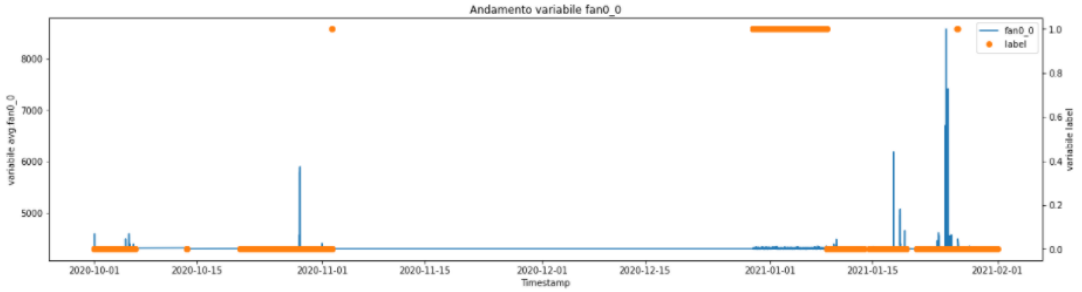

Figura 117 - andamento variabile fan0_0 nodo r218n03

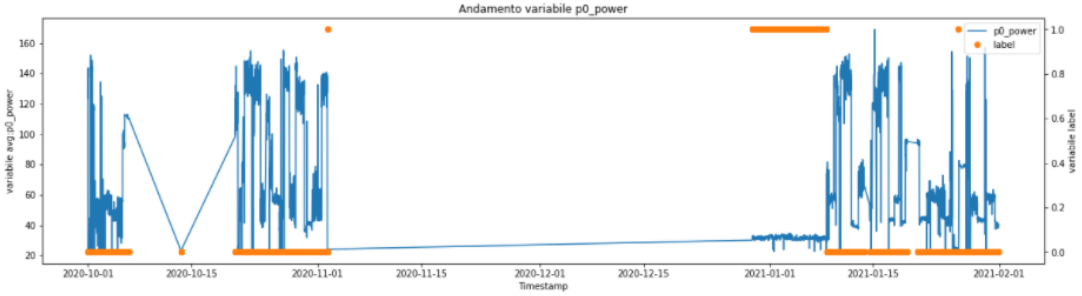

Figura 118 - andamento variabile p0_power nodo r218n03

Osservando la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

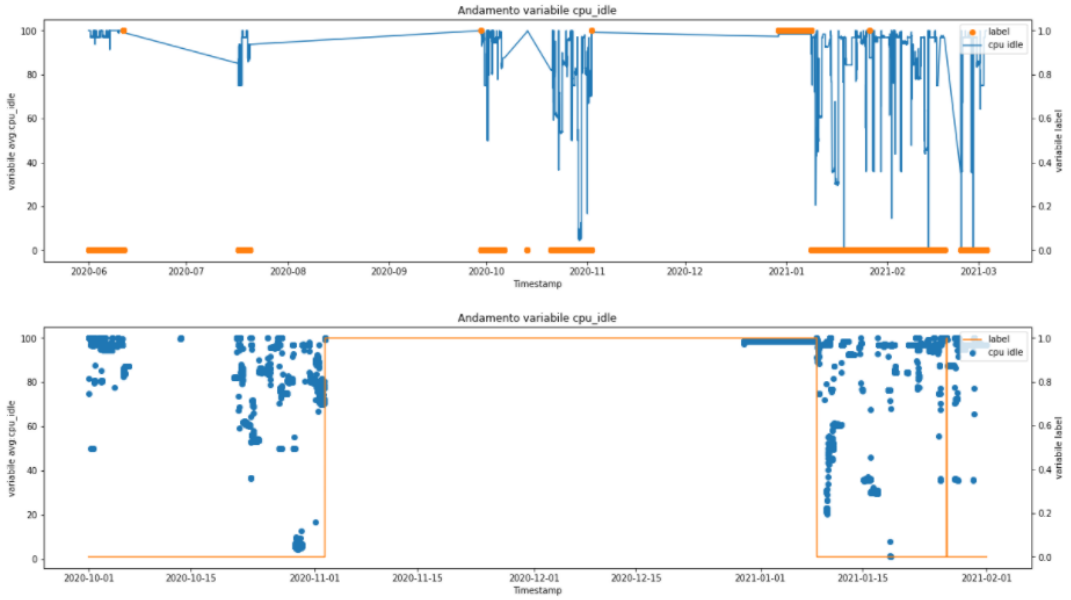

Figura 119 - andamento variabile cpu_idle nodo r218n03

Andando ad effettuare il plot sul carico medio a un minuto, a cinque minuti a quindici minuti, vedremo che si possono riscontrare dei picchi proprio in corrispondenza delle anomalie.

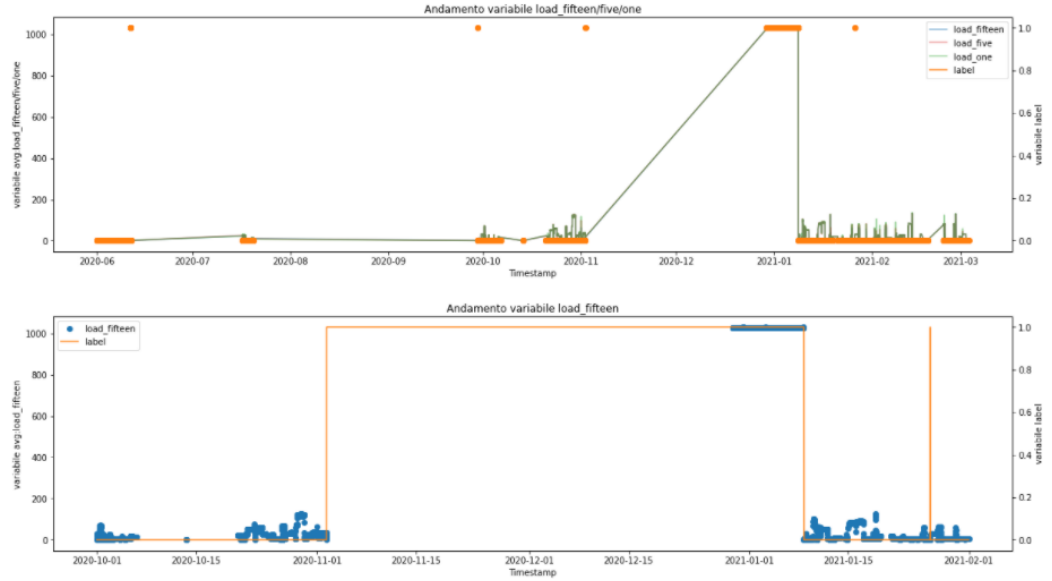

Figura 120 - carico medio nodo r218n03

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

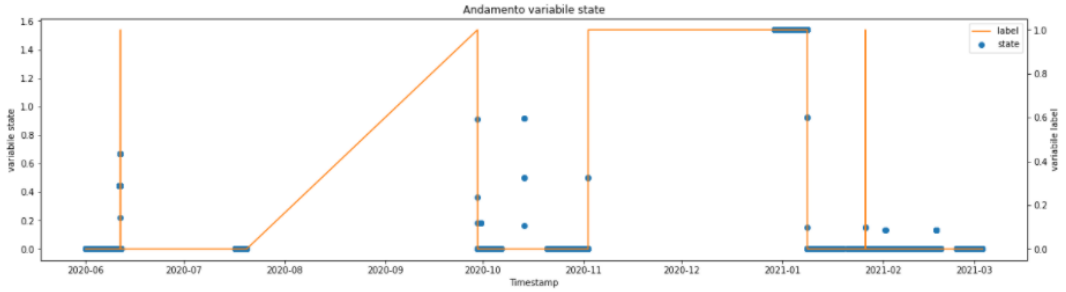

Figura 121 - andamento variabile state nodo r218n03

Nella tabella seguente è possibile osservare le informazioni del nodo più di rilievo.

	Shape	Primo camp.	Ultimo camp.	Stato anomalo	Salti	Numero anomalie	Media anomalie
r218n03	(8300, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	12%	Si	5	37 h 30 min

3.10 Nodo r224n20

Le letture in questo nodo vengono riportate in una tabella di 10792 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:15:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 10634 letture in cui il nodo risulta in stato *normale* e 158 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

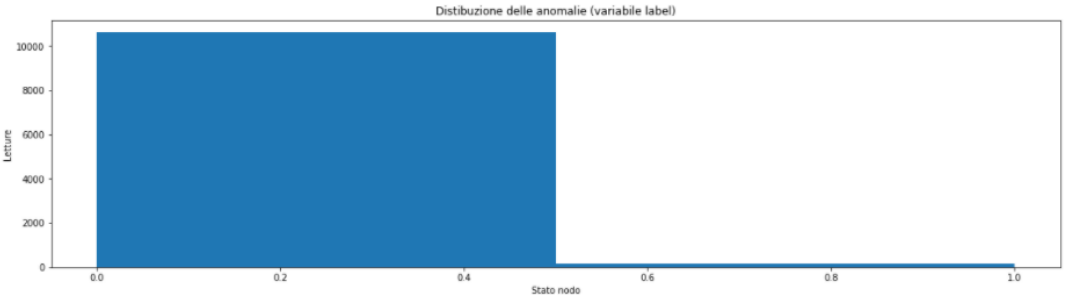

Figura 122 - distribuzione delle anomalie in base alla variabile label nodo r224n20

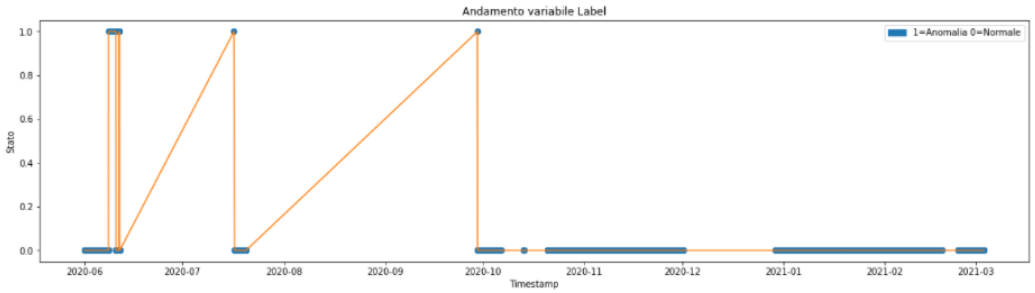

Figura 123 - andamento variabile label nodo r224n20

Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, velocità dei fan e energia consumata, i risultati saranno simili.

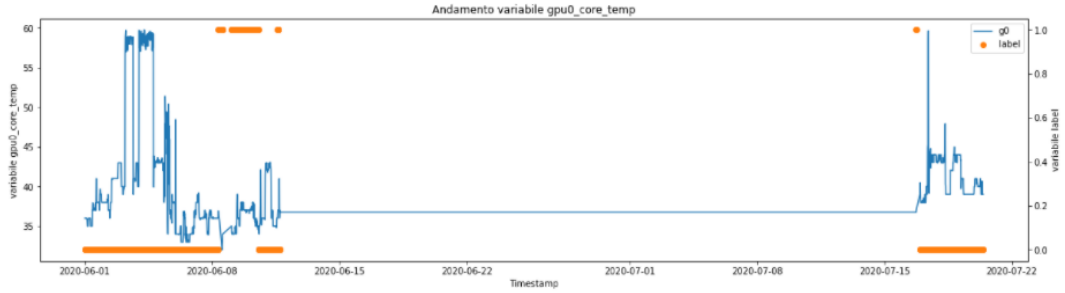

Figura 124 - andamento variabile gpu0_core_temp nodo r224n20

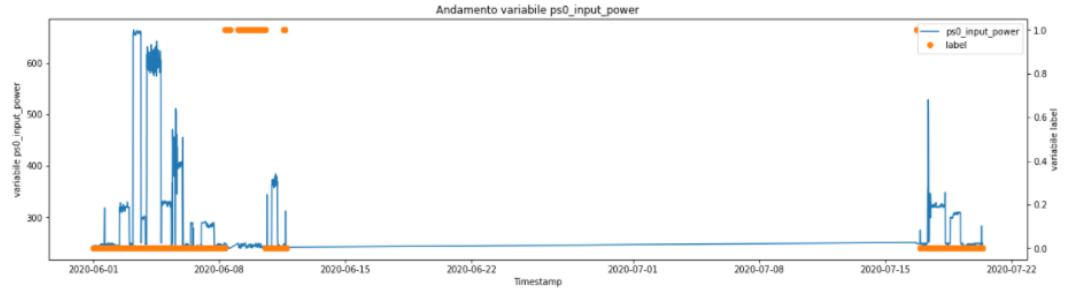

Figura 125 - andamento variabile ps0_input_power nodo r224n20

Osservando la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

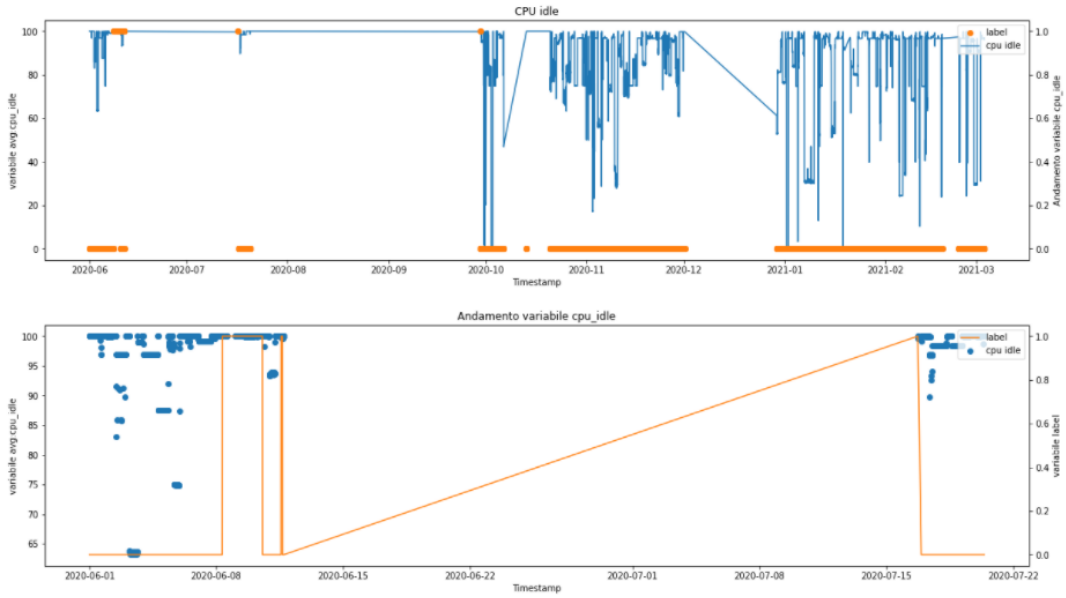

Figura 126 - andamento variabile cpu_idle nodo r224n20

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

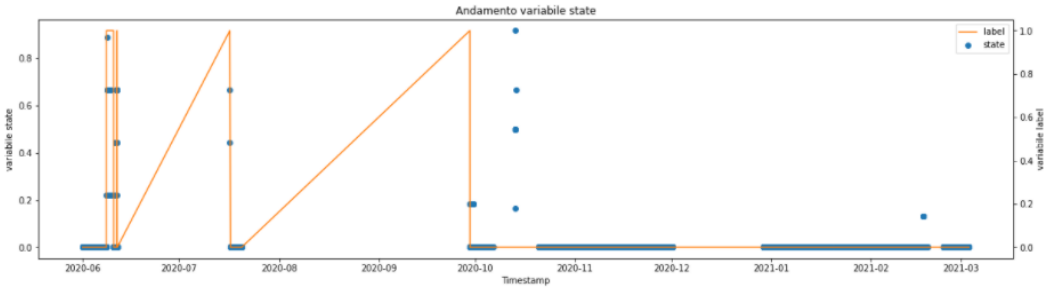

Figura 127 - andamento variabile state nodo r224n20

Nella tabella seguente è possibile osservare le informazioni del nodo più di rilievo.

	Shape	Primo camp.	Ultimo camp.	Stato anomalo	Salti	Numero anomalie	Media anomalie
r224n20	(10792, 234)	2020-05-31 22:15:00	2021-03-03 13:45:00	1,5%	Si	11	3 h 30 min

3.11 Nodo r239n12

Le letture in questo nodo vengono riportate in una tabella di 13495 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:00:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 9102 letture in cui il nodo risulta in stato *normale* e 4393 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

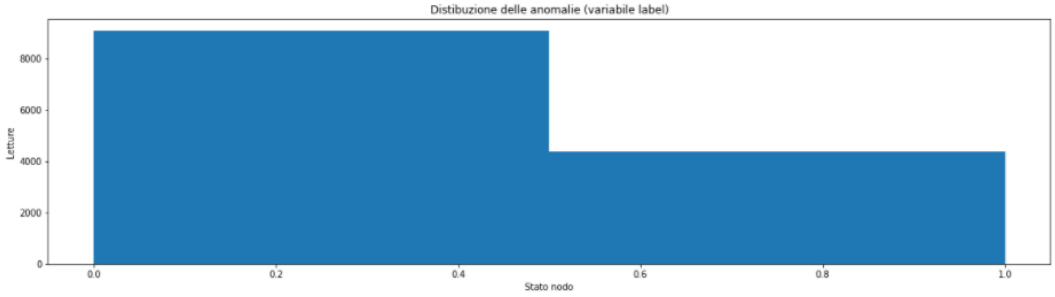

Figura 128 - distribuzione delle anomalie in base alla variabile label nodo r239n12

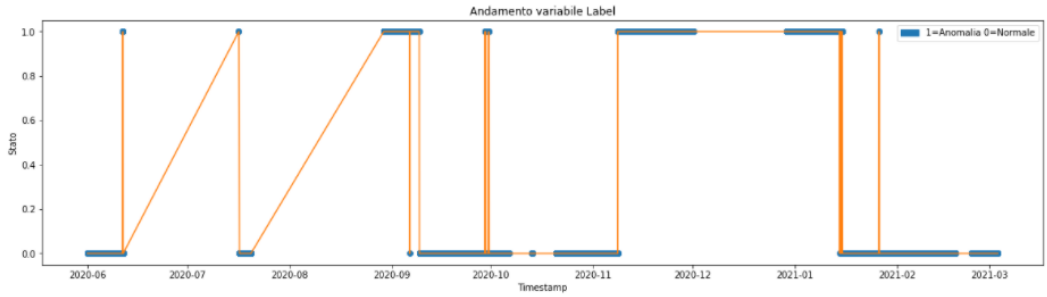

Figura 129 - andamento variabile label nodo r239n12

Si è provato ad effettuare uno zoom nell'intervallo 2020/08 e 2021/02, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, velocità dei fan e energia consumata, i risultati saranno simili.

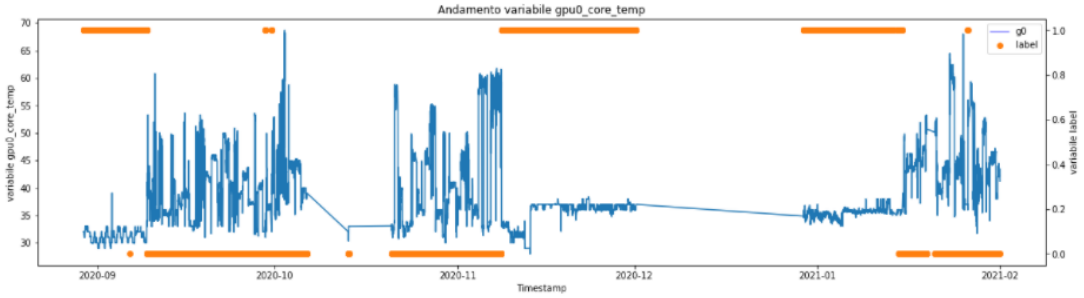

Figura 130 - andamento variabile gpu0_core_temp nodo r239n12

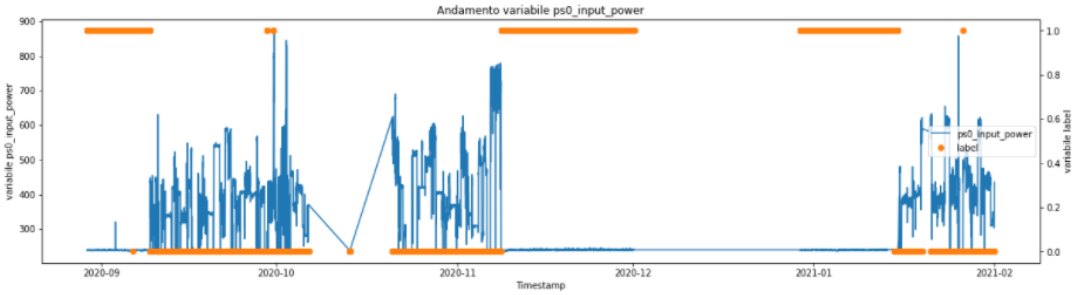

Figura 131 - andamento variabile ps0_input_power nodo r239n12

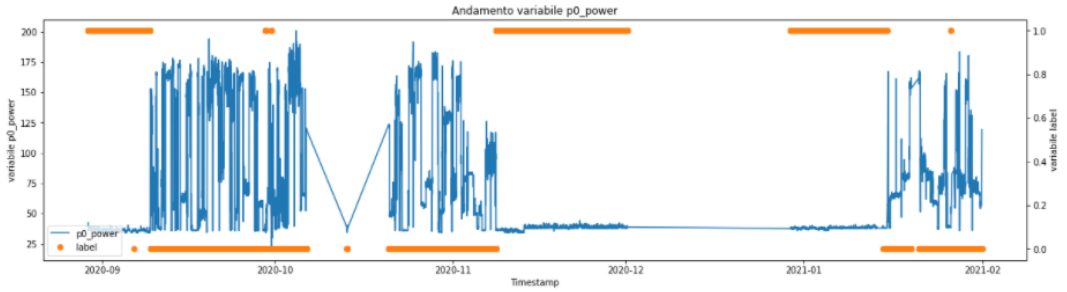

Figura 132 - andamento variabile p0_power nodo r239n12

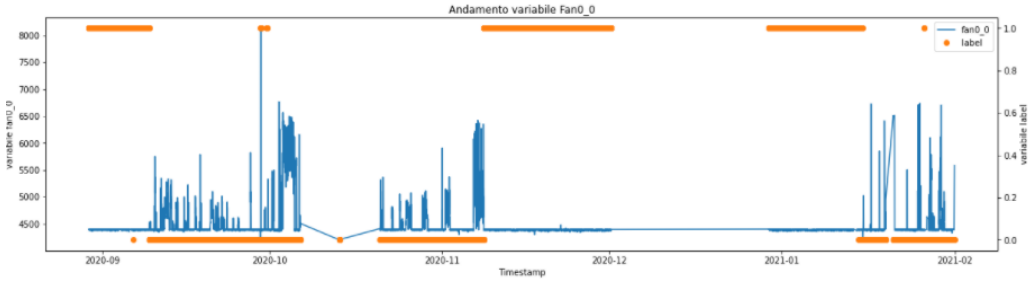

Figura 133 - andamento variabile fan0_0 nodo r239n12

Osservando la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

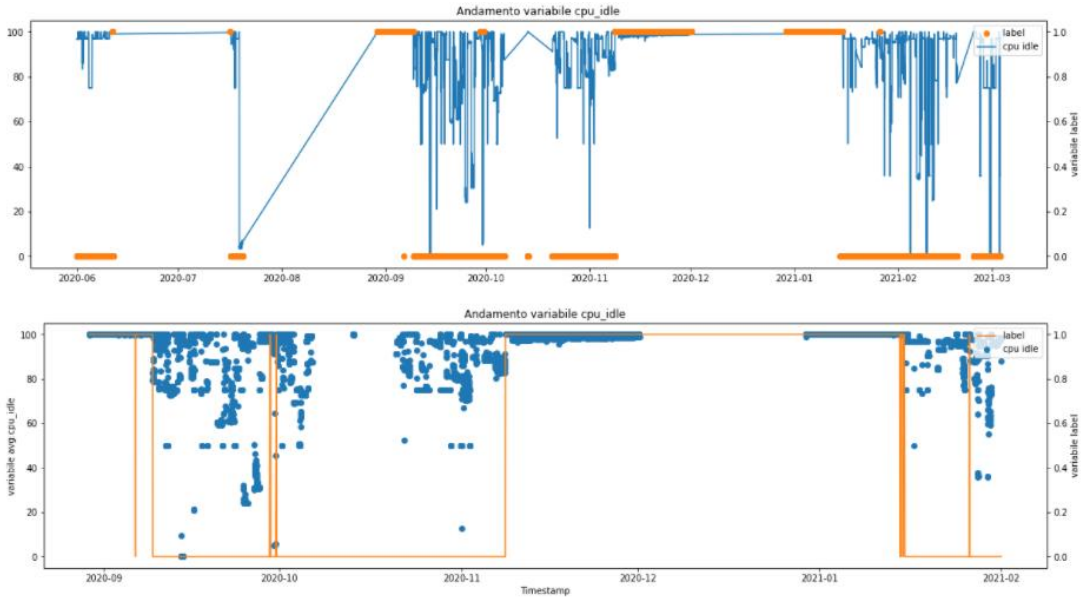

Figura 134 - andamento variabile cpu_idle nodo r239n12

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

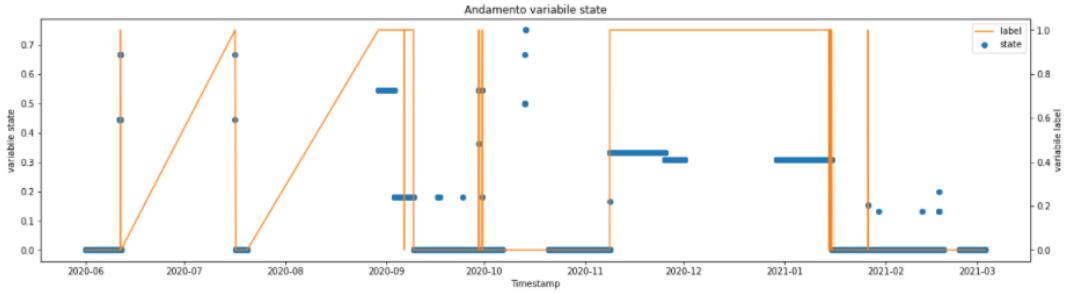

Figura 135 - andamento variabile state nodo r239n12

Nella tabella seguente è possibile osservare le informazioni del nodo più di rilievo.

	Shape	Primo camp.	Ultimo camp.	Stato anomalo	Salti	Numero anomalie	Media anomalie
r239n12	(13495, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	33%	Si	27	43 h 30 min

3.12 Nodo r243n11

Le letture in questo nodo vengono riportate in una tabella di 12182 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:15:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 11717 letture in cui il nodo risulta in stato *normale* e 465 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

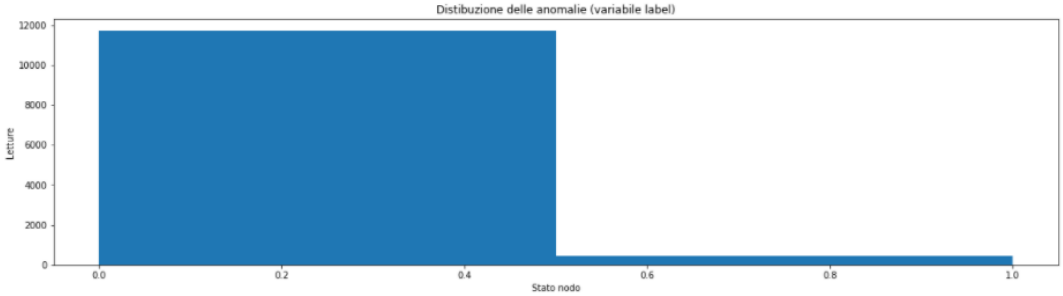

Figura 136 - distribuzione delle anomalie in base alla variabile label nodo r243n11

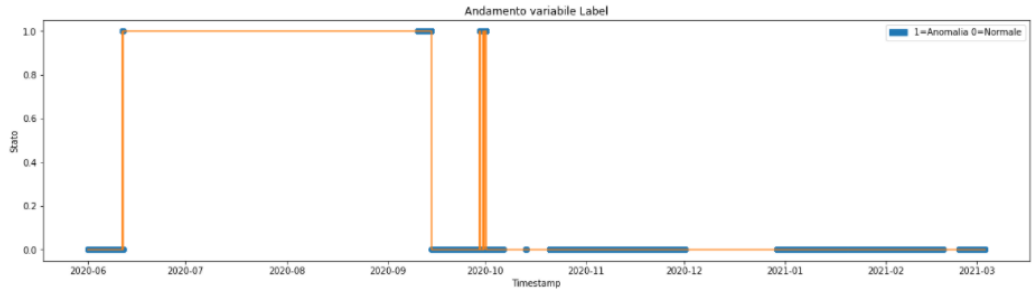

Figura 137 - andamento variabile label nodo r243n11

Si è provato ad effettuare uno zoom nell'intervallo 2020/08 e 2021/02, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, velocità dei fan e energia consumata, i risultati saranno simili.

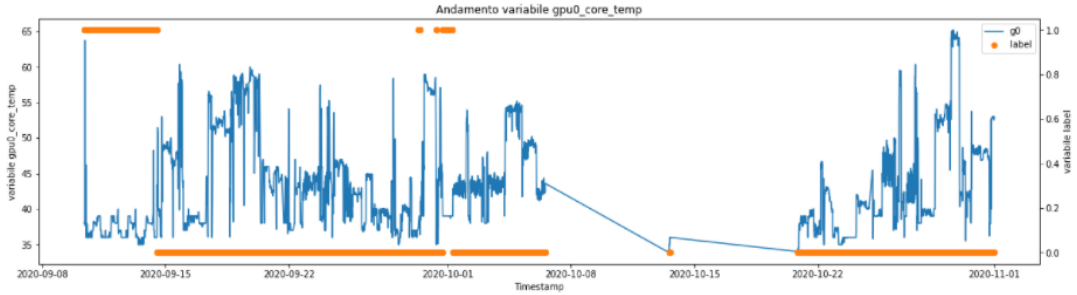

Figura 138 - andamento variabile gpu0_core_temp nodo r243n11

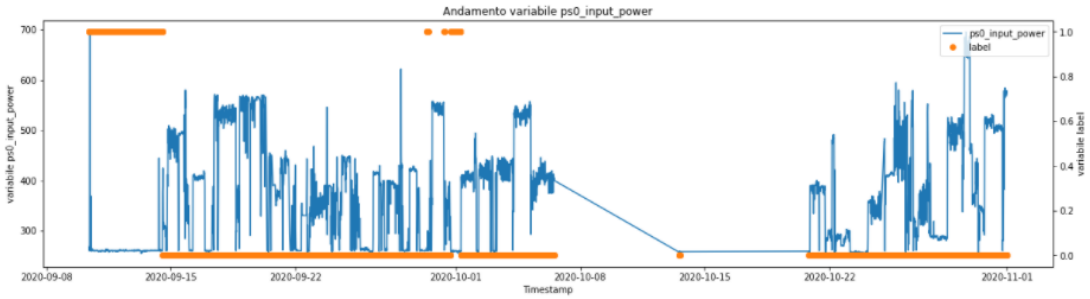

Figura 139 - andamento variabile ps0_input_power nodo r243n11

Osservando la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

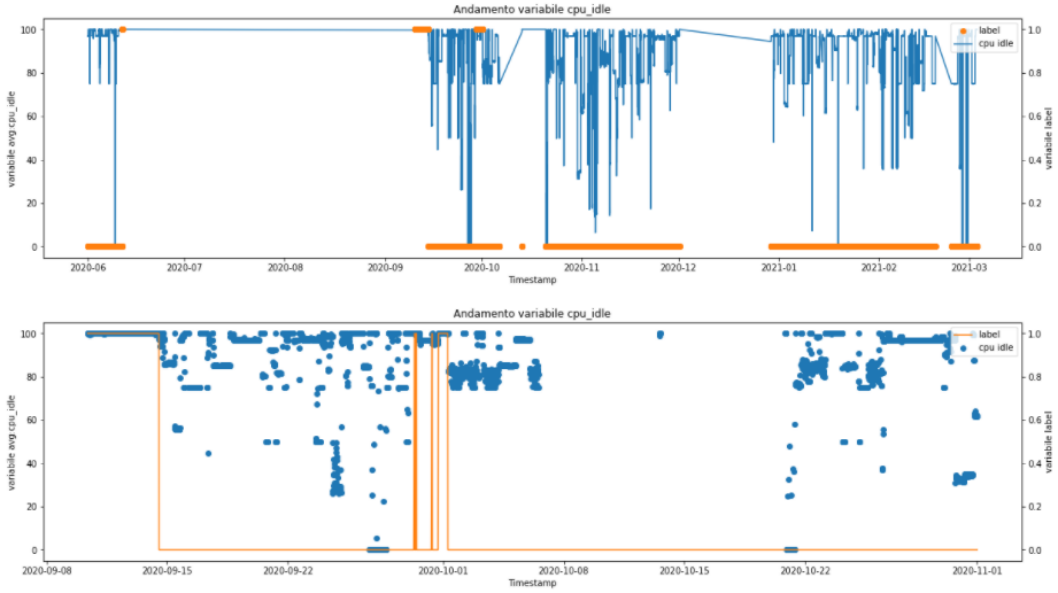

Figura 140 - andamento variabile cpu_idle nodo r243n11

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

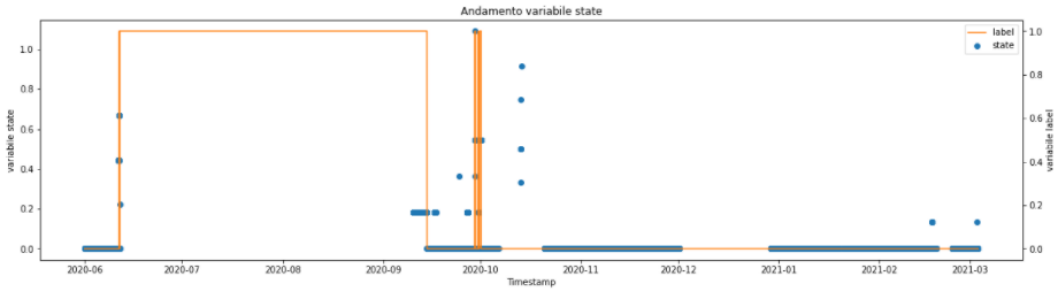

Figura 141 - andamento variabile state nodo r243n11

Nella tabella seguente è possibile osservare le informazioni del nodo più di rilievo.

	Shape	Primo camp.	Ultimo camp.	Stato anomalo	Salti	Numero anomalie	Media anomalie
r243n11	(12182, 234)	2020-05-31 22:15:00	2021-03-03 13:45:00	4%	Si	8	14 h 15 min

3.13 Nodo r249n06

Le letture in questo nodo vengono riportate in una tabella di 14932 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:00:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 14477 letture in cui il nodo risulta in stato *normale* e 455 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

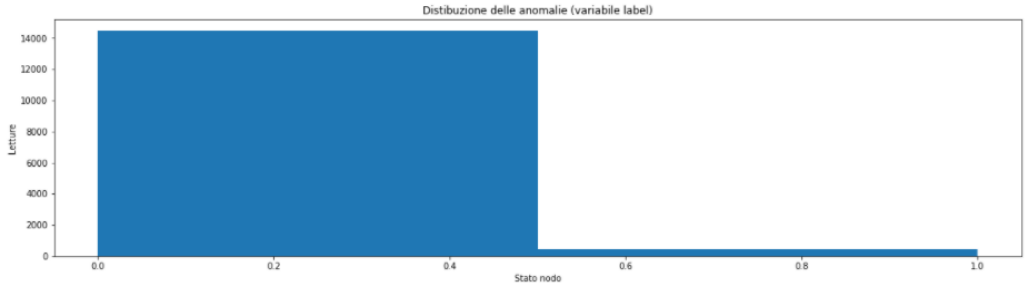

Figura 142 - distribuzione delle anomalie in base alla variabile label nodo r249n06

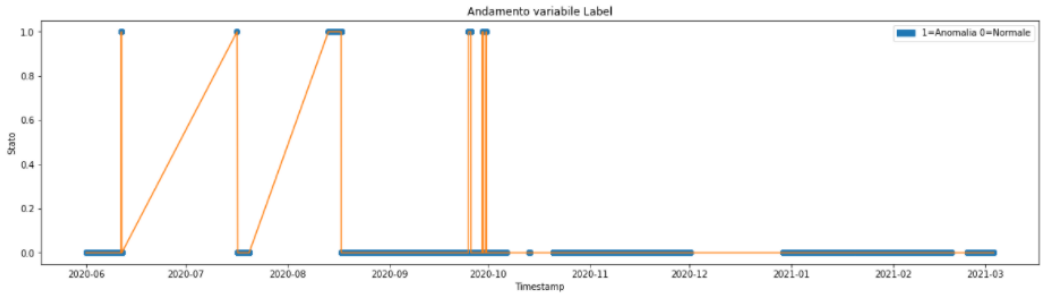

Figura 143 - andamento variabile label nodo n249n06

Si è provato ad effettuare uno zoom nell'intervallo 2020/08 e 2020/11, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, velocità dei fan e energia consumata, i risultati saranno simili.

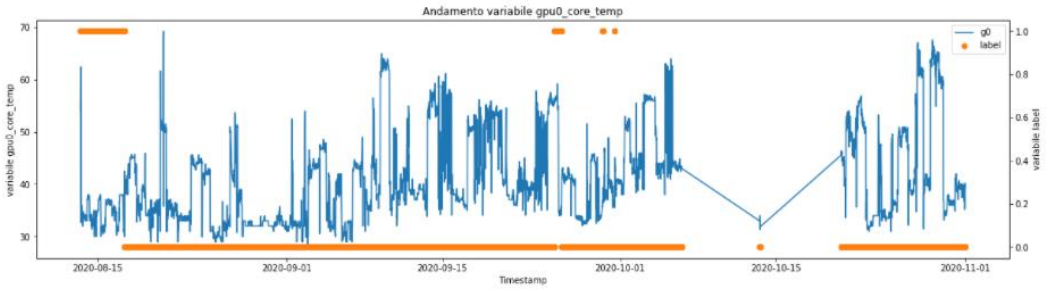

Figura 144 - andamento variabile gpu0_core_temp nodo n249n06

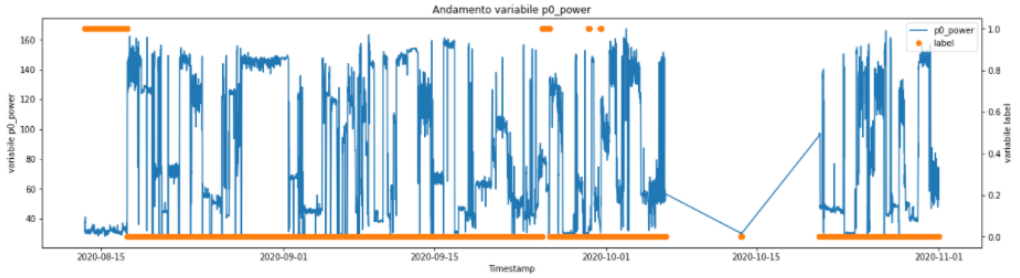

Figura 145 - andamento variabile p0_power nodo r249n06

Osservando la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

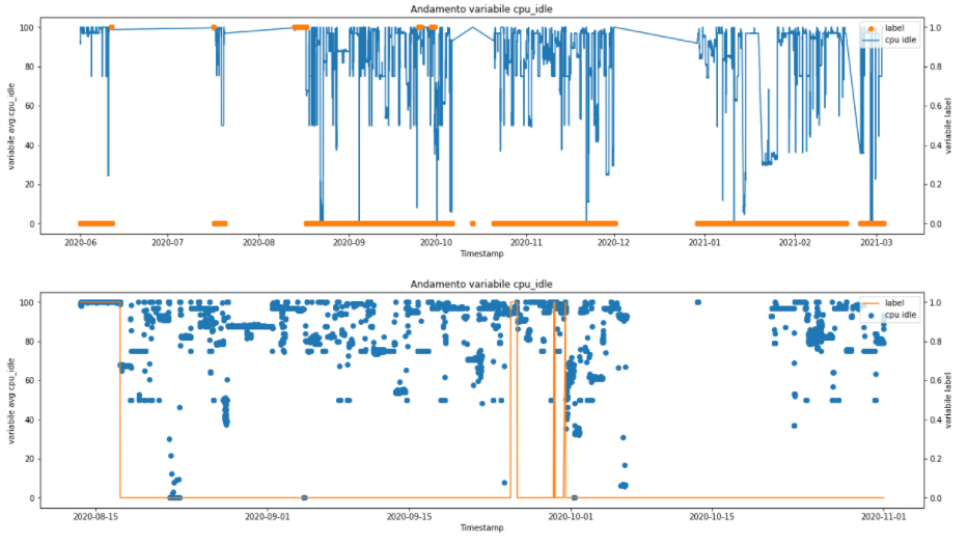

Figura 146 - andamento variabile cpu_idle nodo r249n06

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

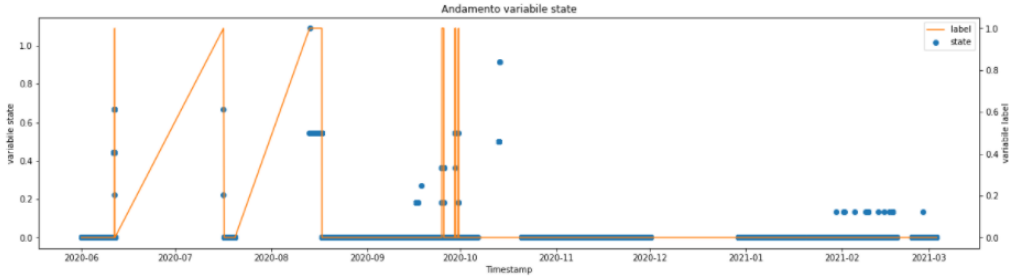

Figura 147 - andamento variabile state nodo r249n06

Nella tabella seguente è possibile osservare le informazioni del nodo più di rilievo.

	Shape	Primo camp.	Ultimo camp.	Stato anomalo	Salti	Numero anomalie	Media anomalie
r249n06	(14932, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	3%	Si	9	12 h 30 min

3.14 Nodo r253n06

Le letture in questo nodo vengono riportate in una tabella di 9862 righe e 234 colonne. Il primo campionamento, risale al 2020-06-16 13:00:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 9684 letture in cui il nodo risulta in stato *normale* e 178 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

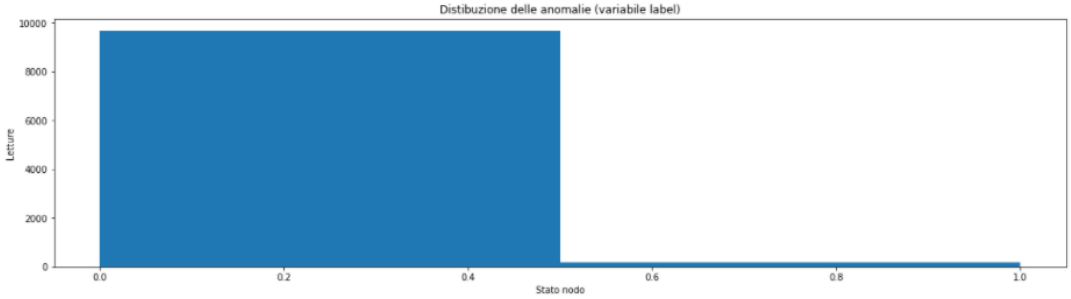

Figura 148 - distribuzione delle anomalie in base alla variabile label nodo r253n06

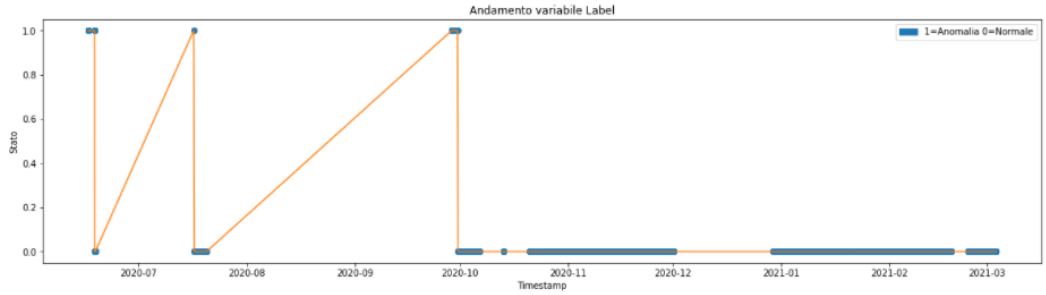

Figura 149 - andamento variabile label nodo r253n06

Si è provato ad effettuare uno zoom nell'intervallo 2020/06 e 2020/10, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, velocità dei fan e energia consumata, i risultati saranno simili.

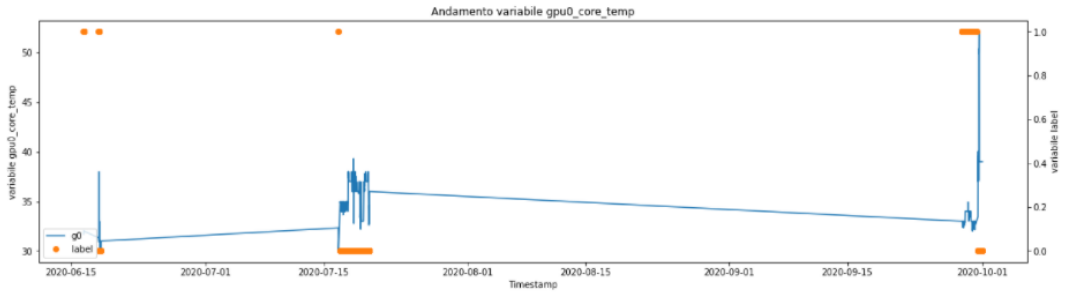

Figura 150 - andamento variabile gpu0_core_temp nodo r253n06

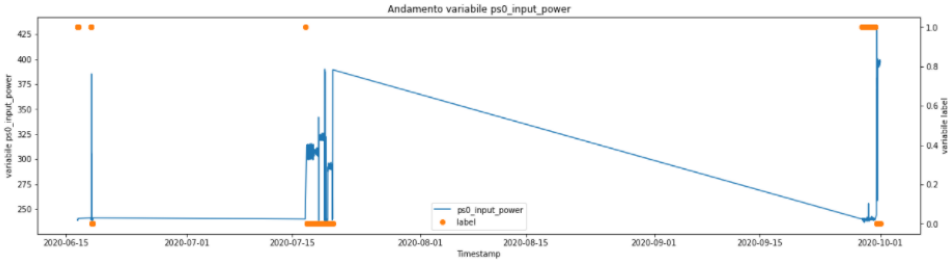

Figura 151 - andamento variabile ps0_input_power nodo r253n06

Osservando la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

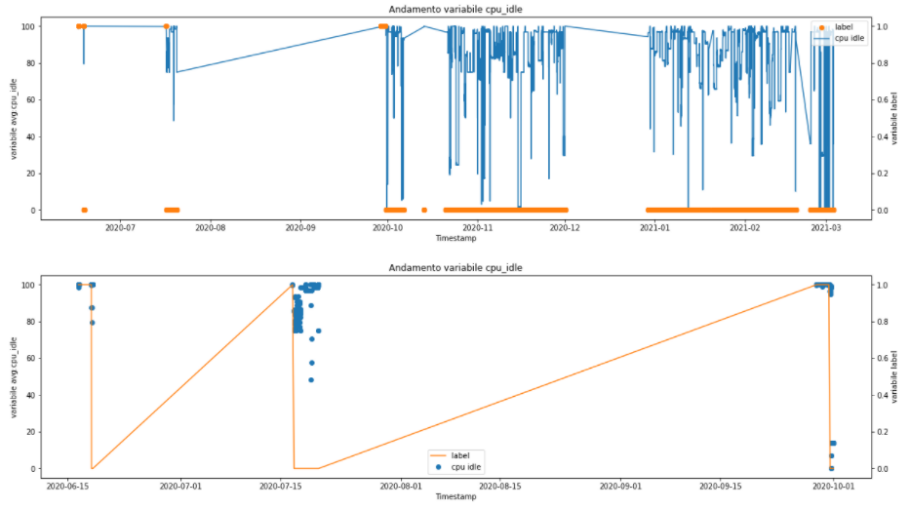

Figura 152 - andamento variabile cpu_idle nodo r253n06

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

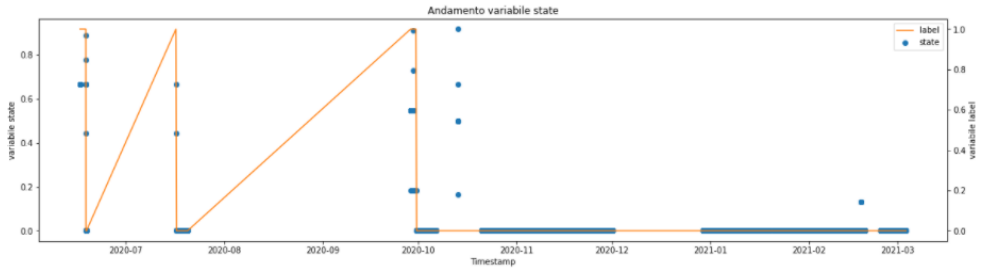

Figura 153 - andamento variabile state nodo r253n06

Nella tabella seguente è possibile osservare le informazioni del nodo più di rilievo.

	Shape	Primo camp.	Ultimo camp.	Stato anomalo	Salti	Numero anomalie	Media anomalie
r253n06	(9862, 234)	2020-06-16 13:00:00	2021-03-03 13:45:00	2%	Si	8	5 h 30 min

3.15 Nodo r254n01

Le letture in questo nodo vengono riportate in una tabella di 17191 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:30:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 15873 letture in cui il nodo risulta in stato *normale* e 1318 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

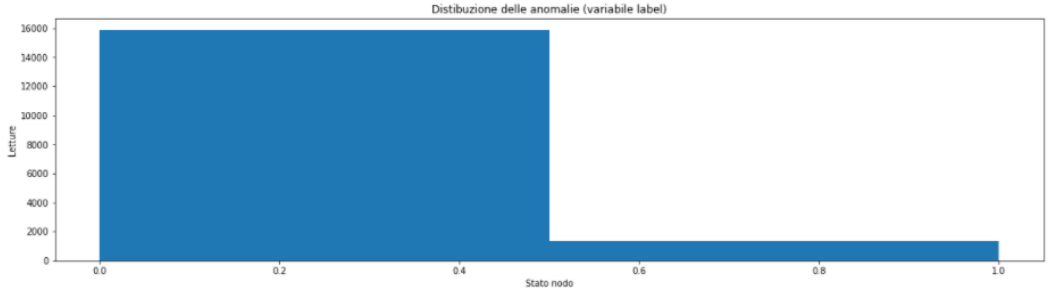

Figura 154 - distribuzione delle anomalie in base alla variabile label nodo r254n01

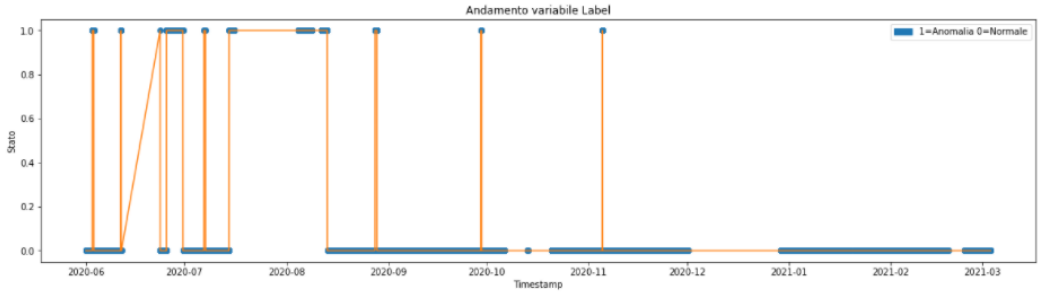

Figura 155 - andamento variabile label nodo r254n01

Si è provato ad effettuare uno zoom nell'intervallo 2020/06 e 2020/10, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, velocità dei fan e energia consumata, i risultati saranno simili.

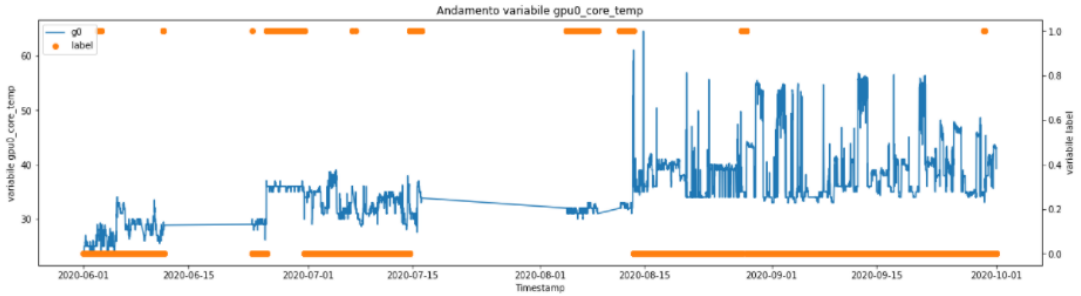

Figura 156 - andamento variabile gpu0_core_temp nodo r254n01

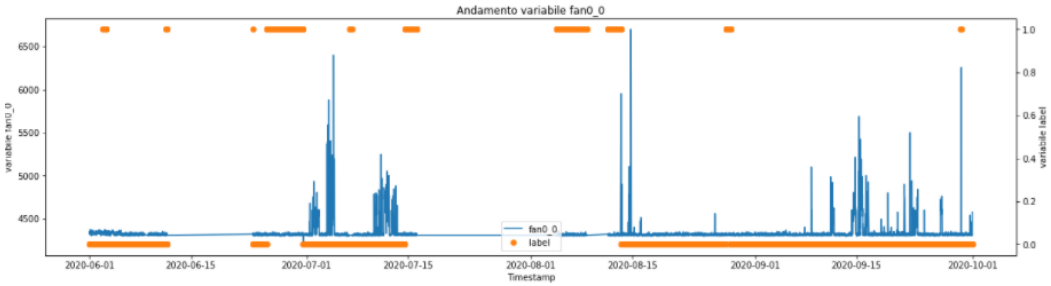

Figura 157 - andamento variabile fan0_0 nodo r254n01

Osservando la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

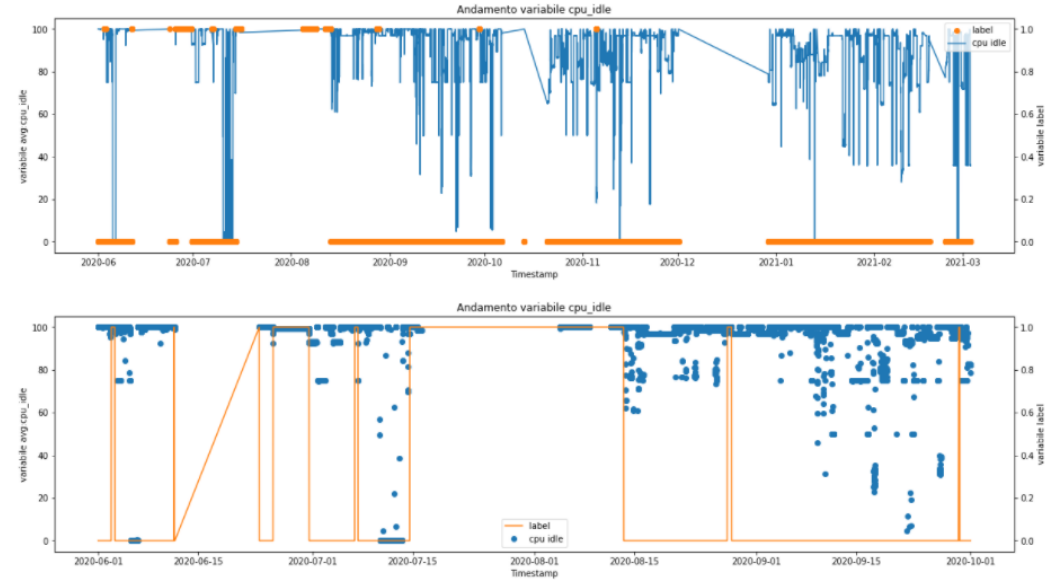

Figura 158 - andamento variabile cpu_idle nodo r254n01

Andando ad effettuare il plot sul carico medio a un minuto, a cinque minuti a quindici minuti, vedremo che si possono riscontrare dei picchi proprio in corrispondenza delle anomalie.

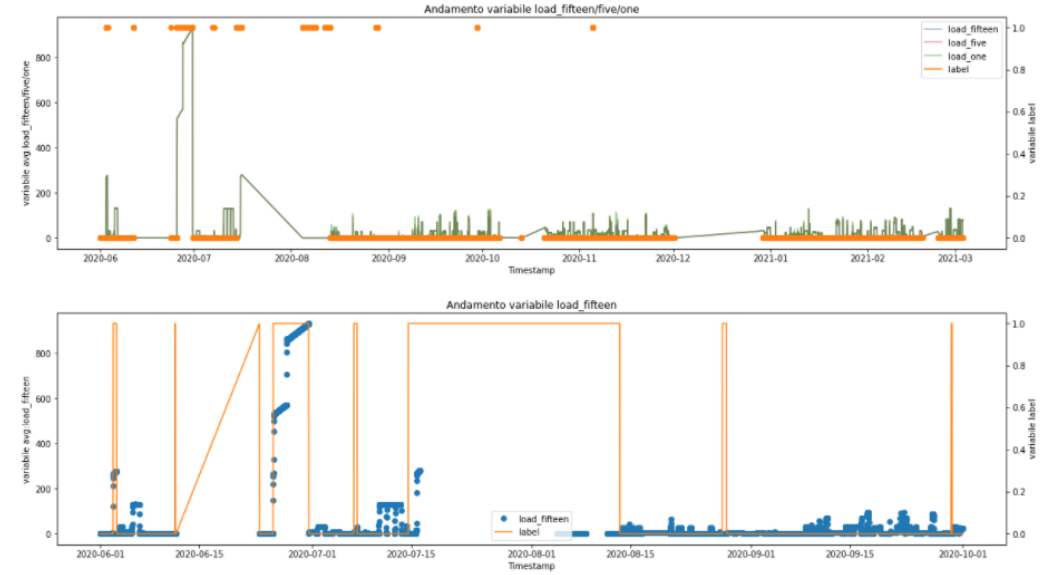

Figura 159 - carico medio nodo r254n01

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

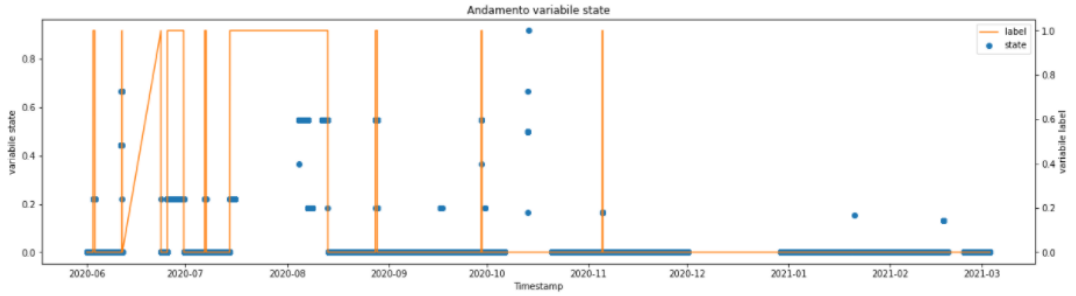

Figura 160 - andamento variabile state nodo r254n01

Nella tabella seguente è possibile osservare le informazioni del nodo più di rilievo.

	Shape	Primo camp.	Ultimo camp.	Stato anomalo	Salti	Numero anomalie	Media anomalie
r254n01	(17191, 234)	2020-05-31 22:30:00	2021-03-03 13:45:00	8%	Si	20	16 h 30 min

3.16 Nodo r256n05

Le letture in questo nodo vengono riportate in una tabella di 11140 righe e 234 colonne. Il primo campionamento, risale al 2020-05-31 22:00:00 mentre l'ultimo risale al 2021-03-03 13:45:00. Si possono individuare 11024 letture in cui il nodo risulta in stato *normale* e 116 in cui risulta in stato *anomalo*. La distribuzione delle anomalie su questo nodo è così rappresentata:

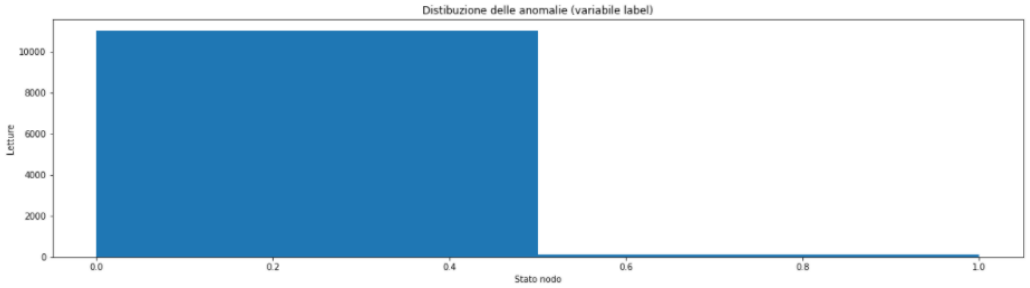

Figura 161 - distribuzione delle anomalie in base alla variabile label nodo r256n05

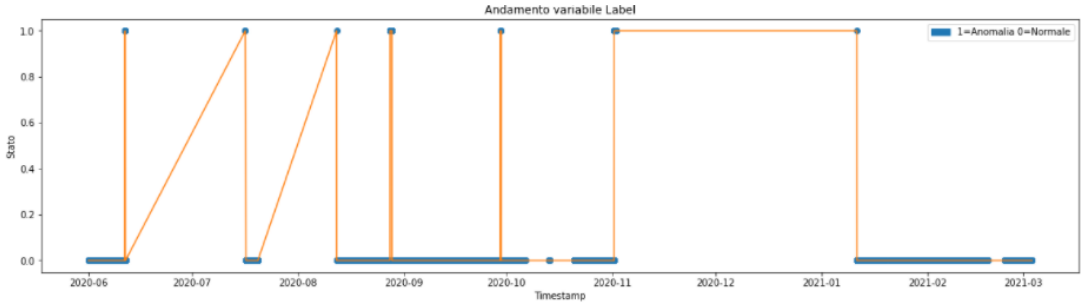

Figura 162 - andamento varibaile label nodo r256n05

Si è provato ad effettuare uno zoom nell'intervallo 2020/06 e 2020/10, poiché si riscontrano vari cambiamenti tra lo stato anomalo/normale, usando le misure di una sola GPU per poter osservare meglio il comportamento di questa feature. Quello che si può notare è che in un momento di anomalia la temperatura varia poco e si mantiene relativamente bassa rispetto al solito. Se si vanno a considerare le temperature delle memorie delle GPU, e le temperature e la potenza dei core nelle CPU socket, velocità dei fan e energia consumata, i risultati saranno simili.

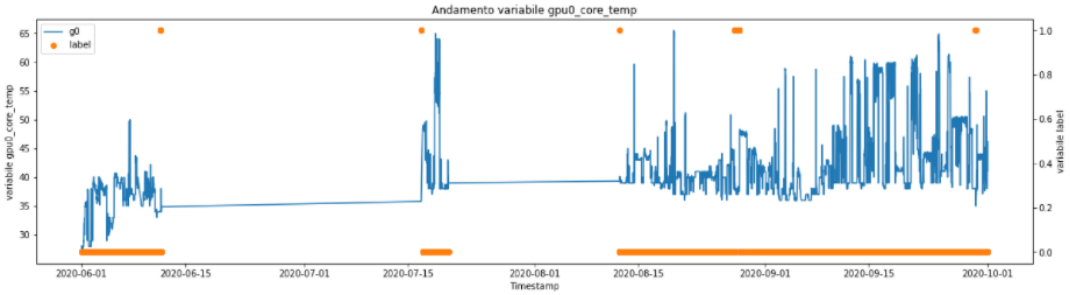

Figura 163 - andamento variabile gpu_core_temp nodo r256n05

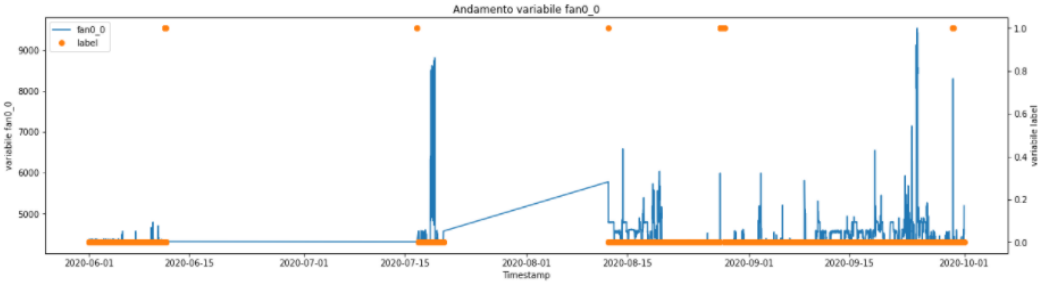

Figura 164 - andamento variabile fan0_0 nodo r256n05

Osservando la percentuale di CPU idle si può notare che nei periodi di anomalia la CPU è sempre idle.

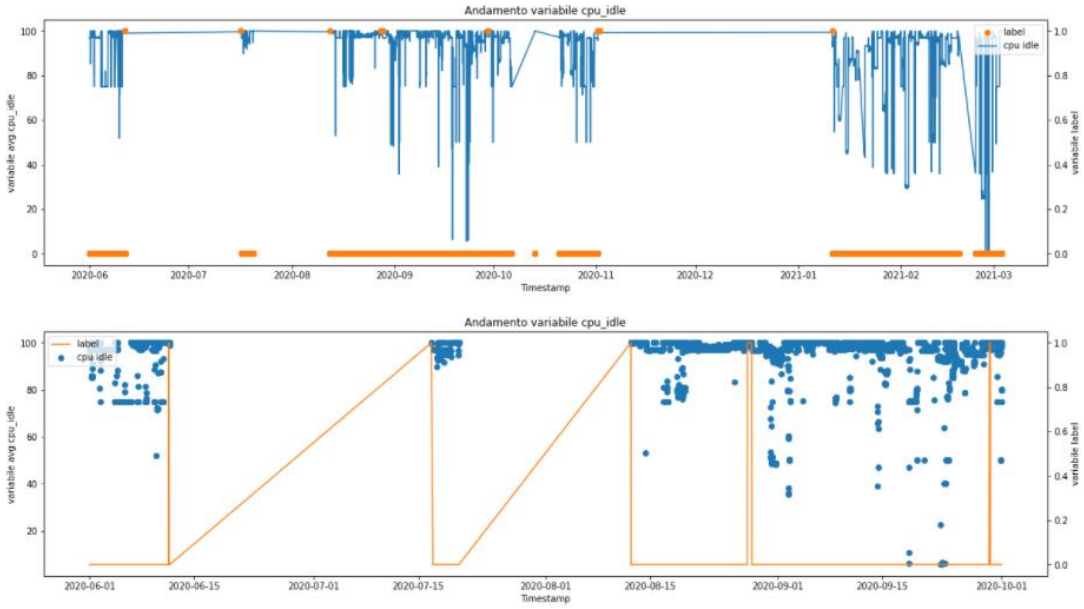

Figura 165 - andamento variabile cpu_idle nodo r256n05

Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la metrica state, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

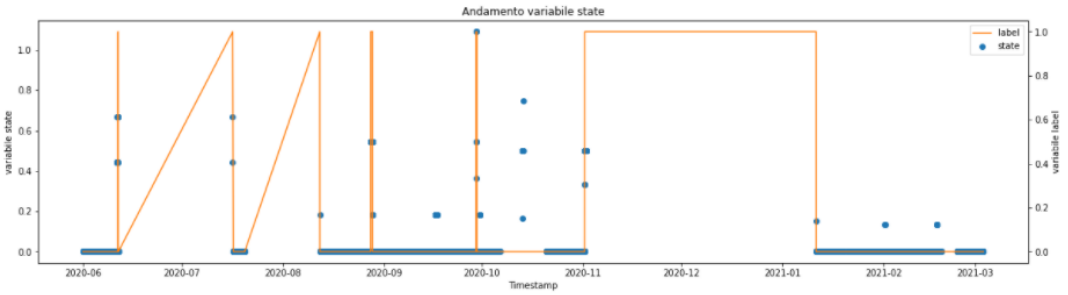

Figura 166 - andamento variabile state nodo r256n05

Nella tabella seguente è possibile osservare le informazioni del nodo più di rilievo.

	Shape	Primo camp.	Ultimo camp.	Stato anomalo	Salti	Numero anomalie	Media anomalie
r256n05	(11140, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	1%	Si	9	3 h

3.17 Confronto generale tra i nodi

Osservando i dati relativi ai nodi, con lo scopo di individuare alcuni collegamenti tra le anomalie e le feature monitorate, si è notato che spesso le anomalie si presentano nello stesso intervallo temporale, specialmente se i nodi appartengono allo stesso rack e inoltre ci sono alcuni intervalli per cui tutti i nodi presentano anomalie.
Inoltre in corrispondenza delle anomalie si nota che la ***temperatura*** dei vari componenti del sistema, varia poco e si mantiene relativamente bassa rispetto al solito. Se si vanno a considerare le temperature delle memorie delle GPU, le temperature e la ***potenza*** dei core nelle CPU socket, la ***velocità dei fan*** e ***l'energia consumata***, anche questi varieranno poco e si terranno relativamente bassi.

Durante i periodi anomali anche le letture riguardati la ***CPU idle*** hanno delle costanti, infatti durante le anomalie la CPU risulta idle al 100% in tutto il periodo. Questo comportamento è giustificato dal fatto che quando un sys admin riceve segnalazioni per un nodo che si sta comportando male (ad esempio le applicazioni del cliente risultano più lente, alcuni servizi non risultano disponibili, ecc…), mette il nodo in stato DOWN+DRAIN usando un tool chiamato Nagios e in questa fase il nodo è sottoposto ad ulteriori controlli da parte del sys admin, e nuovi job non possono esservi eseguiti finché non viene rimesso in stato UP.
Il ***carico medio*** durante le anomalie può avere dei picchi, che si discostano dal normale valore del carico. Inoltre, si può notare che in prossimità di una anomalia il numero dei ***pacchetti di input*** può superare di molto quello dei pacchetti di output.
Un comportamento importate in relazione allo stato anomalo/normale del sistema lo si trova analizzando il la feature ***state***, che è 0 se il sistema funziona, altrimenti assume un altro valore e man mano che il sistema torna alla normalità il valore assunto si abbassa.

Di seguito viene riportata una tabella che riassume e mette a confronto tutte le caratteristiche principali dei nodi presi in analisi. Come si può notate da questa tabella il nodo con percentuale maggiore (39%) di anomalie è **r210n05,** mentre quello con la media della durata dell'anomalia maggiore (114 h e 30 min) è il **r205n20**.
Altri nodi con una percentuale di anomalie molto alta sono: **r239n12** (33%), **r215n05** (29%), **r205n02** (27%), **r205n20** (18%).
Nodi con una media di durata dell'anomalia molto alta sono: **r215n05** (104 h 45 min), **r205n02** (86 h e 30 min), **r206n02** (57 h e 30 min).

	Shape	Primo camp.	Ultimo camp.	Stato anomalo	Salti	Numero anomalie	Media anomalie
RACK 205							
n02	(8384,234)	2020-07-22 15:45:00	2021-01-19 09:15:00	27%	Si	7	86 h e 30 min
n17	(5330,234)	2020-05-31 22:15:00	2021-02-18 14:45:00	2%	Si	2	15 h e 15 min
n20	(14230, 234)	2020-07-02 08:30:00	2021-03-03 13:45:00	18%	Si	6	**114 h e 30 min**
RACK 206							
n02	(10781, 234)	2020-07-06 12:30:00	2021-03-03 13:45:00	8%	Si	4	57 h e 30 min
n14	(7845, 234)	2020-05-31 22:15:00	2021-03-03 13:45:00	0.8%	Si	3	7 h 25min
n17	(11664, 234)	2020-05-31 22:15:00	2021-03-03 13:45:00	0.6%	Si	4	6 h
n18	(11681, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	0.5%	Si	4	3 h 45 min
n19	(11703, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	0.6%	Si	4	4 h
RACK 208							
n02	(11382, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	0.1%	Si	5	36 min
n04	(6114, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	0.1%	Si	1	1h 30 min
n07	(10405, 234)	2020-07-10 11:30:00	2021-03-03 13:45:00	0.1%	Si	5	21 min
n12	(13546, 234)	2020-05-31 22:15:00	2021-03-03 13:45:00	0.2%	Si	8	34 min
RACK 210							
r210n05	(13667, 234)	2020-05-31 22:15:00	2021-03-03 13:45:00	**39%**	Si	27	51 h
RACK 211							
r211n19	(6670, 234)	2020-05-31 22:15:00	2021-03-03 13:45:00	16%	Si	9	31 h

RACK 212							
r212n06	(10244, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	0.4%	Si	14	38 min
RACK 215							
r215n05	(17576, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	29%	Si	13	104 h 45 min
RACK 218							
r218n03	(8300, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	12%	Si	5	37 h 30 min
RACK 224							
r224n20	(10792, 234)	2020-05-31 22:15:00	2021-03-03 13:45:00	1,5%	Si	11	3 h 30 min
RACK 239							
r239n12	(13495, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	33%	Si	27	43 h 30 min
RACK 243							
r243n11	(12182, 234)	2020-05-31 22:15:00	2021-03-03 13:45:00	4%	Si	8	14 h 15 min
RACK 249							
r249n06	(14932, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	3%	Si	9	12 h 30 min
RACK 253							
r253n06	(9862, 234)	2020-06-16 13:00:00	2021-03-03 13:45:00	2%	Si	8	5 h 30 min
RACK 254							
r254n01	(17191, 234)	2020-05-31 22:30:00	2021-03-03 13:45:00	8%	Si	20	16 h 30 min
RACK 256							
r256n05	(11140, 234)	2020-05-31 22:00:00	2021-03-03 13:45:00	1%	Si	9	3 h

L’intervallo temporale di campionamento è dal 2020-05-31 22:00:00 al 2021-03-03 13:45:00, ma non tutti i nodi contengono tutto l’intervallo.

Di seguito vengono riportati gli intervalli in cui sono state riscontrate anomalie nei corrispondenti nodi. Come è possibile notare i periodi con maggiori anomalie sono: 2020-06-11, 2020-07-16, 2020-09-29.

	r205n02	r205n17	r205n20	r206n02	r206n14	r206n17	r206n18	r206n19
2020-06-02 al 2020-06-03								
2020-06-08 al 2020-06-10								
2020-06-11 al 2020-06-11		✖			✖	✖	✖	✖
2020-06-16 al 2020-06-16								
2020-06-18 al 2020-06-18								
2020-06-20 al 2020-06-22								
2020-06-25 al 2020-06-30								
2020-07-06								
2020-07-06 al 2020-07-07				✖				
2020-07-14 al 2020-07-16								
2020-07-07 al 2020-07-08					✖	✖	✖	✖
2020-07-10 al 2020-07-10								
2020-07-16 al 2020-07-16				✖				
2020-07-22 al 2020-08-08	✖		✖					
2020-08-04 al 2020-08-08								
2020-08-11 al 2020-08-13								
2020-08-13 al 2020-08-17								
2020-08-27 al 2020-08-28								
2020-08-28								
2020-08-29 al 2020-09-09								
2020-09-02 al 2020-09-03								
2020-09-05 al 2020-09-05								
2020-09-10 al 2020-09-14								

	r205n02	r205n17	r205n20	r206n02	r206n14	r206n17	r206n18	r206n19
2020-09-25 al 2020-09-25								
2020-09-28 al 2020-09-29								
2020-09-29 al 2020-10-06	✖		✖	✖	✖	✖	✖	✖
2020-09-30 al 2020-09-30								
2020-09-30 al 2020-10-01								
2020-10-20 al 2020-12-01								
2020-10-31 al 2020-11-02				✖				
2020-11-01 al 2020-11-01								
2020-11-02 al 2020-11-02								
2020-11-05 al 2020-11-05								
2020-11-08 al 2020-12-01								
2020-12-01 al 2020-12-01			✖					
2020-12-23 al 2020-12-23	✖							
2020-12-29 al 2021-01-13								
2021-01-01 al 2021-01-08								
2021-01-14 al 2021-01-15			✖					
2021-01-18 al 2021-01-19		✖						
2021-01-21 al 2021-01-21								
2021-01-26 al 2021-01-26								
2021-01-27 al 2021-01-28								
2021-02-02 al 2021-02-05								
2021-02-14 al 2021-02-15								
2021-02-24 al 2021-02-26			✖					
2021-03-01 al 2021-03-03			✖					

	r208n02	r208n04	r208n07	r208n12	r210n05	r211n19	r212n06	r215n05
2020-06-02 al 2020-06-03								
2020-06-08 al 2020-06-10								
2020-06-11 al 2020-06-11	✖	✖		✖	✖	✖	✖	✖
2020-06-16 al 2020-06-16								
2020-06-18 al 2020-06-18								
2020-06-20 al 2020-06-22								✖
2020-06-25 al 2020-06-30								
2020-07-06								
2020-07-06 al 2020-07-07								
2020-07-14 al 2020-07-16								
2020-07-07 al 2020-07-08								
2020-07-10 al 2020-07-10	✖		✖					
2020-07-16 al 2020-07-16	✖		✖	✖		✖	✖	
2020-07-22 al 2020-08-08								
2020-08-04 al 2020-08-08								
2020-08-11 al 2020-08-13								
2020-08-13 al 2020-08-17								
2020-08-27 al 2020-08-28					✖			
2020-08-28				✖				
2020-08-29 al 2020-09-09								
2020-09-02 al 2020-09-03								✖
2020-09-05 al 2020-09-05								
2020-09-10 al 2020-09-14								
2020-09-22 al 2020-09-22				✖				

	r208n02	r208n04	r208n07	r208n12	r210n05	r211n19	r212n06	r215n05
2020-09-25 al 2020-09-25				✖				
2020-09-28 al 2020-09-29								
2020-09-29 al 2020-10-06	✖		✖	✖	✖	✖	✖	✖
2020-09-30 al 2020-09-30								✖
2020-09-30 al 2020-10-01			✖					
2020-10-20 al 2020-12-01					✖			✖
2020-10-31 al 2020-11-02								
2020-11-01 al 2020-11-01								
2020-11-02 al 2020-11-02								
2020-11-05 al 2020-11-05								
2020-11-08 al 2020-12-01								
2020-12-01 al 2020-12-01							✖	
2020-12-23 al 2020-12-23								
2020-12-29 al 2021-01-13					✖	✖		✖
2021-01-01 al 2021-01-08								
2021-01-14 al 2021-01-15					✖			
2021-01-18 al 2021-01-19								
2021-01-21 al 2021-01-21						✖		
2021-01-26 al 2021-01-26					✖			
2021-01-27 al 2021-01-28							✖	
2021-02-02 al 2021-02-05								✖
2021-02-14 al 2021-02-15						✖		
2021-02-24 al 2021-02-26								
2021-03-01 al 2021-03-03								

	r218n03	r224n20	r239n12	r243n11	r249n06	r253n06	r254n01	r256n05
2020-06-02 al 2020-06-03							✖	
2020-06-08 al 2020-06-10		✖						
2020-06-11 al 2020-06-11	✖	✖	✖	✖	✖		✖	✖
2020-06-16 al 2020-06-16						✖		
2020-06-18 al 2020-06-18						✖		
2020-06-20 al 2020-06-22								
2020-06-25 al 2020-06-30							✖	
2020-07-06							✖	
2020-07-06 al 2020-07-07								
2020-07-14 al 2020-07-16							✖	
2020-07-07 al 2020-07-08								
2020-07-10 al 2020-07-10								
2020-07-16 al 2020-07-16		✖	✖		✖	✖		✖
2020-07-22 al 2020-08-08								
2020-08-04 al 2020-08-08							✖	
2020-08-11 al 2020-08-13							✖	
2020-08-13 al 2020-08-17					✖			
2020-08-27 al 2020-08-28							✖	✖
2020-08-28							✖	
2020-08-29 al 2020-09-09			✖					
2020-09-02 al 2020-09-03								
2020-09-05 al 2020-09-05								
2020-09-10 al 2020-09-14				✖				
2020-09-22 al 2020-09-22								

	r218n03	r224n20	r239n12	r243n11	r249n06	r253n06	r254n01	r256n05
2020-09-25 al 2020-09-25					✖			
2020-09-28 al 2020-09-29						✖		
2020-09-29 al 2020-10-06	✖	✖	✖	✖	✖	✖	✖	✖
2020-09-30 al 2020-09-30			✖	✖	✖			
2020-09-30 al 2020-10-01				✖				
2020-10-20 al 2020-12-01								
2020-10-31 al 2020-11-02								
2020-11-01 al 2020-11-01								✖
2020-11-02 al 2020-11-02	✖							
2020-11-05 al 2020-11-05							✖	
2020-11-08 al 2020-12-01			✖					
2020-12-01 al 2020-12-01								
2020-12-23 al 2020-12-23								
2020-12-29 al 2021-01-13			✖					
2021-01-01 al 2021-01-08	✖							
2021-01-14 al 2021-01-15			✖					
2021-01-18 al 2021-01-19								
2021-01-21 al 2021-01-21								
2021-01-26 al 2021-01-26	✖		✖					
2021-01-27 al 2021-01-28								
2021-02-02 al 2021-02-05								
2021-02-14 al 2021-02-15								
2021-02-24 al 2021-02-26								
2021-03-01 al 2021-03-03								

CAPITOLO 4

Le tecniche di Machine Learning e i modelli usati per il caso di studio

4.1 Machine Learning

Quando si parla di machine learning ci si riferisce a differenti meccanismi che permettono a una macchina intelligente di migliorare le proprie capacità e prestazioni nel tempo. La macchina, quindi, sarà in grado di imparare a svolgere determinati compiti migliorando, tramite l'esperienza, le proprie capacità, le proprie risposte e funzioni. Alla base dell'apprendimento automatico ci sono una serie di differenti algoritmi che, partendo da nozioni primitive, sapranno prendere una specifica decisione piuttosto che un'altra o effettuare azioni apprese nel tempo. A coniare per primo il termine machine learning fu Arthur Lee Samuel, scienziato americano pioniere nel campo dell'Intelligenza Artificiale, nel 1959 ma Tom M. Mitchell ha fornito una definizione più formale: "*Si dice che un programma per computer apprenda dall'esperienza E rispetto ad alcune classi di attività T e dalle prestazioni P, se le sue prestazioni in attività in T, misurate da P, migliorano con l'esperienza E*." [15].

L'apprendimento quindi è da intendere come il processo che permette di svolgere l'attività. Le attività di machine learning sono descritte in termini di come il sistema di apprendimento automatico dovrebbe elaborare un ***esempio***. Un esempio è una raccolta di elementi che sono stati misurati quantitativamente da un oggetto o un evento che vogliamo che il sistema di apprendimento automatico elabori [16]. Per valutare le capacità di un algoritmo di apprendimento automatico, è necessario progettare una misura quantitativa delle sue ***prestazioni***. Di solito questa misura di prestazione è specifica per il compito T svolto dal sistema. Un metodo per misurare le prestazioni è l'accuratezza, cioè valutare la proporzione di esempi per cui il modello produce l'output corretto. Specularmente è possibile anche usare il tasso di errore, cioè valutare la proporzione di esempi per cui il modello restituisce l'output sbagliato. Per effettuare queste valutazioni spesso viene usato un insieme di dati "nuovi" al modello creato, il ***test set***. Gli algoritmi di machine learning possono essere suddivisi in due sottocategorie a seconda del fatto che si diano al computer esempi completi da utilizzare come indicazione per eseguire il compito richiesto (***apprendimento supervisionato***) oppure che si

lasci lavorare il software senza alcun "aiuto" (***apprendimento non supervisionato***). Il machine learning può essere impiegato in molteplici ambiti, in questo elaborato ci si focalizzerà sul suo uso per ***l'anomaly detection***. Per anomaly detection si intende il problema di trovare modelli nei dati che non sono conformi al comportamento previsto. Questi modelli non conformi sono spesso indicati come anomalie, valori anomali, osservazioni discordanti, eccezioni, aberrazioni, sorprese, peculiarità o contaminanti in diversi domini di applicazione. Di questi, anomalie e valori anomali sono due termini utilizzati più comunemente nel contesto del rilevamento delle anomalie [10]. Le caratteristiche delle anomalie differiscono di molto rispetto al normale comportamento e avvengono raramente. Le principali tecniche usate per questo scopo sono Supervised Learning, Unsupervised Learning, Semi-Supervised Learning e Reinforcement Learning e verranno analizzate di seguito.

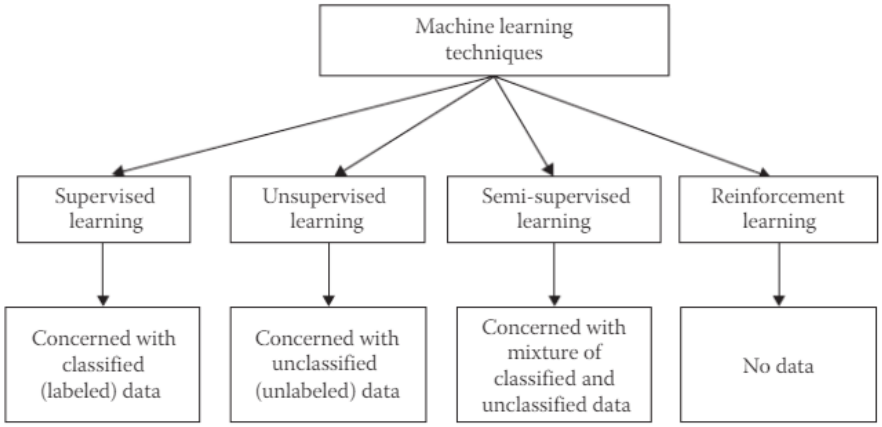

Figura 167 - Diverse tecniche di apprendimento automatico e i relativi dati richiesti [17]

4.1.1 Supervised Learning

Il Supervised Learning [18], noto anche come apprendimento automatico supervisionato, usa un set di dati etichettati per addestrare algoritmi che classificano i dati o prevedono i risultati in modo accurato. L'apprendimento supervisionato utilizza un set di formazione per insegnare ai modelli a produrre l'output desiderato. Questo set di dati di addestramento include input e output corretti, che consentono al modello di apprendere nel tempo. L'algoritmo misura la sua accuratezza attraverso la funzione di perdita, regolando i vari pesi, fino a quando l'errore non è stato sufficientemente ridotto al minimo. Nell'apprendimento supervisionato il primo passo è trattare il set di dati, al fine di eseguire una migliore formazione su di esso. I due approcci che si seguono sono: interpellare un esperto del campo o usare la "forza bruta". Con approccio a "forza bruta" si intende misurare tutto ciò che è disponibile nella speranza che le caratteristiche

giuste (informative, rilevanti) possano essere isolate [19]. Due gruppi di algoritmi rientrano nell'ambito dell'apprendimento supervisionato, ***regressione*** e ***classificazione.***

La ***classificazione*** è il compito di approssimare una funzione di mappatura (f) dalle variabili di input (X) a variabili di output discrete/categoriali (y). Le variabili di output sono spesso chiamate etichette, categorie o classi. La funzione di mappatura prevede la classe o la categoria per una data osservazione.

La ***regressione*** è il compito di approssimare una funzione di mappatura (f) da variabili di input (X) a una variabile di output continua (y). Una variabile di uscita continua è un valore reale, come un valore intero o in virgola mobile.

Per la classificazione dei dati, un buon numero di tecniche, alcune basate sulla logica, altre basate sulla statistica, sono state sviluppate dai ricercatori. Nelle sezioni successive verranno analizzate le tecniche principali usate in particolare per l'anomaly detection.

4.1.1.1 Alberi decisionali

Gli alberi decisionali (Decision Trees, DT) sono alberi che classificano le istanze ordinandole in base ai valori delle caratteristiche, dove ogni nodo in un albero decisionale rappresenta una caratteristica in un'istanza da classificare e ogni ramo rappresenta un valore che il nodo può assumere [20]. Le istanze vengono classificate a partire dal nodo radice e ordinate in base ai valori delle caratteristiche. Tra i principali vantaggi apportati dall'utilizzo degli alberi decisionali si può ritrovare la produzione di risultati intesivi, la facilità di comprensione e la strutturazione della conoscenza in un modo ben organizzato [21]. Di seguito viene illustrato un esempio di albero decisionale (Figura 169) per il training set della tabella in Figura 168 [19].

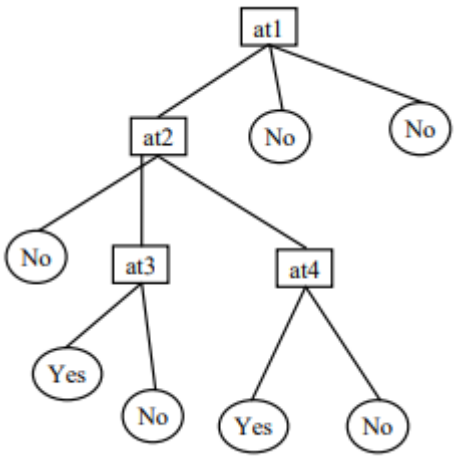

Figura 169 - albero decisionale del training set in Figura 168

at1	at2	at3	at4	Class
a1	a2	a3	a4	Yes
a1	a2	a3	b4	Yes
a1	b2	a3	a4	Yes
a1	b2	b3	b4	No
a1	c2	a3	a4	Yes
a1	c2	a3	b4	No
b1	b2	b3	b4	No
c1	b2	b3	b4	No

Figura 168 - Training set

4.1.1.2 Classificatori Naive Bayes

Le reti bayesiane sono ampiamente utilizzate per eseguire attività di classificazione. Queste reti, (NBN) sono reti bayesiane molto semplici composte da grafi aciclici diretti con un solo genitore (che rappresenta il nodo non osservato) e diversi figli (che corrispondono ai nodi osservati) con una forte ipotesi di indipendenza tra i nodi figli nel contesto del loro genitore [22]. Il modello di indipendenza (Naive Bayes) si basa sulla stima:

$$R = \frac{P(i|X)}{P(j|X)} = \frac{P(i)P(X|i)}{P(j)P(X|j)} = \frac{P(i)\pi P(X_r|i)}{P(j)\pi P(X_r|j)}$$

Confrontando queste due probabilità, la probabilità maggiore indica l'etichetta della classe che è più probabile che sia l'etichetta effettiva (se R> 1: i è l'etichetta predetta, altrimenti viene predetta j) [23]. I collegamenti in un modello Naive Bayes sono diretti dall'output all'input, il che conferisce al modello la sua semplicità, poiché non vi sono interazioni tra gli input, se non indirettamente tramite l'output. Un vantaggio importante di questa soluzione è che necessita si un numero non molto elevato di dati di training per stimare i parametri necessari alla classificazione.

4.1.1.3 Support Vector Machines

L'obiettivo dell'algoritmo Support Vector Machine è trovare l'iperpiano in uno spazio N-dimensionale (N - il numero di caratteristiche) che classifica distintamente i punti dati. Gli iperpiani sono limiti decisionali che aiutano a classificare i punti dati. I punti dati che cadono su entrambi i lati dell'iperpiano possono essere attribuiti a classi diverse. La dimensione dell'iperpiano dipende dal numero di caratteristiche. Ci sono numerosi vantaggi che derivano dell'utilizzo di SVM come l'efficienza nello spazio ad alta dimensione, l'utilizzo di un sottoinsieme di punti di addestramento nella funzione decisionale (chiamati vettori di supporto), che porta ad un uso della memoria efficiente, la versatilità, in quanto ospita diverse funzioni del kernel che possono essere specificate per la funzione decisionale. I vettori di supporto sono punti dati che sono più vicini all'iperpiano e influenzano la posizione e l'orientamento dell'iperpiano. Quindi, usando i vettori di supporto, si massimizza il margine del classificatore. L'eliminazione dei vettori di supporto cambierà la posizione dell'iperpiano. I punti di due classi (grigio, blu in Figura 4) perfettamente separabili da varie linee (nero) illustrano il concetto di

margine (evidenziazione gialla), che è la regione rettangolare che si estende dalla linea di separazione al punto perpendicolarmente più vicino [24].

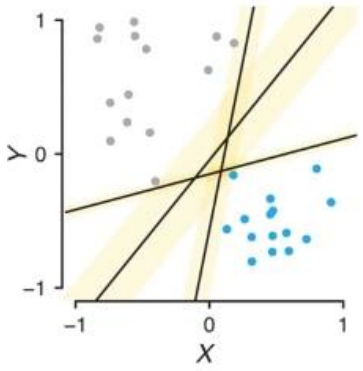

Figura 170 - classi linearmente separabili

4.1.1.4 Reti neurali artificiali

I metodi di apprendimento delle reti neurali (Artificial Neural Network, ANN) forniscono un approccio robusto per l'approssimazione delle funzioni target a valori reali, discreti e vettoriali. Ispirate alle reti neurali biologiche, le ANN sono sistemi di calcolo massivamente paralleli costituiti da un numero estremamente elevato di processori semplici con molte interconnessioni [25]. Quando si crea un modello funzionale del neurone biologico, ci sono tre componenti di fondamentale importanza. Innanzitutto, le sinapsi del neurone sono modellate come ***pesi***. La forza della connessione tra un input e un neurone è rilevata dal valore del peso. Gli altri due componenti modellano l'attività effettiva all'interno della cellula neuronale. Un sommatore somma tutti gli ingressi modificati dai rispettivi pesi. Questa attività è denominata ***combinazione lineare***. Infine, una ***funzione di attivazione*** controlla l'ampiezza dell'uscita del neurone. Un intervallo accettabile di output è solitamente compreso tra 0 e 1, o -1 e 1 [26].

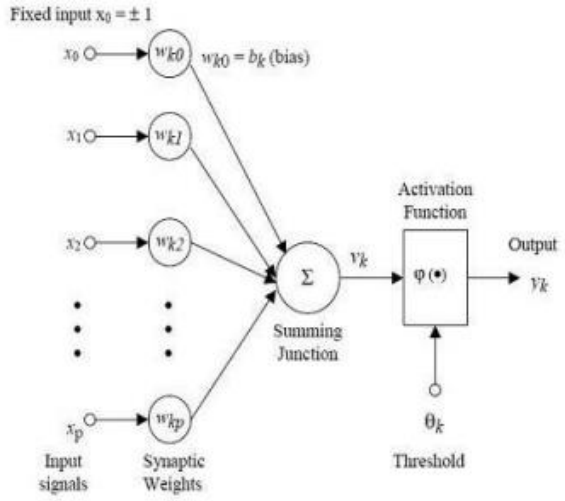

Figura 171 - modello rete neurale

Le reti neurali feedforward sono state il primo tipo di rete neurale artificiale inventata e sono tre le più semplici da realizzare. La loro caratteristica è che le connessioni tra le unità non formano un ciclo, infatti sono chiamate feedforward perché le informazioni viaggiano solo in avanti nella rete (senza loop), prima attraverso i nodi di ***input***, poi attraverso i nodi ***hidden*** (se presenti) e infine attraverso i nodi di ***output***, formando una struttura a ***layer*** [27].

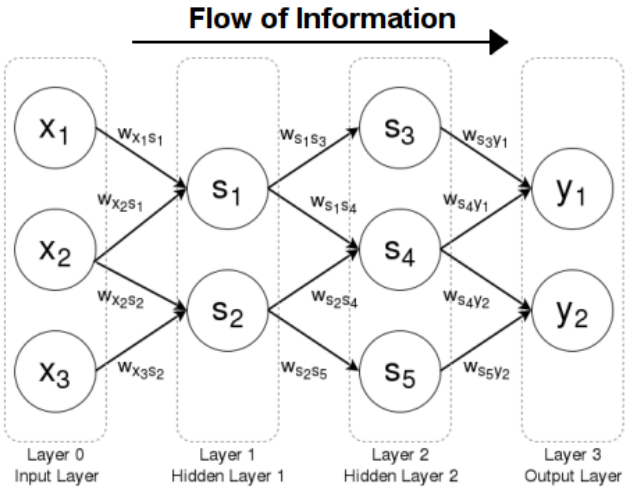

Figura 172 - rete feedforward

La rete impara a riconoscere una serie di configurazioni di input desiderati e opera in due fasi distinte:

- Fase di apprendimento, durante la quale memorizza le informazioni desiderate tramite esempi
- Fase di evoluzione, in cui recupera le informazioni memorizzate

L'algoritmo di apprendimento di sviluppa in vari passaggi. Inizialmente deve essere fornito alla rete, un insieme di esempi, chiamato **training set**:

$$TS = \{(X_k, Y_{dk}), K = 1 \dots . m$$

I pesi della rete vengono inizializzati con valori random W_i. La rete ha lo scopo di classificare i dati di un nuovo data set. Quindi prende in ingresso una coppia di valori (X_k, Y_{dk}) e computa la risposta Y_k attraverso la rete. La rete valuta la risposta attraverso la funzione di errore e aggiorna i pesi con la delta rule: $\Delta w = \eta \delta x$. Questo processo viene ripetuto finché $Y_k = Y_{dk}\ \forall k \in [1 \dots m]$. Esistono vari algoritmi di apprendimento, ma quello più usato è ***Back Propagation*** (BP). Questo algoritmo ha come obiettivi **l'apprendimento** attraverso un training

set, la **convergenza** cioè ridurre l'errore globale E alla variazione dei pesi, in modo che $E < \varepsilon$ e la **generalizzazione** cioè assicurarsi che la rete lavori bene con esempi mai visti prima. Per ridurre l'errore sulla variazione dei pesi, si adotta un metodo di discesa del gradiente. L'algoritmo segue genericamente i seguenti passi:

1. Si inizializzano i pesi in modo random
2. do {
3. si inizializza l'errore globale $E = 0$
4. per ogni coppia $(X_k, Y_{dk}) \in TS$ {
5. si computa Y_k e l'errore E_k
6. si calcola δ_j per l'output layer con $\delta_j = (Y_{dk} - Y_k) f'(net_k)$
7. si calcola δ_i per l'hidden layer con $\delta_i = f'(net_i) \sum_{j=1}^{n} \delta_j w_{ji}$
8. si aggiorna il peso della rete con $\Delta w = \eta \delta x$
9. si aggiorna l'errore globale $E = E + E_k$
10. } while $(E < \varepsilon)$

In sintesi BP Per ciascun neurone, calcola quale dovrebbe essere stato l'output e un fattore di ridimensionamento, quanto più basso o più alto deve essere regolato in modo da corrispondere all'output desiderato. Questo è l'errore locale. Regola i pesi di ciascun neurone per ridurre l'errore locale. Assegnare la "responsabilità" per l'errore locale ai neuroni al livello precedente, dando maggiore responsabilità ai neuroni collegati da pesi più forti [28].
Esistono vari tipi di ANN, oltre le classiche feedforward, come Recurrent Neural Network, Long Short-Term Memory, Convolutional Neural Network…
La rete ***neurale di convoluzione*** (CNN) contiene una disposizione tridimensionale di neuroni, invece della matrice bidimensionale standard. Il primo strato è chiamato strato convoluzionale. Ogni neurone nello strato convoluzionale elabora solo le informazioni provenienti da una piccola parte del campo visivo. Le caratteristiche di input sono prese in batch come un filtro. La rete comprende le immagini in parti e può calcolare queste operazioni più volte per completare l'elaborazione completa dell'immagine.
L'elaborazione prevede la conversione dell'immagine dalla scala RGB o HSI alla scala di grigi. Risulta particolarmente adatta in ambiti come image processing, computer vision. [29].

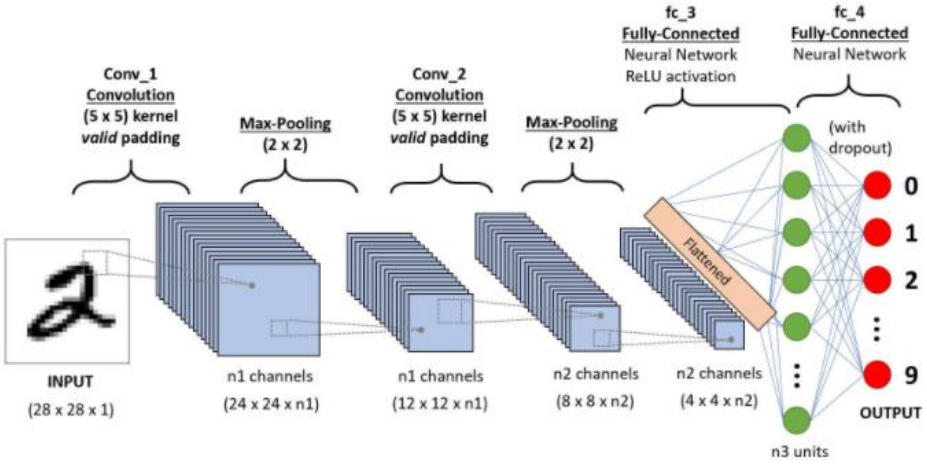

Figura 173 - CNN per classificare un numero scritto a mano [30]

4.1.2 Unsupervised Learning

Il **Unsupervised Learning**, noto anche come apprendimento automatico senza supervisione, utilizza algoritmi di apprendimento automatico per analizzare e raggruppare set di dati senza etichetta. Questi algoritmi scoprono modelli nascosti o raggruppamenti di dati senza la necessità dell'intervento umano. La sua capacità di scoprire somiglianze e differenze nelle informazioni lo rende la soluzione ideale per l'analisi esplorativa dei dati e il riconoscimento delle immagini. La distinzione tra algoritmi supervisionati e non supervisionati non è formalmente e rigidamente definita perché non esiste un test oggettivo per distinguere se un valore è una caratteristica o un obiettivo fornito da un supervisore. Informalmente, l'apprendimento non supervisionato si riferisce alla maggior parte dei tentativi di estrarre informazioni da una distribuzione che non richiedono lavoro umano per annotare gli esempi. Il termine è solitamente associato alla stima della densità, all'apprendimento dell'estrazione di campioni da una distribuzione, all'apprendimento del denoising dei dati da una distribuzione, alla ricerca di una varietà vicina ai dati o al raggruppamento dei dati in gruppi di esempi correlati [16]. Un obiettivo dell'apprendimento non supervisionato è trovare la "migliore" rappresentazione dei dati. Per "migliore" si possono intendere cose diverse, ma in generale si cerca una rappresentazione che conservi quante più informazioni possibili su x ottemperando a qualche penalità o vincolo volto a mantenere la rappresentazione più semplice o più accessibile di x stesso. Ci sono vari algoritmi che si possono impiegare per tale scopo. Tra i più semplici e usati si ritrovano il ***K-means clustering*** e **DBSCAN**.

4.1.2.1 K-means clustering

L'obiettivo di K-means è raggruppare punti dati simili e scoprire i modelli sottostanti. Per raggiungere questo scopo, K-means cerca un numero fisso (k) di cluster in un set di dati. Quindi è necessario definire un numero di destinazione k, che si riferisce al numero di centroidi necessari nel set di dati. Un centroide è la posizione immaginaria o reale che rappresenta il centro del cluster. Ogni dato viene allocato in ciascuno dei cluster riducendo la somma dei quadrati nel cluster. L'algoritmo K-means identifica k numero di centroidi e quindi alloca ogni punto dati al cluster più vicino, mantenendo i centroidi il più piccoli possibile. Per elaborare i dati di apprendimento, l'algoritmo K-means inizia con un primo gruppo di centroidi selezionati casualmente, che vengono utilizzati come punti di inizio per ogni cluster, quindi esegue calcoli iterativi per ottimizzare le posizioni dei centroidi. Interrompe la creazione e l'ottimizzazione dei cluster quando i centroidi si sono stabilizzati cioè non c'è alcun cambiamento nei loro valori perché il clustering ha avuto successo e il numero definito di iterazioni è stato raggiunto [31].

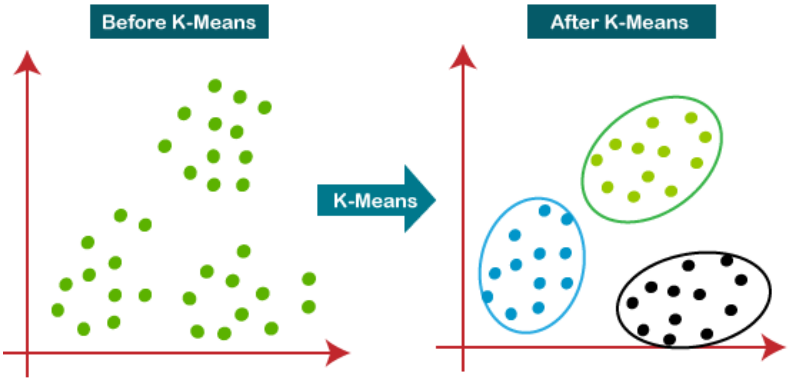

Figura 174 - K-means clustering [32]

4.1.2.2 DBSCAN

Il DBSCAN (Density-Based Spatial Clustering of Applications with Noise) è un metodo di clustering proposto nel 1996 da Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu. È basato sulla densità perché connette regioni di punti con densità sufficientemente alta. Questo algoritmo stima la densità attorno a ciascun punto (item) contando il numero di punti in un intorno ε specificato dall'utente, ed applica delle soglie chiamate ***minPts*** per identificare i punti core, border e noise. In un secondo passaggio, i punti core sono riuniti in un cluster, se sono **density-reachable** (raggiungibili per densità, cioè se esiste una catena di punti core in cui ogni punto ricade all'interno dell'eps-intorno del successivo). Infine i punti di bordo sono

assegnati ai cluster. Per gli obiettivi del clustering DBSCAN, i punti (item) sono classificati come core point, density-reachable o outlier, in base alle seguenti regole:

- Un punto *p* è un punto core se almeno *minPts* punti sono entro la distanza ε da esso. Questi punti sono detti direttamente raggiungibili da *p*.
- Un punto *q* è direttamente raggiungibile da *p* se il punto *q* è entro la distanza ε dal punto *p*, con *p* che deve essere un core point.
- Un punto *q* è raggiungibile da *p* se esiste un percorso $p_i \dots p_n$ con $p_i = p$ e $p_n = q$, dove ogni $p_i + 1$ è direttamente raggiungibile da p_i.
- Tutti i punti non raggiungibili da altri punti sono outlier.

Se *p* è un core point, allora questo forma un cluster insieme a tutti gli altri punti (core o non-core) che sono raggiungibili da esso. Ogni cluster contiene almeno un punto core; punti non-core possono essere parte di un cluster, ma ne formano il bordo (edge), poiché non possono essere usati per raggiungere altri punti [33].

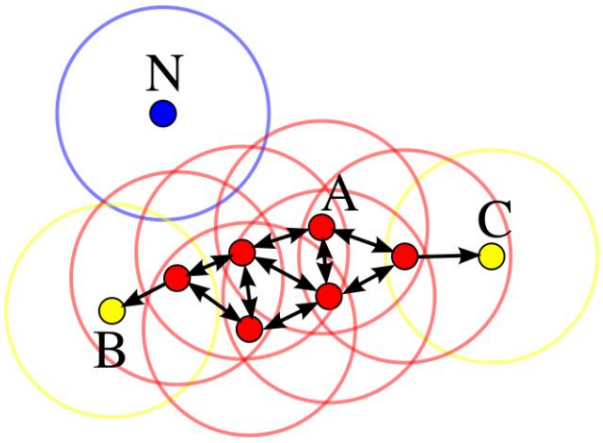

Figura 175 - DBSCAN [34]

4.1.3 Reinforcement Learning

Il **Reinforcement Learning** (apprendimento per rinforzo) rappresenta probabilmente il sistema di apprendimento più complesso, che prevede che la macchina sia dotata di sistemi e strumenti in grado di migliorare il proprio apprendimento e, soprattutto, di comprendere le caratteristiche dell'ambiente circostante. In questo caso, quindi, alla macchina vengono forniti una serie di elementi di supporto, quali sensori, telecamere, GPS eccetera, che permettono di rilevare quanto avviene nell'ambiente circostante ed effettuare scelte per un migliore adattamento all'ambiente

intorno a loro [35]. Nel modello standard di apprendimento per rinforzo, un agente è connesso al suo ambiente tramite la percezione e l'azione. Ad ogni fase dell'interazione l'agente riceve come input, i, qualche indicazione dello stato attuale, s, dell'ambiente; l'agente sceglie quindi un'azione, a, da generare come output. L'azione cambia lo stato dell'ambiente e il valore di questa transizione di stato viene comunicato all'agente attraverso un segnale di rinforzo scalare, r. Il comportamento dell'agente, B, dovrebbe scegliere azioni che tendono ad aumentare la somma dei valori di lungo periodo del segnale di rinforzo. Può imparare a farlo nel tempo per tentativi ed errori sistematici, guidati da un'ampia varietà di algoritmi. Formalmente, il modello è costituito da:

- un insieme discreto di stati di ambiente, S;
- un insieme discreto di azioni dell'agente, A;
- un insieme di segnali di rinforzo scalare; tipicamente {0,1}, o numeri reali. [36]

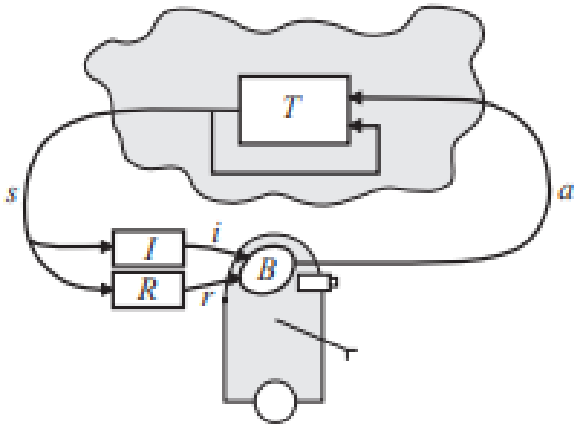

Figura 176 - modello renforcement learning

4.1.4 Semi-Supervised Learning

Il **Semi-Supervised Learning** si riferisce a un problema di apprendimento coinvolge una piccola porzione di esempi etichettati e un gran numero di esempi senza etichetta da cui un modello deve imparare e fare previsioni su nuovi esempi, è a metà strada tra l'apprendimento supervisionato e non supervisionato [37]. Data la poca necessità di dati etichettati, questa tecnica risulta essere molto usata in ambito industriale in quanto non si hanno molti dati classificati.

4.2 Modelli usati per il caso di studio: gli Autoencoders

Gli autoencoders sono reti neurali con lo scopo di generare nuovi dati dapprima comprimendo l'input in uno spazio di variabili latenti e, successivamente, ricostruendo l'output sulla base delle informazioni acquisite. Questa tipologia di network è composta da due parti:

- Encoder: la parte della rete che comprime l'input in uno spazio di variabili latenti e che può essere rappresentato dalla funzione di codifica $h = f(x)$.
- Decoder: la parte che si occupa di ricostruire l'input sulla base delle informazioni precedentemente raccolte. È rappresentato dalla funzione di decodifica $r = g(h)$.

L'autoencoder nel suo complesso può quindi essere descritto dalla funzione $d(f(x)) = r$ dove r è quanto più simile all'input originale x [38].

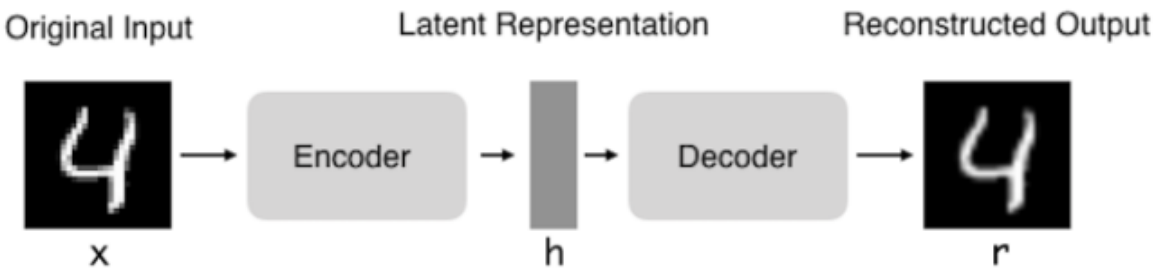

Figura 177 - schema base di un autoencoder

Esistono varie tipologie di autoencoder [16]:

- Un autoencoder la cui dimensione del code layer (h) è inferiore della dimensione dell'input layer (x) è chiamato **undercomplete**. Il suo obiettivo è catturare le caratteristiche più importanti presenti nei dati. Minimizza la funzione di perdita penalizzando $g(f(x))$ per essere diverso dall'ingresso x. Tra i vantaggi apportati si sottolinea che non necessitano di alcuna regolarizzazione in quanto massimizzano la probabilità dei dati anziché copiare l'input nell'output; mentre come svantaggi si ha che l'utilizzo di un modello con parametrizzazione eccessiva a causa della mancanza di dati di addestramento sufficienti può creare un adattamento eccessivo.
- Un autoencoder la cui dimensione del code layer (h) è maggiore della dimensione dell'input layer (x) è chiamato **overcomplete**. Questo modello induce la scarsità dei parametri: si hanno il minor numero possibile di neuroni che si attivano in un dato momento quando viene presentato uno stimolo, si ottiene maggiore flessibilità e capacità di apprendimento multi-task (neuroni diversi possono concentrarsi su funzioni diverse) , e le reti neurali vengono generalmente codificate come un tensore (matrice).

Inoltre, si scoprono relazioni non lineari che difficilmente vengono individuate tramite metodi statistici.

- Un autoencoder **regularized** impedisce di apprendere mappe inutili come ad esempio la funzione identità e la funzione di perdita incoraggia il modello ad trovare altre proprietà oltre alla possibilità di copiare il suo input nel suo output. I vantaggi ottenuti sono: scarsità della rappresentazione, piccoli derivati della rappresentazione, robustezza al rumore e dati di ingresso mancanti.
- Un autoencoder **sparse** è un autoencoder il cui criterio di allenamento prevede una penalità di scarsità $\Omega(h)$ sul code layer (h), oltre all'errore di ricostruzione:

$$L(x, g(f(x))) + \Omega(h)$$

Gli autoencoder sparse vengono in genere utilizzati per apprendere funzionalità per un'altra attività, come la classificazione.

- Gli autoencoder **penalizing** costringono il modello ad apprendere una funzione che non cambia molto quando x cambia leggermente:

$$L\left(x, g\left(f(x)\right)\right) + \Omega(h, x)$$

Poiché questa penalità viene applicata solo agli esempi di addestramento, forza l'autoencoder ad apprendere le funzionalità che acquisiscono informazioni sulla distribuzione dell'addestramento.

$$\Omega(h, x) = \lambda \sum_i ||\nabla_x h_i||^2$$

- Invece di aggiungere una penalità, un autoencoder **denoising** ha una diversa funzione di perdita calcolata su un vettore di input danneggiato x':

$$L(x, g(f(x')))$$

L'autoencoder denoising (DAE) riceve un punto dati danneggiato come input ed è addestrato per prevedere il punto dati originale e non corrotto come output, quindi si ottiene maggiore robustezza.

- L'idea di fondo del **variational** autoencoder (VAE) è che input simili sono vicini tra loro nello spazio latente, quindi si addestra un AE e si usa la sua rappresentazione latente per identificare i cluster corrispondenti alle diverse classi. I VAE vengono solitamente utilizzati nei modelli generativi; creano (in base alla progettazione) uno spazio latente regolare e continuo, facilitando l'identificazione dei cluster (dopo la fase di addestramento). Proprio come un autoencoder standard, un autoencoder variazionale è un'architettura composta sia da un encoder che da un decoder e che è addestrato per

ridurre al minimo l'errore di ricostruzione tra i dati codificati-decodificati e i dati iniziali. Tuttavia, per introdurre una certa regolarizzazione dello spazio latente, si fa una leggera modifica del processo di codifica-decodifica: invece di codificare un input come un unico punto, lo si codifica come una distribuzione sullo spazio latente [39]. Il modello viene quindi addestrato come segue:

- l'input è codificato come distribuzione nello spazio latente
- un punto dello spazio latente viene campionato da quella distribuzione
- il punto campionato viene decodificato e può essere calcolato l'errore di ricostruzione
- l'errore di ricostruzione viene propagato indietro attraverso la rete

La rappresentazione latente conterrà un cluster per ogni classe di comportamento (il più separabile possibile). In seguito si dovrà applicare un algoritmo di clustering alla rappresentazione latente per identificare i cluster. Questo rappresenta un problema di apprendimento standard senza supervisione. Esistono diversi algoritmi che possono essere trovati in letteratura (e implementati nelle librerie python ML): K-means, DBScan, Gaussian Mixture…

La rappresentazione latente VAE è una distribuzione di probabilità: la codifica di un input non è un singolo punto ma una regione (un cerchio in uno spazio latente k-D), la cui posizione e dimensione sono determinate da *mu* e *sigma.* Per creare uno spazio latente continuo (e avere le codifiche vicine l'una all'altra ma comunque separabili), la distribuzione di probabilità appresa dal VAE viene addestrata per assomigliare a una distribuzione target, generalmente una distribuzione normale standard. Per addestrare la VAE è necessario avere una funzione di perdita più complessa per ridurre al minimo sia l'errore di ricostruzione che la distanza tra la distribuzione appresa e quella target:

$$VAE_{loss} = \alpha * recon + \beta * KL$$

$$recon = L\left(X, g\left(f(x)\right)\right) \quad errore\ di\ ricostruzione$$

$$KL = Kullback - Leibler, misura\ la\ differenza\ tra\ due\ distribuzioni\ di\ probabilità$$

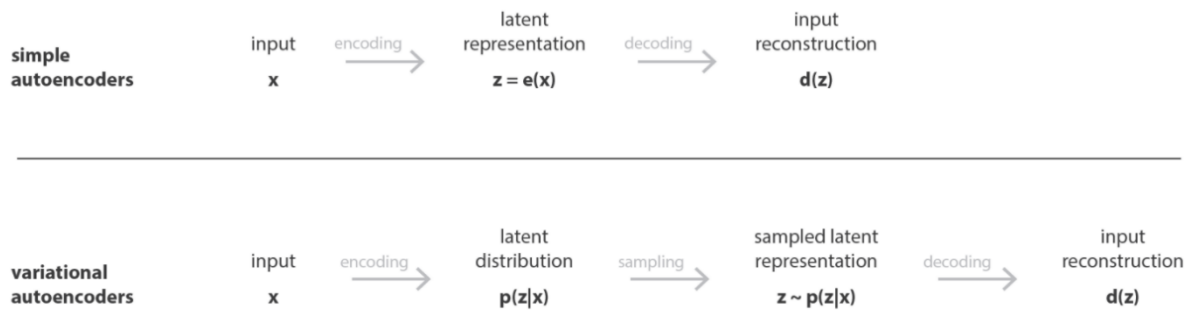

Figura 178 - differenza tra VAE e AE

4.3 Il caso di studio

Gli attuali metodi per il rilevamento delle anomalie nei supercomputer appartengono alla classe di Machine Learning supervisionato. Nella fase di addestramento devono essere forniti esempi sia normali che anomali per assicurare il successo del compito di apprendimento. Questi requisiti complicano notevolmente il compito di addestramento: nei supercomputer sono disponibili grandi quantità di dati ma le etichette sono scarse ed individuare le stesse anomalie risulta difficile. Un altro ramo ML che si può utilizzare per questo scopo, è l'apprendimento non supervisionato, poiché richiede solo un set di dati contenente le caratteristiche che descrivono lo stato del sistema (senza etichette) e durante il training l'algoritmo apprende proprietà utili sulla struttura del dataset. Le problematiche che si possono riscontrare con questo metodo sono la complessità e l'ottenere un rilevamento accurato delle anomalie [11]. Per questo motivo si è scelto di procedere percorrendo tre strade:

- Approccio semi supervisionato basato su Autoencoder
- Approccio non supervisionato basto su un Autoencoder variazionale (VAE)
- Approccio semi supervisionato basto su un Autoencoder variazionale (VAE)

Per allenare l'autoencoder varrà usata una porzione di dati (***train* set**) del data set che contiene solo situazioni normali, in modo tale che il modello possa apprendere il comportamento standard, mentre in seguito si andrà a testare con una porzione di dati (***test set***) che contiene anche delle anomalie. Ci sono 4 iperparametri che devono essere impostati prima di addestrare un autoencoder [40]:

- Dimensione layer latente: numero di nodi nel livello intermedio. Le dimensioni più piccole comportano una maggiore compressione.
- Numero di layer: l'autoencoder può essere profondo.

- Numero di nodi per layer: il numero di nodi per livello diminuisce con ogni livello successivo del codificatore e aumenta di nuovo nel decodificatore. Il decodificatore è simmetrico al codificatore in termini di struttura a strati.
- Funzione di perdita: errore quadratico, errore quadratico medio, errore assoluto, entropia incrociata, probabilità logaritmica negativa

4.3.1 Test set e Training set

Come primo passo si è deciso di usare un solo nodo per allenare l'autoencoder. In particolare il nodo scelto è il r249n06. Il campionamento avviene ogni 5 minuti, ma nel dataset sono riportati i valori ogni 15 minuti. Per poter lavorare con gli auteconder si sono dovuti definire due *dataFrame* che hanno come indice il timestamp:

- Train set: con solo elementi classificati come normali, nell'intervallo temporale 2020-08-18 e 2020-09-22 (2998 letture).
- Test set: con dati normali e qualche anomalia, nell'intervallo temporale 2020-09-25 12:00 e 2020-10-02 (602 letture di cui 29 sono punti anomali).

Prima di questa operazione è stato necessario normalizzare i dati fra 0 e 1 per velocizzare la fase di training, evitare eventuali blocchi in minimi locali, ridurre l'overfitting e per abilitare ogni layer ad apprendere in modo più autonomo. Per farlo si è deciso di usare un *MinMaxScaler*. La trasformazione è data da [41]:

$$\mathrm{X_{std}} = \frac{(\mathrm{X} - \mathrm{X.min(axis} = 0))}{(X.\max(axis = 0) - X.\min(axis = 0))}$$

$$X_{scaled} = X_{std} * (max - \min) + min$$

In cui min,max sono impostati in base al range desiderato, nel nostro caso 0 e 1. Inoltre, sono state escluse dall' algoritmo le features *label* (elemento discriminatorio per individuare stati anomali) e *timestamp* in quanto non dovranno essere utilizzate dai modelli di Machine Learning; quindi sia ha un passaggio da 234 a 232 features.

timestamp	avg:ambient	var:ambient	avg:dimm0_temp	var:dimm0_temp	avg:dimm10_temp	var:dimm10_temp	avg:dimm11_temp	var:dimm11_temp	avg:dimm
2020-08-18 00:00:00	0.804607	0.123515	0.825472	0.000000	0.695238	0.123653	0.746269	0.000000	
2020-08-18 00:15:00	0.822660	0.093349	0.825472	0.000000	0.704762	0.073068	0.746269	0.000000	
2020-08-18 00:30:00	0.804607	0.034442	0.825472	0.000000	0.676190	0.157377	0.746269	0.000000	
2020-08-18 00:45:00	0.873632	0.078385	0.863208	0.233333	0.714286	0.000000	0.766169	0.123653	
2020-08-18 01:00:00	0.833279	0.077910	0.825472	0.000000	0.714286	0.000000	0.746269	0.000000	

5 rows × 232 columns

Figura 179 - train set

timestamp	avg:ambient	var:ambient	avg:dimm0_temp	var:dimm0_temp	avg:dimm10_temp	var:dimm10_temp	avg:dimm11_temp	var:dimm11_temp	avg:dimm
2020-09-25 12:00:00	0.276685	0.022737	0.194631	0.000000	0.177914	0.079545	0.208333	0.000000	
2020-09-25 12:15:00	0.268258	0.032211	0.194631	0.000000	0.171779	0.000000	0.208333	0.000000	
2020-09-25 12:30:00	0.207865	0.058316	0.167785	0.244444	0.171779	0.000000	0.118056	0.147727	
2020-09-25 12:45:00	0.217697	0.012000	0.194631	0.000000	0.171779	0.000000	0.138889	0.284091	
2020-09-25 13:00:00	0.234551	0.035368	0.194631	0.000000	0.171779	0.000000	0.208333	0.000000	

5 rows × 232 columns

Figura 180 - test set

CAPITOLO 5

Approccio semi supervisionato basato su Autoencoder

In questo capitolo verrà mostrato l'approccio basato su un Autoencoder semplice con apprendimento profondo. In particolare si andrà a valutare l'efficacia di questa soluzione nel riconoscere situazione anomale tramite l'errore di ricostruzione.

5.1 Fase di training con Autoencoder semi supervisionato

La struttura dell'autoencoder si presenta come in Figura 181 (realizzata tramite il tool online [42]), dove ogni neurone rappresenta 5 unità.

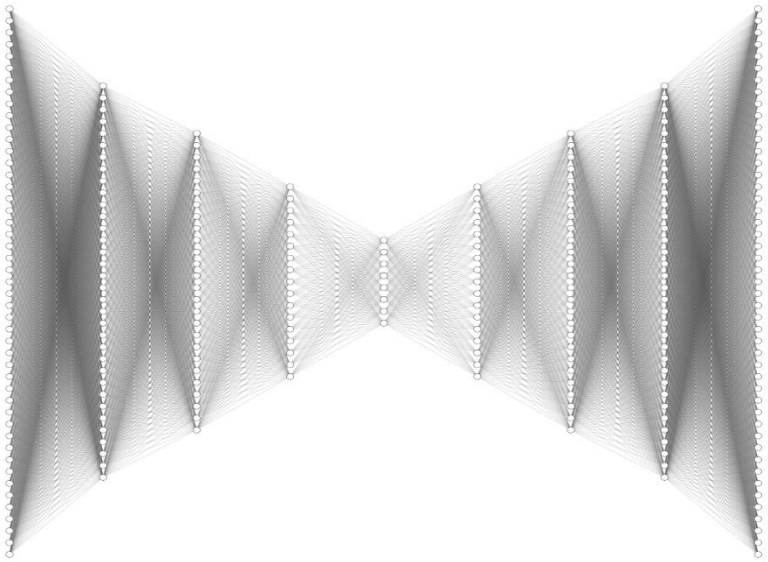

Figura 181 - struttura autoencoder

Quando i dati vengono inseriti in un codificatore automatico, vengono codificati e quindi compressi in una dimensione inferiore. La rete viene quindi addestrata sui dati codificati/compressi e produce una ricreazione di quei dati, poiché apprende l'essenza o le caratteristiche più importanti dei dati di input. Per creare l'autoencoder si è usato la libreria Keras di TensorFlow. Di seguito viene riportato il sommario dell'architettura dell'autoencoder.

```
Model: "autoencoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input (InputLayer)           [(None, 232)]             0
_________________________________________________________________
encoder_0 (Dense)            (None, 170)               39610
_________________________________________________________________
encoder_1 (Dense)            (None, 130)               22230
_________________________________________________________________
encoder_2 (Dense)            (None, 85)                11135
_________________________________________________________________
latent (Dense)               (None, 40)                3440
_________________________________________________________________
decoder_0 (Dense)            (None, 85)                3485
_________________________________________________________________
decoder_1 (Dense)            (None, 130)               11180
_________________________________________________________________
decoder_2 (Dense)            (None, 170)               22270
_________________________________________________________________
Output (Dense)               (None, 232)               39672
=================================================================
Total params: 153,022
Trainable params: 153,022
Non-trainable params: 0
```

Figura 182 - sommario autoencoder

L'autoencoder quindi sarà così costituito:

- Encoder: i primi 3 layers decrescenti (da encoder_0 a encoder_2)
- Un layer latente (latent)
- Decoder: gli ultimi 3 layers crescenti, simmetrici all'encoder (da decoder_0 a decoder_2)

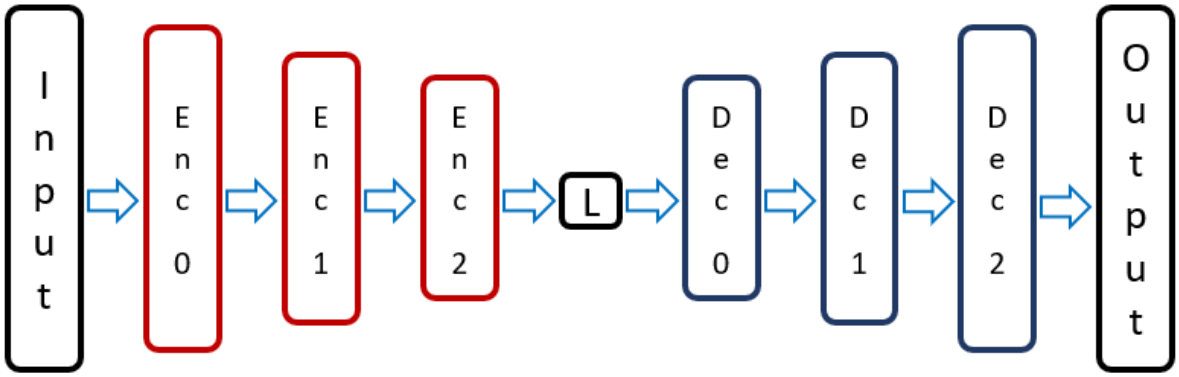

Figura 183 - dettaglio struttura autoencoder

Per il livello di input si prevedono un numero di risorse pari al numero delle feature del dataset (232) poi ma mano che si scende in fondo si riducono le risorse per ottenere un fattore di compressione come spiegato in precedenza. Come è possibile notare dal sommario dall'autoencoder ogni layer è di tipologia ***Dense***, ciò implica la scelta di usare una rete con architettura fully-connected in cui ogni neurone è collegato a tutti i neuroni dello strato precedente. Questa è una delle architetture più semplici ma che garantisce un rapido

apprendimento. Altro elemento importante è la funzione di attivazione [43], che serve per mappare l'ingresso all'uscita e aiuta una rete neurale ad apprendere relazioni e schemi complessi nei dati. Per questo caso di studio si è scelto di usare come funzione di attivazione ***ReLU*** poiché sta diventando una scelta predefinita in questi ambiti per i molti vantaggi che si hanno [44]. In particolare è utile perché è semplice da implementare, è computazionalmente economica, è capace di generare output con un vero valore zero (sparsità rappresentazionale), è molto diffusa, non ha problemi con la perdita di gradiente e conserva molte proprietà dei modelli lineari.

$$R(z) = \max(0, z) \quad dove\ z = Wx + b$$

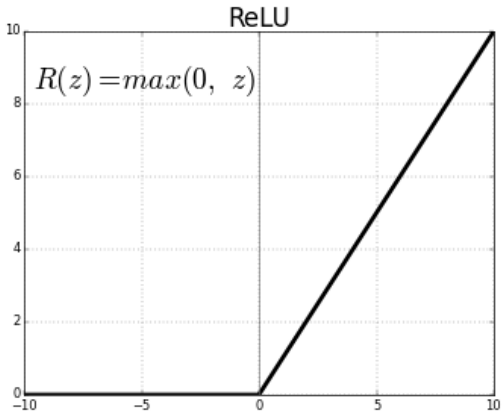

Figura 184 – ReLU

Altri elementi importanti per la creazione dell'autoencoder sono ***l'optimizer*** e la ***loss function***. Come optimizer si è scelto ***Adam***, un'estensione del stochastic gradient descent che ha recentemente visto una più ampia adozione per le applicazioni di apprendimento profondo nella visione artificiale e nell'elaborazione del linguaggio naturale. Adam viene descritto come la combinazione dei vantaggi di altri due algoritmi: AdaGrad and RMSProp. Invece di adattare i tassi di apprendimento dei parametri in base al primo momento medio (la media) come in RMSProp, Adam utilizza anche la media dei secondi momenti dei gradienti (la varianza non centrata). Nello specifico, l'algoritmo calcola una media mobile esponenziale del gradiente e del gradiente quadrato, e i parametri beta1 e beta2 controllano i tassi di decadimento di queste medie mobili. Il valore iniziale delle medie mobili e i valori beta1 e beta2 vicini a 1.0 determinano una distorsione delle stime dei momenti verso lo zero. Secondo Kingma [45] il metodo è *"computazionalmente efficiente, ha pochi requisiti di memoria, invariante al riscalamento diagonale dei gradienti ed è adatto per problemi che sono grandi in termini di dati/parametri"*.

Per quanto riguarda la loss function si sono seguite due strade usano le funzioni più comuni fornite da Kears: ***mean absolute error (MAE)*** e ***mean squared error (MSE)***. Lo scopo delle funzioni di perdita è calcolare la quantità che un modello dovrebbe cercare di ridurre al minimo durante l'addestramento [46].

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(y_t - x_t)^2$$

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_t - x_t|$$

Per addestrare il modello si è usato il metodo ***fit*** fornito da Keras. Attraverso questo metodo si sono settati dei parametri di input: il numero di epoche per l'allenamento del modello, la batch size e la dimensione di un validation set. Come numero di epoche si è scelto 100, come batch size 256. Osservando l'andamento dell'encoder con funzione di perdita MAE si può notare che il valore di partenza di perdita è pari a circa 0.14 ed arriva, andando avanti con le epoche a circa 0.02, mentre la perdita sul validation set parte da 0.13 e arriva a 0.07(Figura 185). Osservando l'andamento dell'encoder con funzione di perdita MSE si nota come la perdita parta da un valore di 0.07 e arrivi ad un valore di 0.005, mentre la perdita sul validation set parte da 0.05 e arriva a 0.02 (Figura 186). Da questo si può evincere che la fase di training è stata effettuata abbastanza bene sui dati e che non è necessario fare una modifica strutturale ai dati.

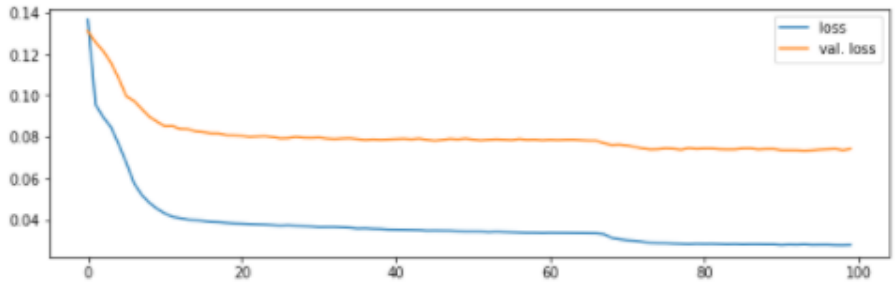

Figura 185 - Training con MAE

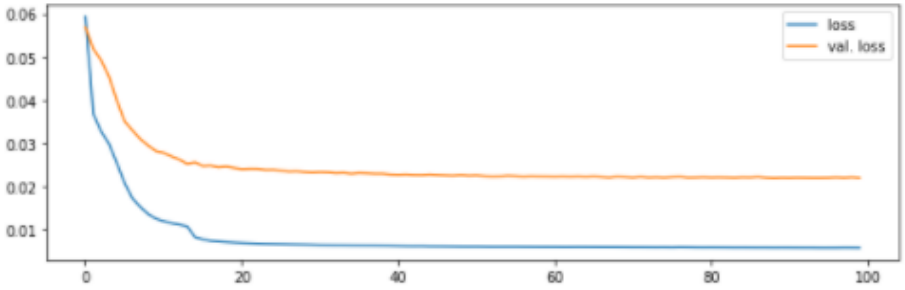

Figura 186 - Training con MSE

Si vuole verificare che l'autoencoder allenato con MSE, sia in grado di ricostruire le feature del testset, quindi si esegue questa prova sulle prime sei feature. Inoltre si vuole mettere in evidenza il comportamento dell'AE in durante gli stati anomali/normali, evidenziati nella figura seguente attraverso la feature label (punti in verde) che assume valore uno in corrispondenza di uno stato anomalo, zero altrimenti. Il risultato ottenuto è il seguente:

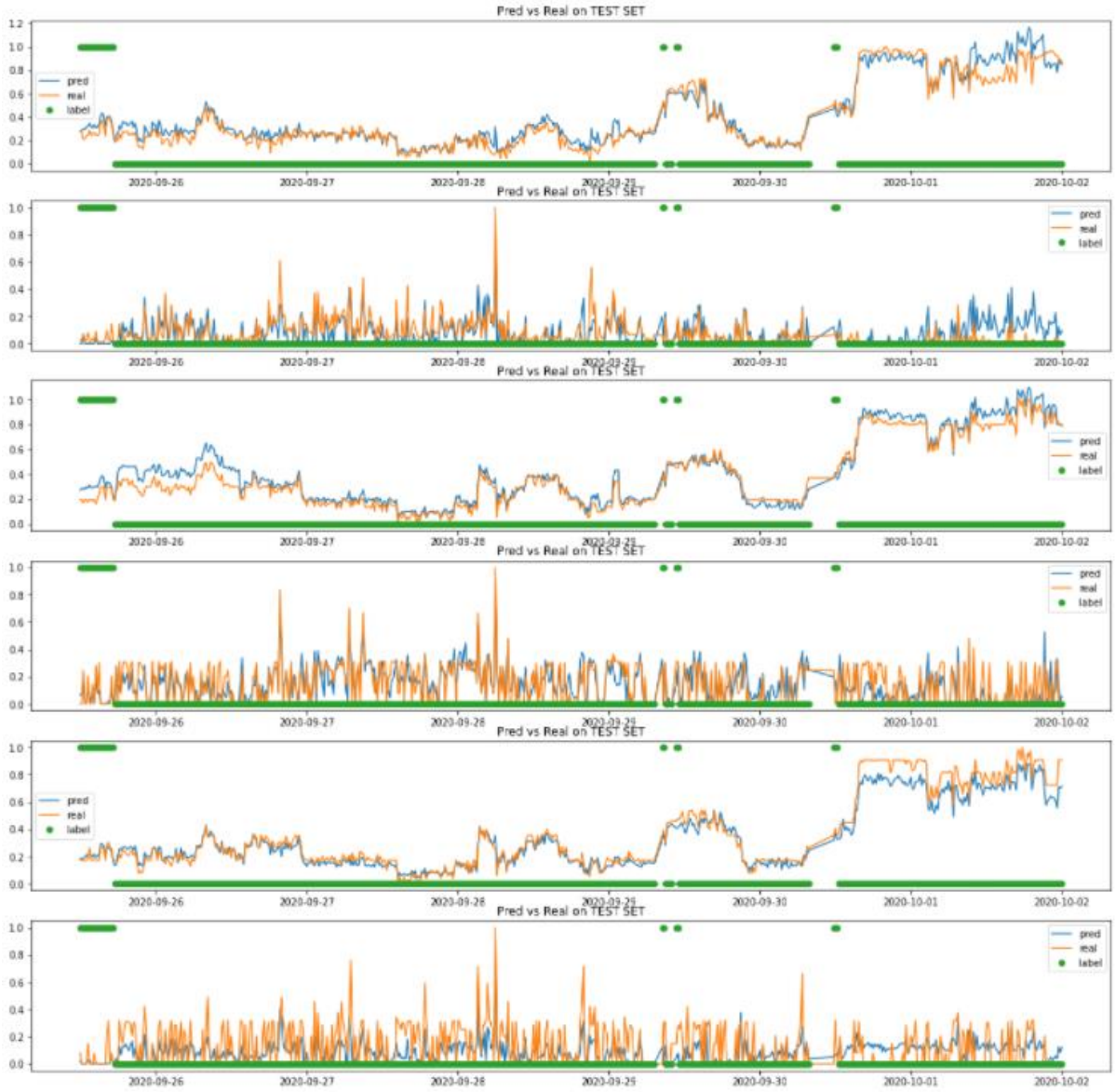

Figura 187 - Ricostruzione prime sei feature con autoencoder allenato con MSE

Come si può osservare nella figura 187, l'AE è in grado di ricostruire le prime sei feature del test set in modo abbastanza coerente.

Si è poi provato ad usare l'autoencoder su dati *nuovi* per osservare il suo comportamento tramite l'errore di ricostruzione per valutare la capacità di riproduzione dell'output. Il nodo preso in analisi è il **r205n20**. Nelle figure seguenti vengono usati due assi y per poter mettere a confronto l'errore di ricostruzione con lo stato (anomalo/normale) del nodo, in particolare l'asse di sinistra indica il valore dell'errore di ricostruzione, mentre quello di destra mostra la feature label che assume valore 0 per una situazione normale e 1 per una anomalia.

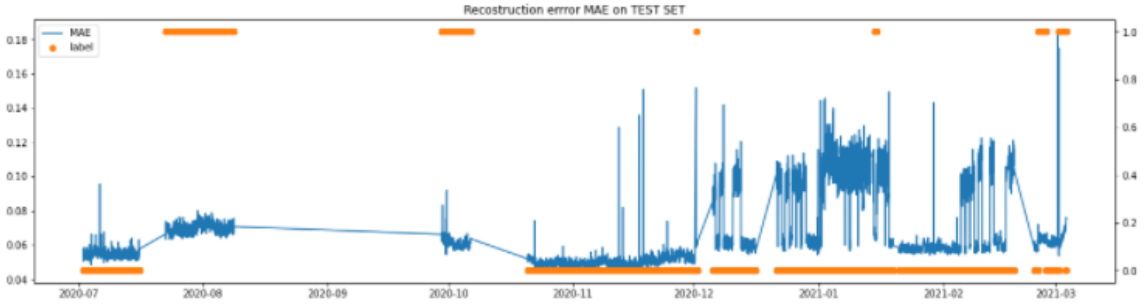

Figura 188 – Reconstruction error MAE

Figura 189 - Reconstruction error MSE

Quello che si può evincere dalle figure precedenti (188-189) è che ***prima*** che si presentino delle anomalie, ***l'errore di ricostruzione risulta essere più alto***. Pertanto si riportano alcuni zoom dei momenti precedenti e successivi di alcuni periodi anomali, per osservare meglio questo comportamento. Per alcuni intervalli non si hanno a disposizione i dati precedenti ad una anomalia (fig. 190-191), pertanto in questo caso verrà mostrato solo il periodo relativo all'anomalia stessa, utilizzando un doppio asse y, come per le figure precedenti.

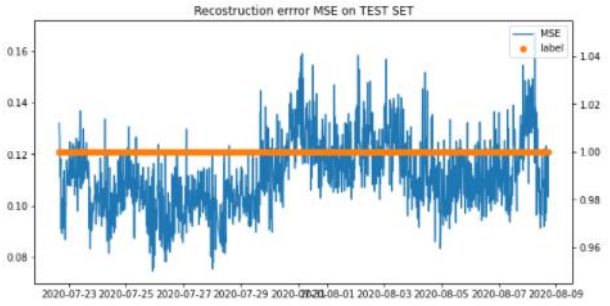

Figura 191 - Intervallo 23/07/20 – 09/08/2020

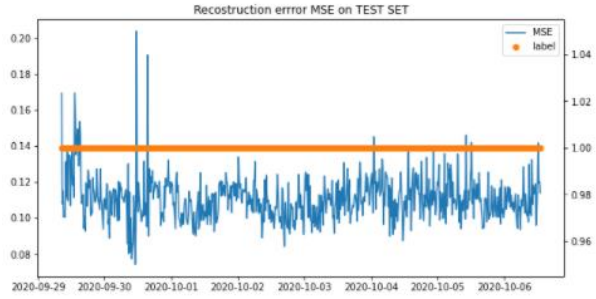

Figura 190 - Intervallo 29/09/20 – 06/10/2020

Figura 193 - Intervallo 12/01/21

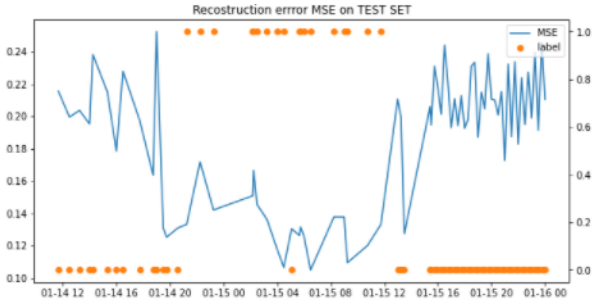

Figura 192 - Intervallo 14/01/21 – 16/01/2021

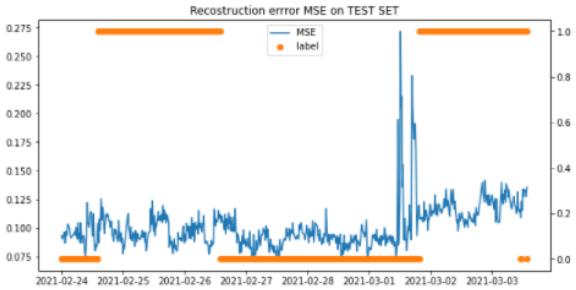

Figura 194 - Intervallo 23/07/20 – 09/08/2020

Nella figura 193 si può notare come effettivamente, prima dell'insorgere di una anomalia, ci sia un netto aumento del reconstruction error e la stessa situazione è evidenziata anche in figura 194 prima della seconda anomalia. Mentre nel primo periodo anomalo della figura 194, si rileva un comportamento non particolarmente ottimale in quanto il reconstruction error risulta essere molto simile sia durante il periodo pre-anomalia che durante l'anomalia. Nella figura 192 si rileva una situazione non chiara che può essere giustificata dal fatto che si hanno molti salti nei dati, sia nel periodo pre-anomalia che durante il periodo anomalo.

Per poter effettuare una migliore valutazione sul comportamento dell'autoencoder si è pensato di dividere il data set del nodo r205n20 che contiene i dati relativi al periodo dal 2020-07-02 08:30:00 al 2021-03-03 13:45:00, in *training_set (T_r)*, *test_set* (T_e^n) con soli punti normali, *test_set(T_e^a)* con solo punti anomali *e test_set_pre_anomalia* che contiene i dati di 2h e 30m precedenti alle anomalie e calcolare l'errore medio di ricostruzione su ognuno di essi. Di seguito viene mostrata la distribuzione dell'errore di distribuzioni nei vari casi presi analisi. Si evidenzia che nel caso del test set con anomalie i valori risultanti sono molto più alti, rispetto a agli altri due data set, confermando che ***durante i periodi anomali il valore dell'errore di ricostruzione risulta essere più alto***.

5.2 Distribuzione dell'errore di ricostruzione allenato con MSE

Figura 195 - Distribuzione dell'errore di ricostruzione su Training_set (T_r) MSE

Figura 196 - Distribuzione dell'errore di ricostruzion su Test_set_normale (T_e^n) MSE

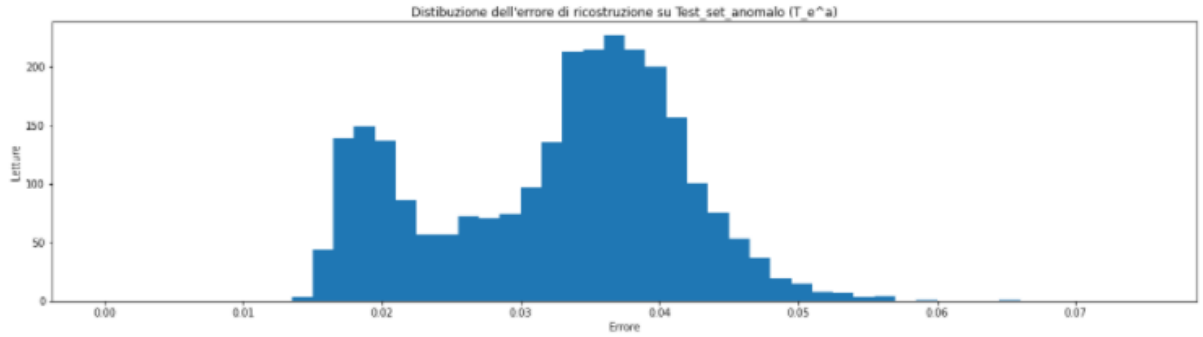

Figura 197 - Distribuzione dell'errore di ricostruzione su Test_set_anomalo (T_e^a) MSE

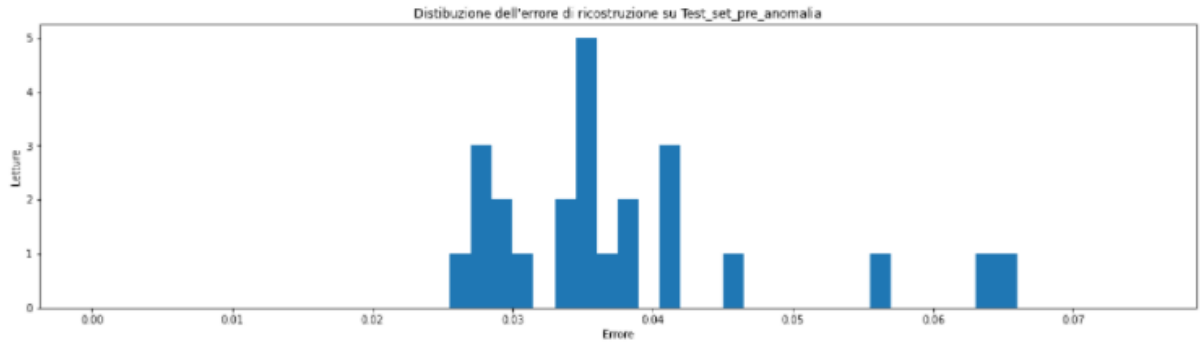

Figura 198 - Distribuzione dell'errore di ricostruzione su Test_set_pre_anomalia MSE

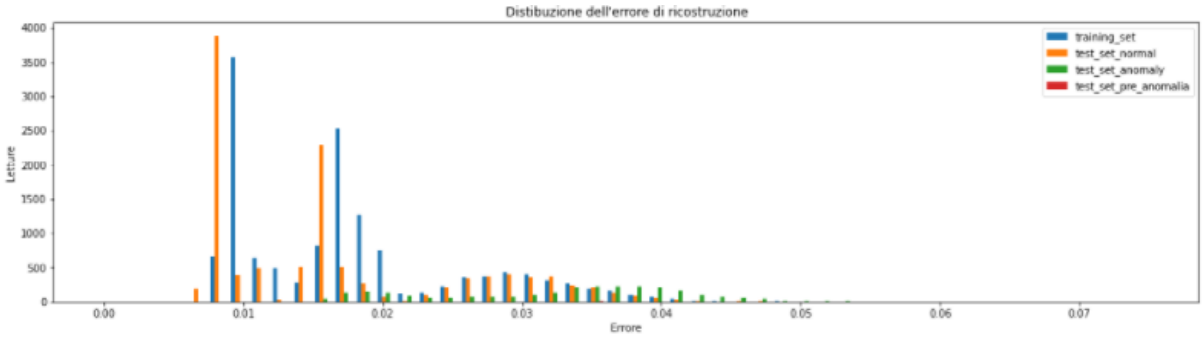

Figura 199 - Confronto errore di distribuzione nei diversi data set MSE

5.3 Distribuzione dell'errore di ricostruzione allenato con MAE

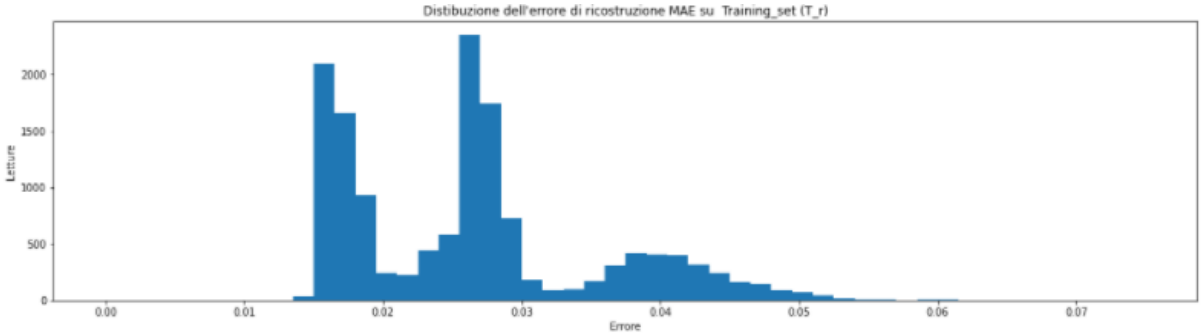

Figura 200 - Distribuzione dell'errore di ricostruzione su Training_set (T_r) MAE

Figura 201 - Distribuzione dell'errore di ricostruzione su Test_set_normale (T_e^n) MAE

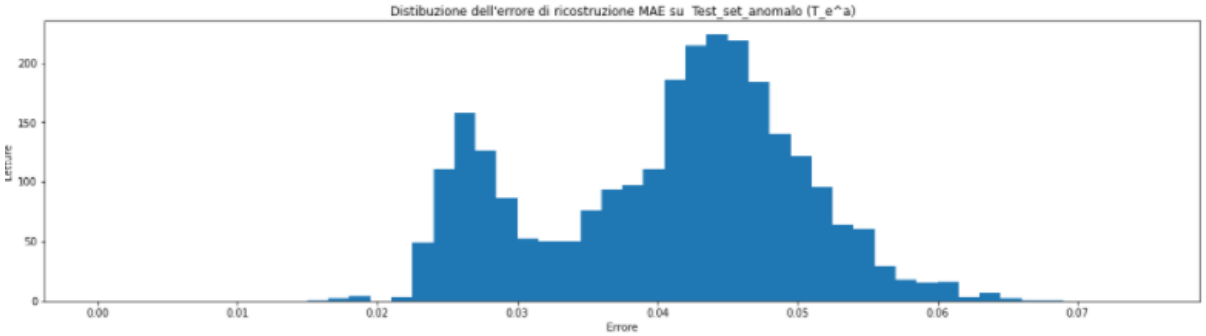

Figura 202 - Distribuzione dell'errore di ricostruzione su Test_set_anomalo (T_e^a) MAE

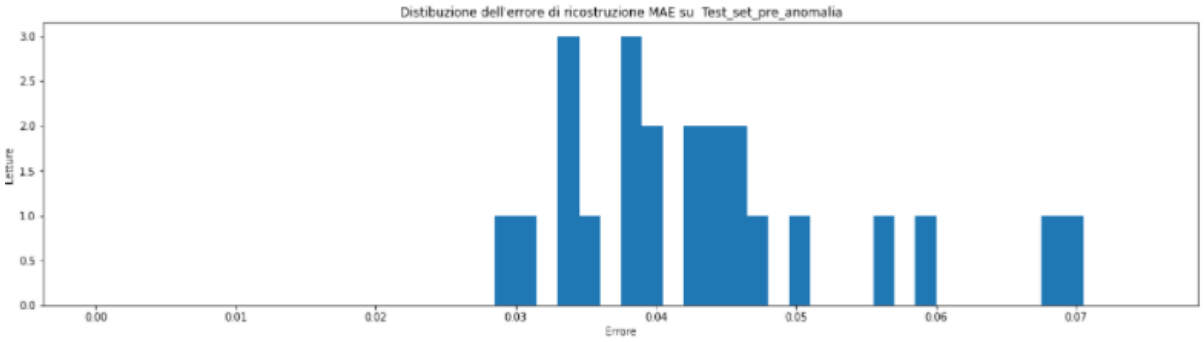

Figura 203 - Distribuzione dell'errore di ricostruzione su Test_set_pre_anomalia MAE

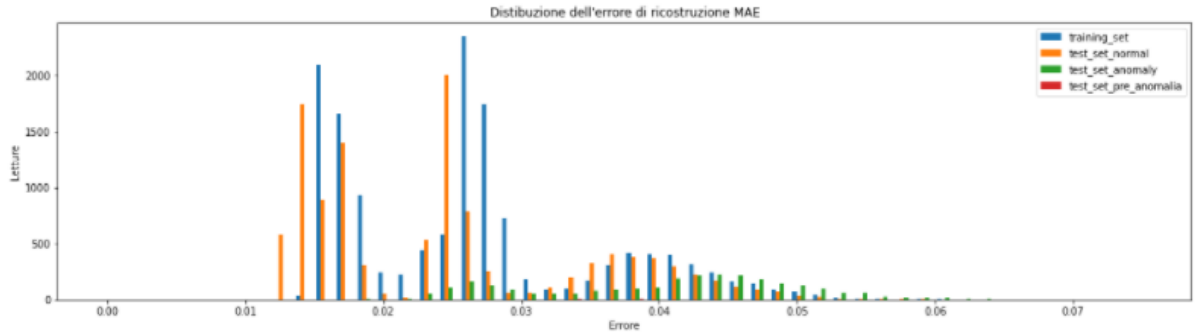

Figura 204 - Confronto errore di distribuzione nei diversi data set MAE

5.4 Valutazione dell'errore di ricostruzione su tutti i nodi

Si è pensato di valutare l'autoencoder tramite l'errore di ricostruzione anche su tutti gli altri dati dei nodi che si hanno a disposizione.

Come si può notare dalla tabella seguente, l'errore di ricostruzione sul dataset contenente solo valori relativi a periodi anomali, risulta essere molto più alto rispetto a quello degli altri due set di dati. In generale si può evincere che training_set e test_set_normale hanno sempre valori molto vicini tra loro e questo è giustificato anche dal fatto che le letture valutate come normali sono molto di più rispetto a quelle dei periodi anomali e il loro errore di ricostruzione varia tra 0.01 e 0.03, per il test_set_anomalo l'errore di ricostruzione varia tra 0.02 e 0.12, per il test_set_pre_anomalia l'errore di ricostruzione varia tra 0.02 e 0.15.

	Numero letture	Errore di ricostruzione medio MSE	Errore di ricostruzione medio MAE
Nodo: r205n20 Periodo: 2020-07-02 08:30:00 - 2021-03-03 13:45:00			
Training_set (T_r)	14230	0.01760	0.02639
Test_set_normale (T_e^n)	11556	0.01649	0.02466
Test_set_anomalo (T_e^a)	2674	0.03262	0.04081
Test_set_pre_anomalia	31	0.05760	0.07058
Nodo: r205n02 Periodo: 2020-07-22 15:45:00 - 2021-01-19 09:15:00			
Training_set (T_r)	8384	0.01419	0.02391
Test_set_normale (T_e^n)	6081	0.01939	0.02868
Test_set_anomalo (T_e^a)	2303	0.03694	0.04205
Test_set_pre_anomalia	No data	No data	No data
Nodo: r205n17 Periodo: 2020-05-31 22:15:00 - 2021-02-18 14:45:00			
Training_set (T_r)	5330	0.02435	0.03430
Test_set_normale (T_e^n)	5208	0.02427	0.03436
Test_set_anomalo (T_e^a)	122	0.07492	0.09042
Test_set_pre_anomalia	22	0.08091	0.09484
Nodo: r206n02 Periodo: 2020-07-06 12:30:00 - 2021-03-03 13:45:00			
Training_set (T_r)	10781	0.01681	0.02149
Test_set_normale (T_e^n)	9886	0.01723	0.02187
Test_set_anomalo (T_e^a)	895	0.04010	0.04593
Test_set_pre_anomalia	11	0.12336	0.14842
Nodo: r206n14 Periodo: 2020-05-31 22:15:00 - 2021-03-03 13:45:00			
Training_set (T_r)	7845	0.01657	0.02278
Test_set_normale (T_e^n)	7779	0.01530	0.02102
Test_set_anomalo (T_e^a)	66	0.05191	0.06504
Test_set_pre_anomalia	11	0.12049	0.14296
Nodo: r206n17 Periodo: 2020-05-31 22:15:00 - 2021-03-03 13:45:00			
Training_set (T_r)	11664	0.01738	0.02334
Test_set_normale (T_e^n)	11594	0.01466	0.02095
Test_set_anomalo (T_e^a)	70	0.03263	0.04297
Test_set_pre_anomalia	12	0.08800	0.10278
Nodo: r206n18 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	11681	0.01449	0.01983
Test_set_normale (T_e^n)	11617	0.01453	0.01985
Test_set_anomalo (T_e^a)	64	0.04853	0.06352
Test_set_pre_anomalia	11	0.10066	0.11465
Nodo: r206n19 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	11703	0.01557	0.02345
Test_set_normale (T_e^n)	11635	0.01325	0.02099
Test_set_anomalo (T_e^a)	68	0.04126	0.05388
Test_set_pre_anomalia	12	0.08171	0.09451
Nodo: r208n02 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	11382	0.01599	0.02118
Test_set_normale (T_e^n)	11365	0.01618	0.02137
Test_set_anomalo (T_e^a)	17	0.06434	0.08157
Test_set_pre_anomalia	11	0.10713	0.12989
Nodo: r208n04 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			

Training_set (T_r)	6114	0.01535	0.02014
Test_set_normale (T_e^n)	6107	0.01536	0.02016
Test_set_anomalo (T_e^a)	7	0.11072	0.12938
Test_set_pre_anomalia	11	0.10763	0.12396
Nodo: r208n07 Periodo: 2020-07-10 11:30:00 - 2021-03-03 13:45:00			
Training_set (T_r)	10405	0.01518	0.02206
Test_set_normale (T_e^n)	10393	0.01539	0.02224
Test_set_anomalo (T_e^a)	12	0.07301	0.08931
Test_set_pre_anomalia	8	0.13300	0.15494
Nodo: r208n12 Periodo: 2020-05-31 22:15:00 - 2021-03-03 13:45:00			
Training_set (T_r)	13546	0.01715	0.02331
Test_set_normale (T_e^n)	13520	0.01716	0.02332
Test_set_anomalo (T_e^a)	26	0.06033	0.07403
Test_set_pre_anomalia	50	0.02684	0.03850
Nodo: r210n05 Periodo: 2020-05-31 22:15:00 - 2021-03-03 13:45:00			
Training_set (T_r)	13667	0.01466	0.02317
Test_set_normale (T_e^n)	8302	0.01629	0.02407
Test_set_anomalo (T_e^a)	5365	0.02853	0.03310
Test_set_pre_anomalia	39	0.05688	0.06725
Nodo: r211n19 Periodo: 2020-05-31 22:15:00 - 2021-03-03 13:45:00			
Training_set (T_r)	6670	0.02158	0.02605
Test_set_normale (T_e^n)	5588	0.01988	0.02564
Test_set_anomalo (T_e^a)	1082	0.03594	0.04821
Test_set_pre_anomalia	42	0.05898	0.08437
Nodo: r212n06 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	10244	0.01357	0.01886
Test_set_normale (T_e^n)	10195	0.01357	0.01886
Test_set_anomalo (T_e^a)	49	0.04512	0.05470
Test_set_pre_anomalia	16	0.08877	0.11105
Nodo: r215n05 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	17576	0.01042	0.01599
Test_set_normale (T_e^n)	12388	0.00946	0.01480
Test_set_anomalo (T_e^a)	5188	0.02383	0.03203
Test_set_pre_anomalia	41	0.04052	0.05735
Nodo: r218n03 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	8300	0.02002	0.02744
Test_set_normale (T_e^n)	7332	0.01841	0.02673
Test_set_anomalo (T_e^a)	968	0.05312	0.06762
Test_set_pre_anomalia	28	0.05663	0.07624
Nodo: r224n20 Periodo: 2020-05-31 22:15:00 - 2021-03-03 13:45:00			
Training_set (T_r)	10792	0.01854	0.02404
Test_set_normale (T_e^n)	10634	0.01875	0.02423
Test_set_anomalo (T_e^a)	158	0.03483	0.04266
	21	0.09530	0.10741
Nodo: r239n12 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	13495	0.01463	0.02270
Test_set_normale (T_e^n)	9102	0.01652	0.02447
Test_set_anomalo (T_e^a)	4393	0.03234	0.04487

Test_set_pre_anomalia	42	0.04347	0.05995
Nodo: r243n11 Periodo: 2020-05-31 22:15:00 - 2021-03-03 13:45:00			
Training_set (T_r)	12182	0.01344	0.01989
Test_set_normale (T_e^n)	11717	0.01390	0.02044
Test_set_anomalo (T_e^a)	465	0.02527	0.03292
Test_set_pre_anomalia	48	0.05147	0.06438
Nodo: r249n06 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	14932	0.01992	0.02521
Test_set_normale (T_e^n)	14477	0.01233	0.01785
Test_set_anomalo (T_e^a)	455	0.02798	0.03680
Test_set_pre_anomalia	34	0.06334	0.08084
Nodo: r253n06 Periodo: 2020-06-16 13:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	9862	0.01447	0.01993
Test_set_normale (T_e^n)	9684	0.01449	0.01991
Test_set_anomalo (T_e^a)	178	0.04243	0.04847
Test_set_pre_anomalia	No data	No data	No data
Nodo: r254n01 Periodo: 2020-05-31 22:30:00 - 2021-03-03 13:45:00			
Training_set (T_r)	17191	0.01623	0.02490
Test_set_normale (T_e^n)	15873	0.01612	0.02247
Test_set_anomalo (T_e^a)	1318	0.02309	0.03331
Test_set_pre_anomalia	89	0.03398	0.04918
Nodo: r256n05 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	11140	0.01071	0.01675
Test_set_normale (T_e^n)	11024	0.01082	0.01695
Test_set_anomalo (T_e^a)	116	0.04383	0.05232
Test_set_pre_anomalia	45	0.03259	0.04438

	Numero letture	**Errore di ricostruzione medio MSE**	**Errore di ricostruzione medio MAE**
Media calcolata su tutti i nodi			
Training_set (T_r)	269116	0.01617	0.02280
Test_set_normale (T_e^n)	243057	0.01582	0.02233
Test_set_anomalo (T_e^a)	26059	0.04515	0.05608
Test_set_pre_anomalia	635	0.07497	0.09175

Il risultato ottenuto e che ha particolare valore per lo scopo di questa tesi è che in tutti i nodi, l'errore di ricostruzione durante e prima di una anomalia risulta essere molto più alto rispetto a quello di un periodo normale. La feature label, che indentifica una anomalia quando assume valore 1, viene settata da un sys admin solo dopo che un problema viene riscontrato sul nodo e quindi è molto probabile che un nodo sia in stato anomalo senza che la sua label sia 1. Il vantaggio che si avrebbe utilizzando l'autoencoder è rilevare questa situazione e questo viene confermato dai valori riportati nella tabella precedente, infatti l'AE si accorge che sul nodo sta succedendo qualcosa di strano prima che la label venga impostata su "Anomalia" avendo un

errore di ricostruzione sul valori pre_anomalia molto alto. Per poter analizzare meglio l'errore di ricostruzione sulle singole feature, si è pensato di utilizzare un heatmap, col quale possiamo mettere in evidenza quali feature hanno avuto maggiori difficoltà ad essere ricostruite. Questi grafici possono essere strumenti molto utili per una root cause analysis in quanto ci permettono di individuare agevolmente eventuali parametri insoliti. Per una visualizzazione migliore si è scelto di mostrare solo alcune delle 232 feature. Osservando la figura seguente si deduce che le feature con errore di ricostruzione maggiore durante un periodo di anomalia (evidenziato dalla linea arancione in alto) sono: ***avg:dimm15_temp, avg:dimm3_temp, avg:p0_core15_temp, avg:p0_core21_temp, avg:p0_core22_temp, avg:p0_core9_temp, avg:p0_core23_temp, avg:p0_core2_temp, avg:p0_core3_temp, avg:p0_core8_temp, avg:ps0_output_volta, avg:load_fifteen, avg:load_five, avg:load_one, avg:part_max_used, avg:proc_total, avg:swap_free, avg:state***. Si può anche notare che spesso prima dell'insorgere di una anomalia l'errore di ricostruzione aumenta, confermando cioè che è stato riportato in precedenza.

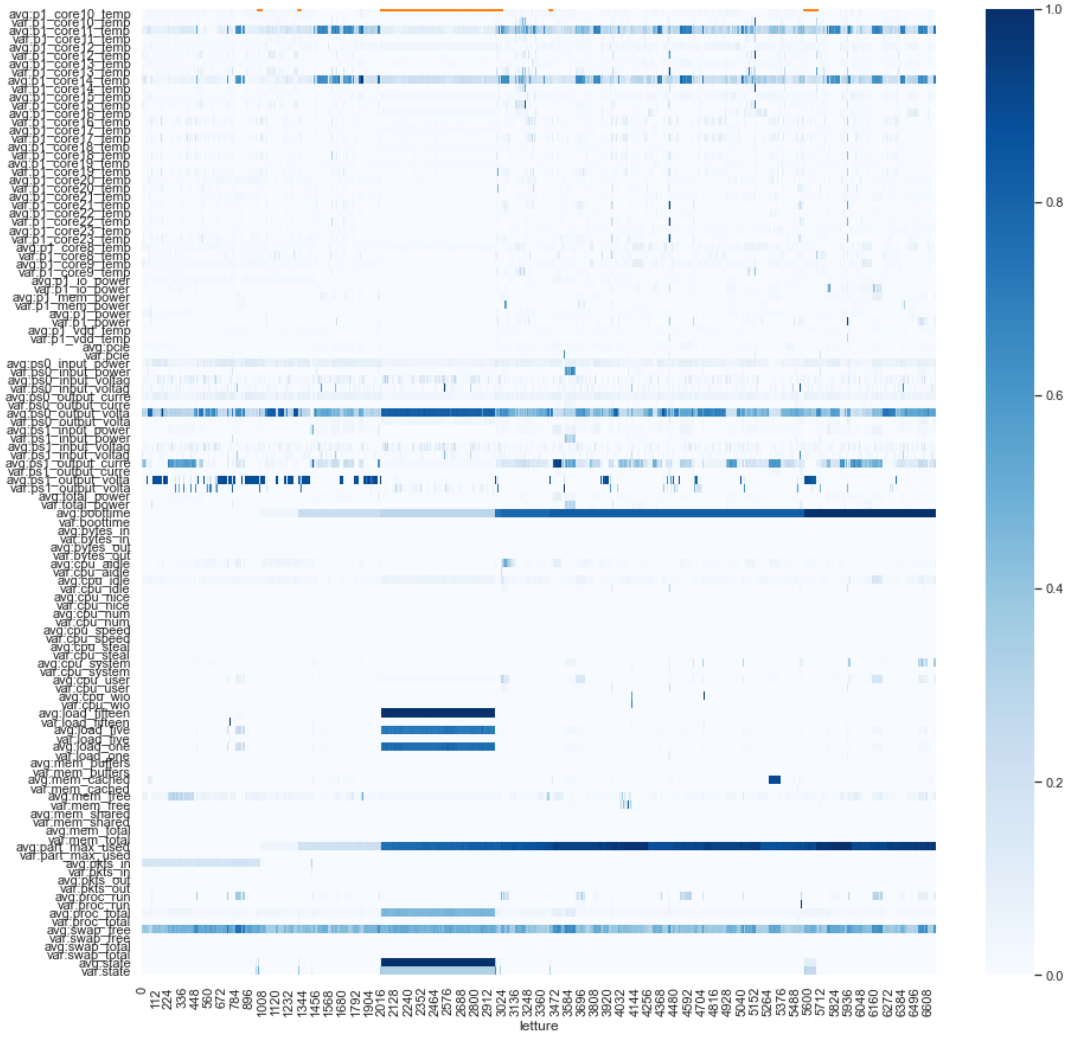

Figura 205 – Heatmap nodo n211n19 AE con MAE

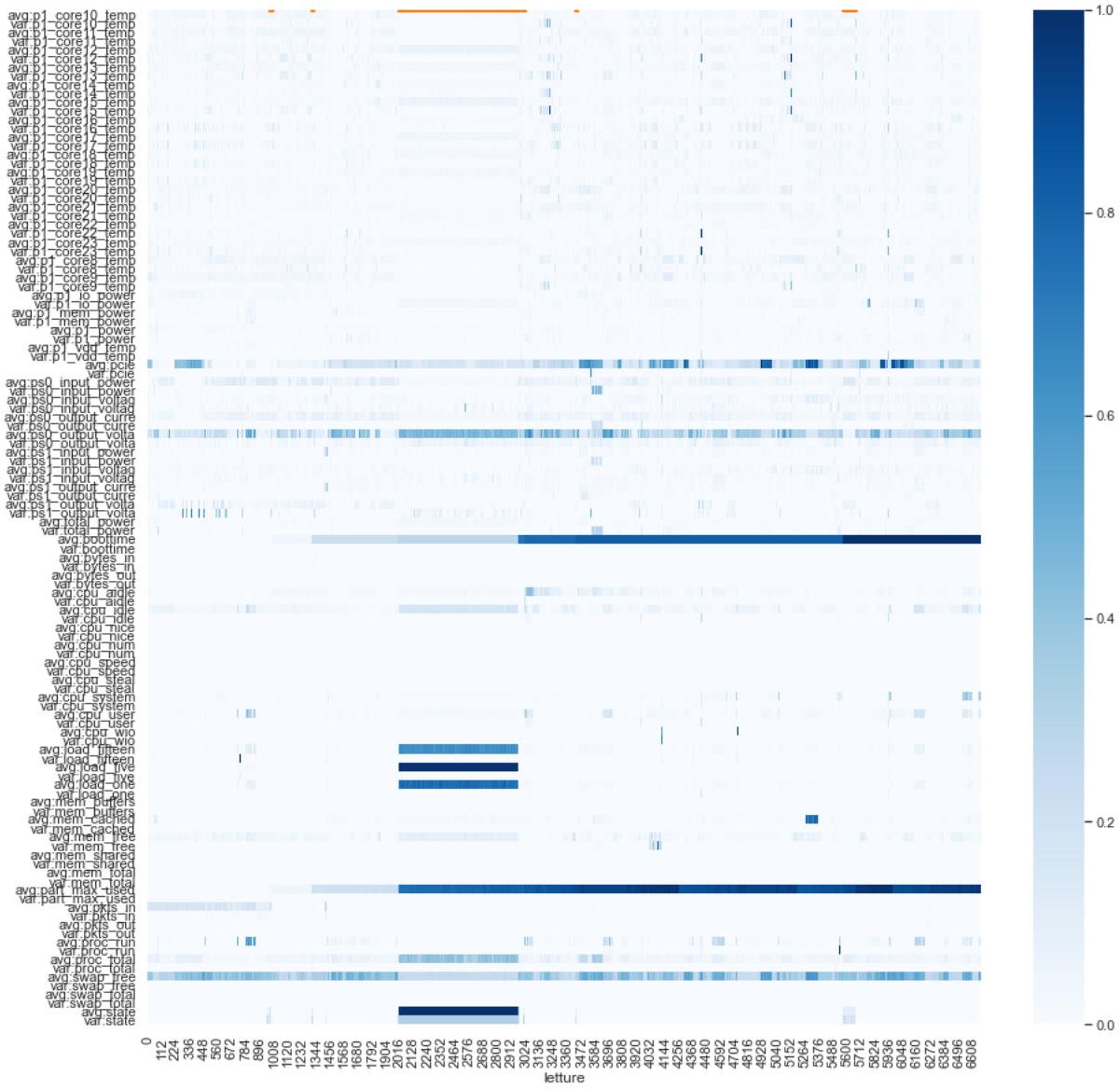

Figura 206 - Heatmap nodo n211n19 AE con MSE

CAPITOLO 6

Approccio basato su Autoencoder variazionale (VAE)

I risultati ottenuti con un AE semplice sono molto promettenti ma dato che si è in una fase esplorativa del problema, si è pensato di espandere l'analisi considerando anche una rete neurale più complessa per capire se si possono ottenere risultati migliori o si possono notare altre caratteristiche che possano essere utili allo scopo di Anomaly Detection. In particolare verrà utilizzato prima un VAE unsupervised, e in seguito un VAE con approccio semi-supervised, cioè con la stessa strategia di apprendimento che è stata usata in precedenza per allenare l'AE semplice, così da confrontare meglio i risultati ottenuti.

6.1 Autoencoder variazionale non supervisionato

L'architettura dell'autoencoder VAE che si è pensato di realizzare è così strutturata:

```
Model: "encoder"
__________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================================
input_1 (InputLayer)         [(None, 232)]             0
__________________________________________________________________________________________________
encoder_0 (Dense)            (None, 150)               34950       input_1[0][0]
__________________________________________________________________________________________________
encoder_1 (Dense)            (None, 100)               15100       encoder_0[0][0]
__________________________________________________________________________________________________
encoder_2 (Dense)            (None, 50)                5050        encoder_1[0][0]
__________________________________________________________________________________________________
z_mean (Dense)               (None, 2)                 102         encoder_2[0][0]
__________________________________________________________________________________________________
z_log_var (Dense)            (None, 2)                 102         encoder_2[0][0]
__________________________________________________________________________________________________
sampling (Sampling)          (None, 2)                 0           z_mean[0][0]
                                                                   z_log_var[0][0]
==================================================================================================
Total params: 55,304
Trainable params: 55,304
Non-trainable params: 0
```

Figura 207 - Encoder VAE

```
_________________________________________________________________
Model: "decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 2)]               0
_________________________________________________________________
decoder_0 (Dense)            (None, 50)                150
_________________________________________________________________
decoder_1 (Dense)            (None, 100)               5100
_________________________________________________________________
decoder_2 (Dense)            (None, 150)               15150
_________________________________________________________________
dense (Dense)                (None, 232)               35032
=================================================================
Total params: 55,432
Trainable params: 55,432
Non-trainable params: 0
```

Figura 208 - Decoder VAE

```
_________________________________________________________________
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 232)]             0
_________________________________________________________________
encoder (Functional)         [(None, 2), (None, 2), (N 55304
_________________________________________________________________
decoder (Functional)         (None, 232)               55432
=================================================================
Total params: 110,736
Trainable params: 110,736
Non-trainable params: 0
```

Figura 209 – VAE

La struttura dell'autoencoder VAE si presenta come in Figura 210.

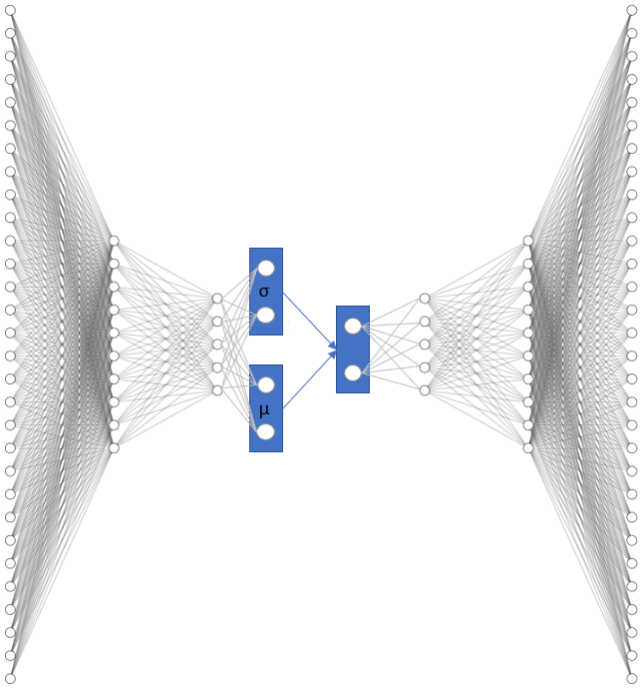

Figura 210 - struttura VAE

Come primo passo si è deciso di allenare l'autoecoder in modo non supervisionato, e osservando l'andamento dell'encoder si può notare che il valore perdita è pari a circa 0.07 ed arriva, andando avanti con le epoche a circa 0.007, mentre la perdita sul validation set parte da 0.04 e arriva a 0.02 (Figura 211).

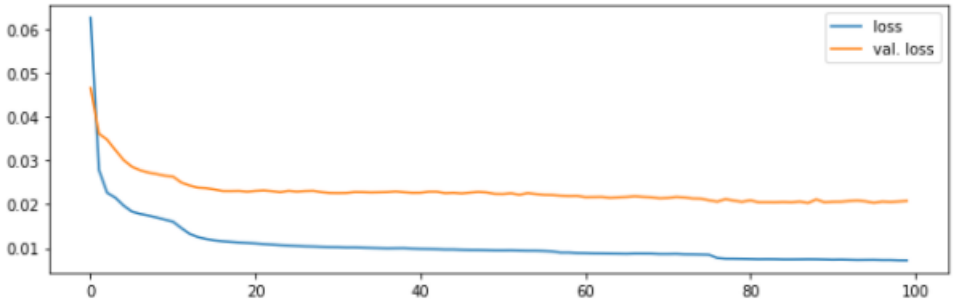

Figura 211 - Training con VAE

Si è poi provato ad usare l'autoencoder su dati "nuovi" per osservare il suo comportamento tramite il raggruppamento in cluster in base alla variabile "label". Il nodo preso in analisi è il **r239n12**.

Come si può osservare dalle figure seguenti, che rappresentano come si comportano le classi nello spazio latente al variare delle epoche, si può notare che le due classi individuate (anomalie in giallo, normali in azzurro) risultano essere raggruppate e distribuite nella stessa zona.

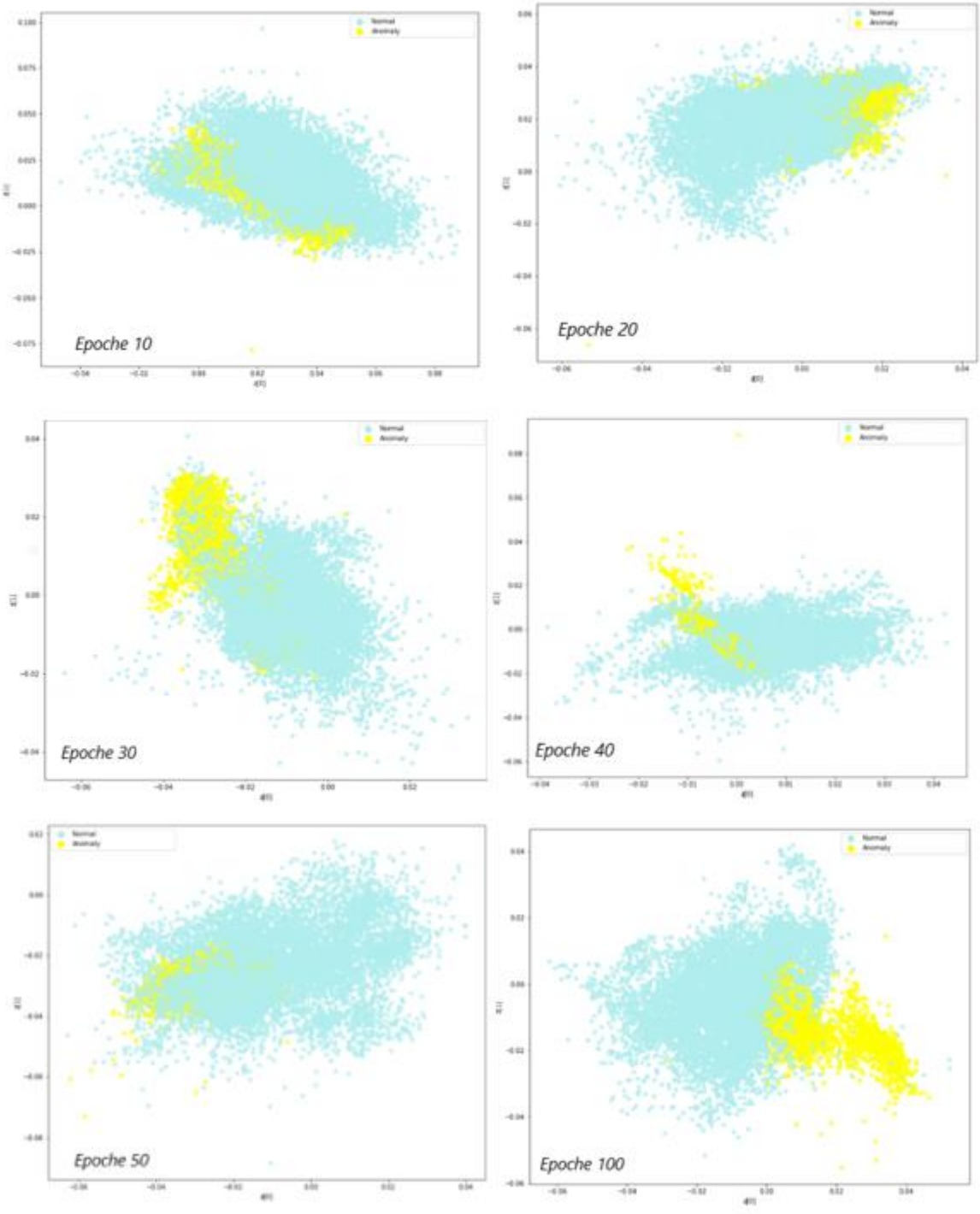

Figura 212 - Proiezione su spazio latente in varie epoche

6.1.1 Distribuzione dell'errore di ricostruzione

Per poter effettuare una migliore valutazione sul comportamento dell'autoencoder si è pensato di dividere il data set del nodo **r205n20** che contiene i dati relativi al periodo dal 2020-07-02 08:30:00 al 2021-03-03 13:45:00, in *training_set(T_r), test_set con soli punti normali (T_e^n), test_set con solo punti anomali (T_e^a) , test_set_pre_anomalia* che contiene i dati di 2h e 30m precedenti alle anomalie e calcolare l'errore medio di ricostruzione su ognuno di essi. Di seguito viene mostrata la distribuzione dell'errore di distribuzioni nei vari casi presi analisi. Si evidenzia che nel caso del test set con anomalie i valori risultanti sono molto più alti, rispetto a agli altri due data set, confermando che durante i periodi anomali il valore dell'errore di ricostruzione risulta essere più alto.

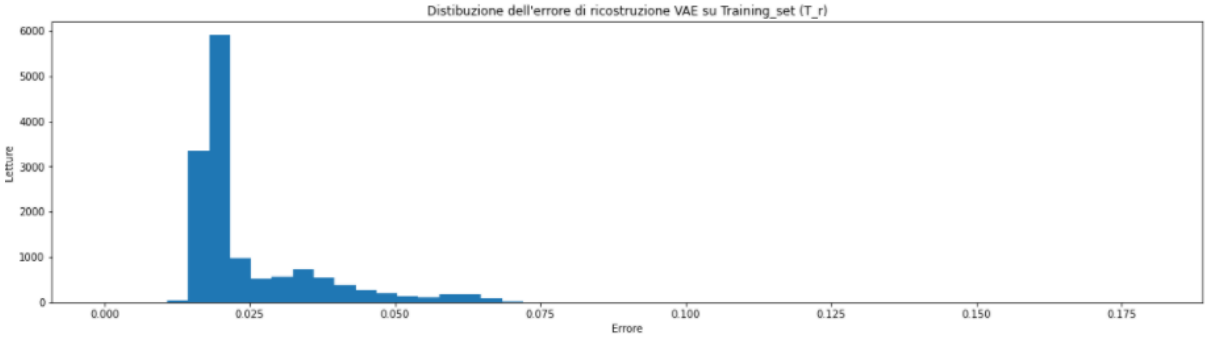

Figura 213 - Distribuzione dell'errore di ricostruzione su Training_set (T_r) VAE non sup.

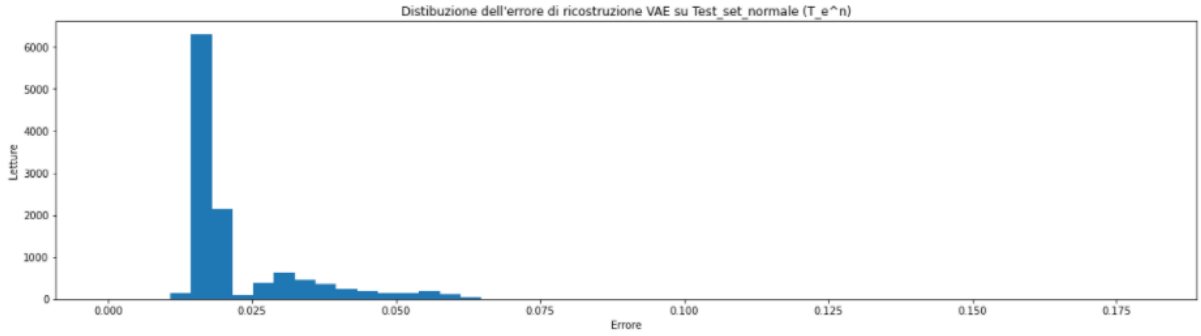

Figura 214 - Distribuzione dell'errore di ricostruzione su Test_set_normale (T_e^n) VAE non sup.

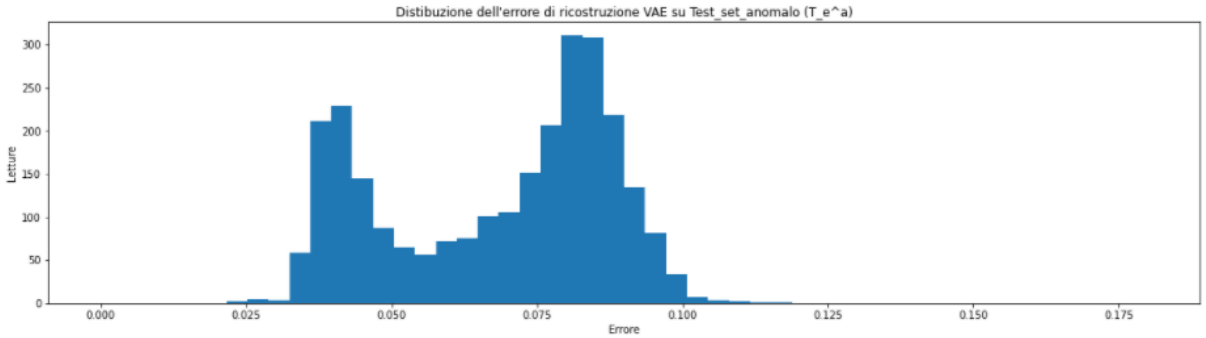

Figura 215 - Distribuzione dell'errore di ricostruzione su Test_set_anomalo (T_e^a) VAE non sup.

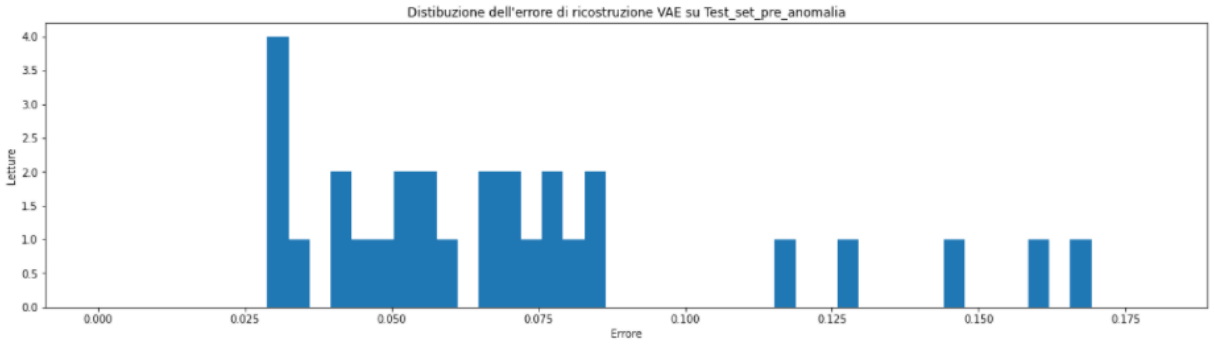

Figura 216 - Distribuzione dell'errore di ricostruzione su Test_set_pre_anomalia VAE non sup.

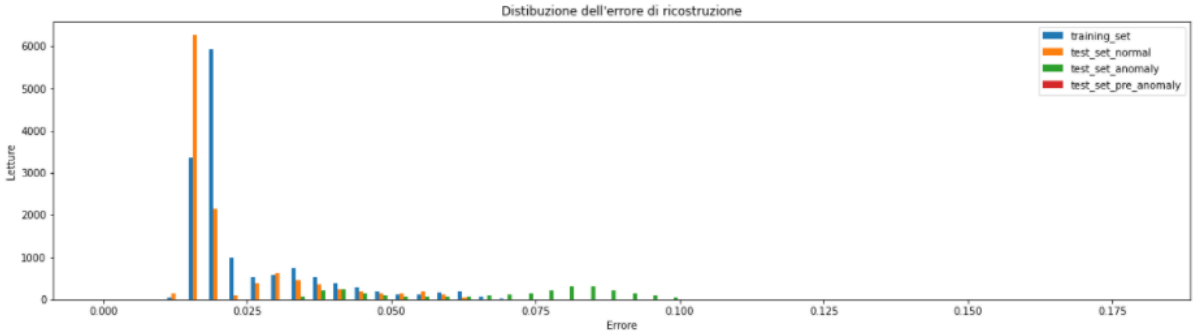

Figura 217 - Distribuzione errore di ricostruzione VAE non sup.

6.1.2 VAE con spazio latente a tre dimensioni

Si è pensato di provare a plottare i punti in uno spazio latente a 3 dimensioni per poter valutare meglio l'efficacia di questa soluzione. Per fare questo si è dovuto creare un nuovo autoencoder, di cui viene riportata la struttura:

```
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_9 (InputLayer)         [(None, 232)]             0
_________________________________________________________________________________________________
encoder_0 (Dense)            (None, 150)               34950       input_9[0][0]
_________________________________________________________________________________________________
encoder_1 (Dense)            (None, 100)               15100       encoder_0[0][0]
_________________________________________________________________________________________________
encoder_2 (Dense)            (None, 50)                5050        encoder_1[0][0]
_________________________________________________________________________________________________
z_mean (Dense)               (None, 3)                 153         encoder_2[0][0]
_________________________________________________________________________________________________
z_log_var (Dense)            (None, 3)                 153         encoder_2[0][0]
_________________________________________________________________________________________________
z_log_t (Dense)              (None, 3)                 153         encoder_2[0][0]
_________________________________________________________________________________________________
sampling_4 (Sampling)        (None, 3)                 0           z_mean[0][0]
                                                                   z_log_var[0][0]
                                                                   z_log_t[0][0]
=================================================================================================
Total params: 55,559
Trainable params: 55,559
Non-trainable params: 0
_________________________________________________________________________________________________
Model: "decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_10 (InputLayer)        [(None, 3)]               0
_________________________________________________________________
decoder_0 (Dense)            (None, 50)                200
_________________________________________________________________
decoder_1 (Dense)            (None, 100)               5100
_________________________________________________________________
decoder_2 (Dense)            (None, 150)               15150
_________________________________________________________________
dense_4 (Dense)              (None, 232)               35032
=================================================================
Total params: 55,482
Trainable params: 55,482
Non-trainable params: 0
```

Figura 218 - VAE con spazio latente a 3 dimensioni

Osservando l'andamento dell'encoder durante l'allenamento in 100 epoche, si può notare che il valore di partenza della perdita è pari a circa 0.07 ed arriva, andando avanti con le epoche a circa 0.005, mentre la perdita sul validation set parte da 0.04 e arriva a 0.01 (figura 219).

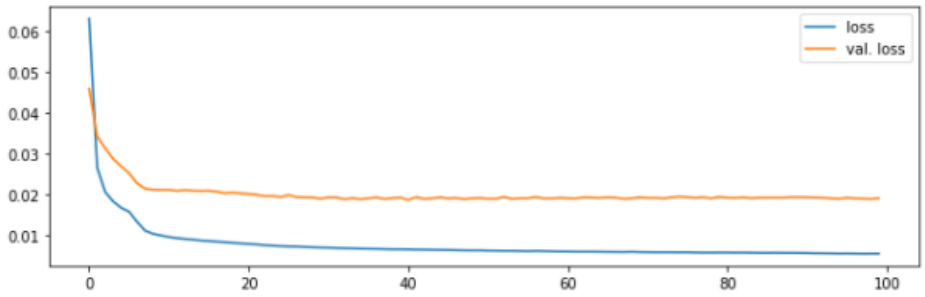

Figura 219 - Training con VAE con spazio latente a 3 dimensioni

Si è poi provato ad usare l'autoencoder su dati "nuovi" per osservare il suo comportamento tramite il raggruppamento in cluster in base alla variabile "label". Il nodo preso in analisi è il **r239n12**.

Come si può osservare dalla figura seguente, si può notare che le due classi individuate (anomalie in giallo, normali in azzurro) risultano essere raggruppate e distribuite nella stessa zona.

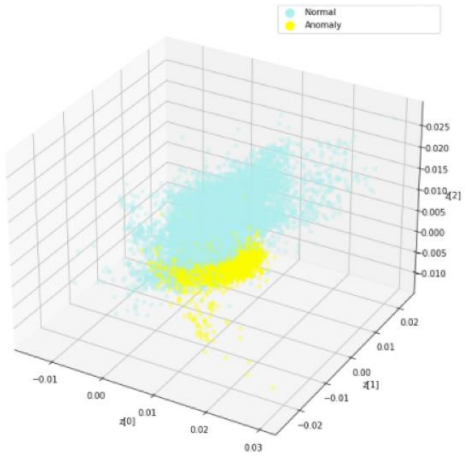

Figura 220 – Proiezione su spazio latente di 3 dimensioni

6.1.2.1 Distribuzione dell'errore di ricostruzione su VAE con spazio latente a tre dimensioni

Per poter effettuare una migliore valutazione sul comportamento dell'autoencoder si è pensato di dividere il data set del nodo r205n20 che contiene i dati relativi al periodo dal 2020-07-02 08:30:00 al 2021-03-03 13:45:00, in *training_set(T_r), test_set con soli punti normali (T_e^n), test_set con solo punti anomali (T_e^a) e test_set_pre_anomalia* che contiene i dati di 2h e 30m precedenti alle anomalie e calcolare l'errore medio di ricostruzione su ognuno di essi. Di

seguito viene mostrata la distribuzione dell'errore di distribuzioni nei vari casi presi analisi. Si evidenzia che nel caso del test set con anomalie i valori risultanti sono molto più alti, rispetto a agli altri due data set, confermando che durante i periodi anomali il valore dell'errore di ricostruzione risulta essere più alto.

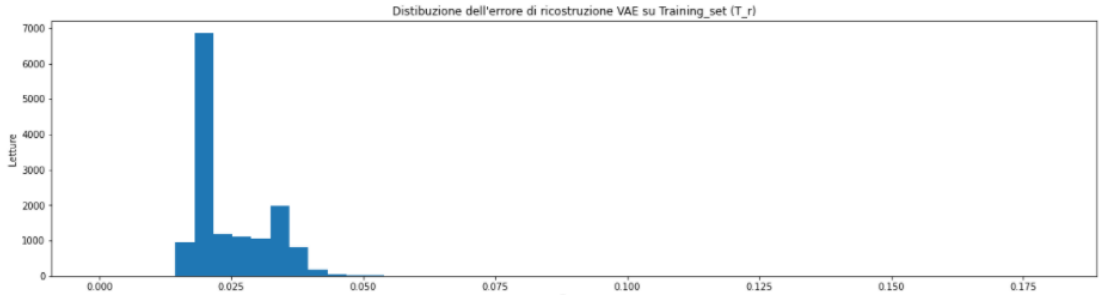

Figura 221 - Distribuzione dell'errore di ricostruzione su Training_set (T_r) VAE non sup. a spazio latente a 3 dim

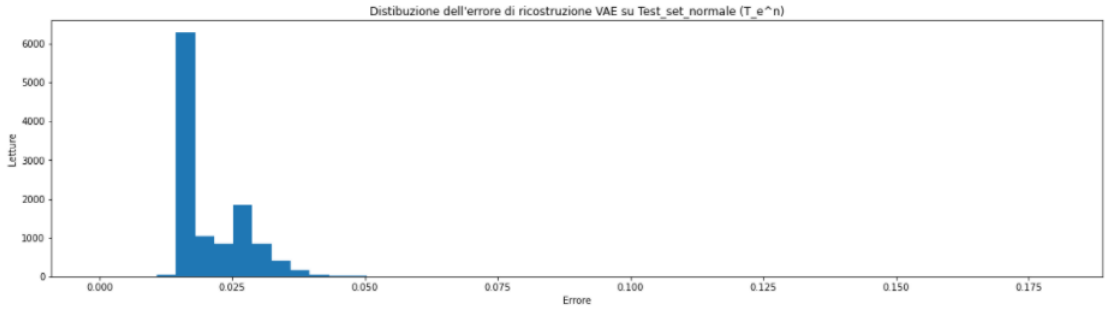

Figura 222 - Distribuzione dell'errore di ricostruzione su Test_set_normale (T_e^n) VAE non sup. a spazio latente a 3 dim

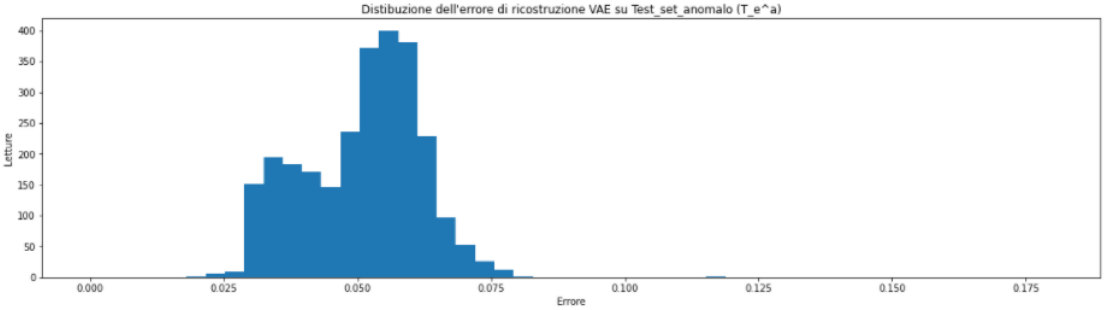

Figura 223 - Distribuzione dell'errore di ricostruzione su Test_set_anomalo (T_e^a) VAE non sup. a spazio latente a 3 dim

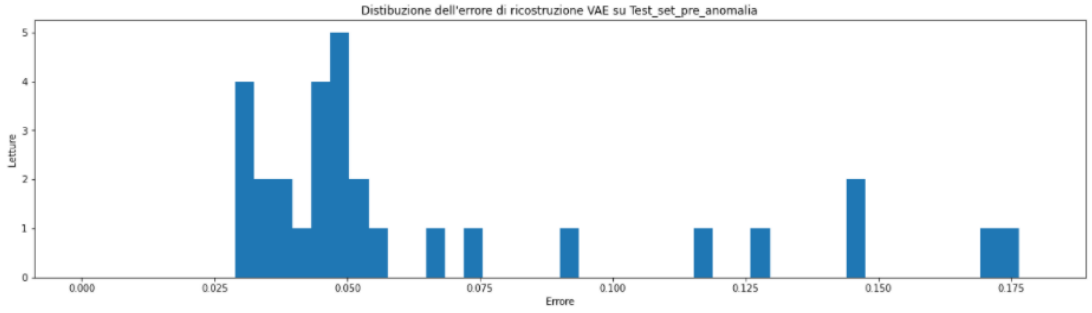

Figura 224 - Distribuzione dell'errore di ricostruzione su Test_set_pre_anomalia VAE non sup. a spazio latente a 3 dim

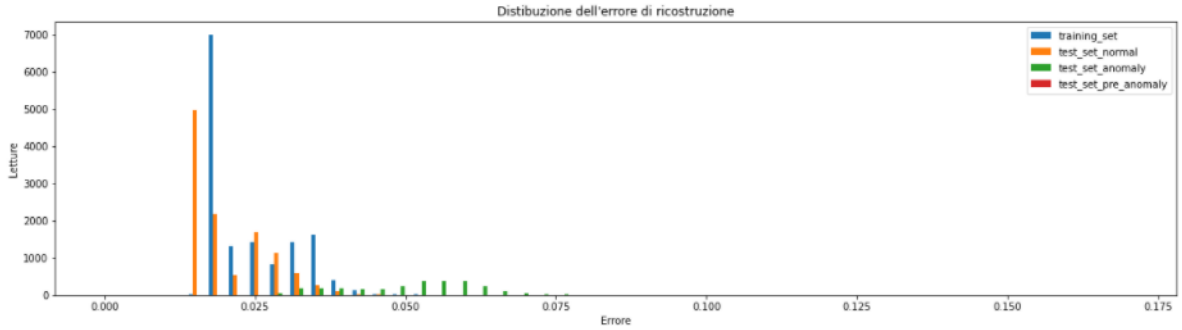

Figura 225 - Distribuzione errore di ricostruzione VAE non sup. a spazio latente a 3 dim

6.1.3 Proiezione dei dati pre-anomalia nello spazio latente

Con questa tipologia di autoencoder è interessate capire se immediatamente prima all'insorgenza di un'anomalia i dati vengono proiettati vicini nello spazio latente. Per fare questo si modifica la feature ***label*** dei punti che precedono l'anomalia di 2h e 30m con un valore 2. Il risultato ottenuto sia con VAE con spazio latente a due dimensioni che con VAE con spazio latente a tre dimensioni è il seguente:

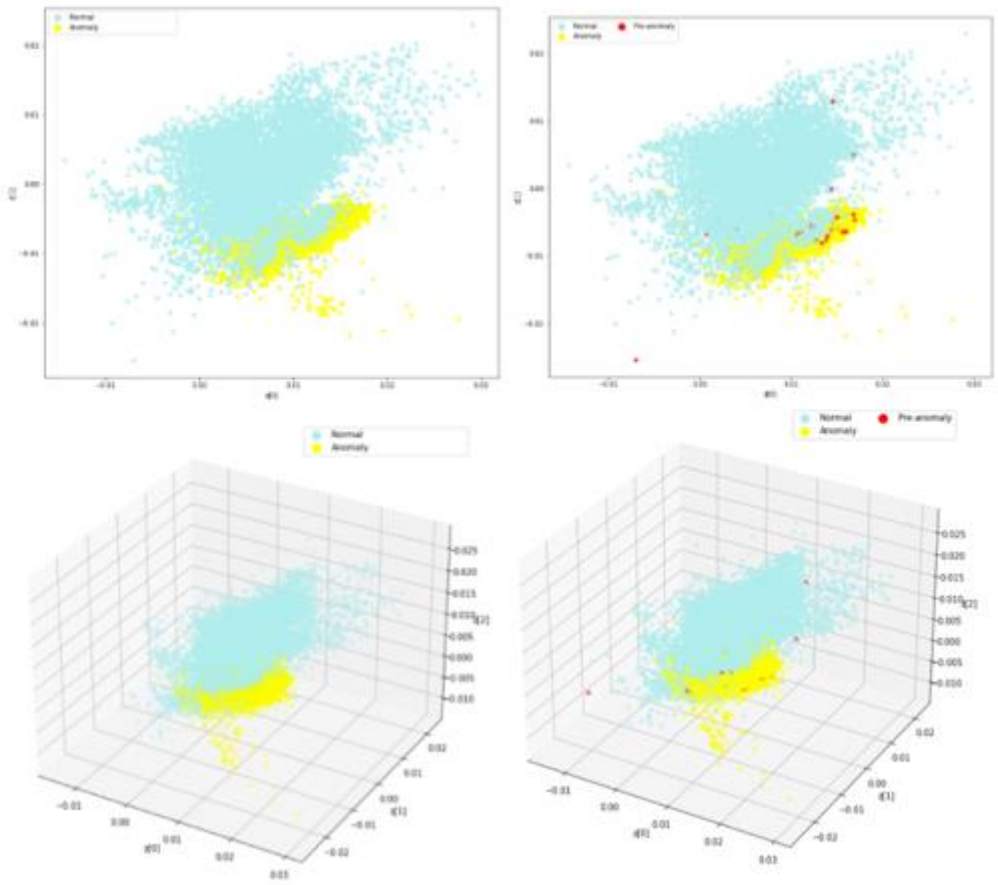

Figura 226 - Proiezione dei dati prima dell'anomalia VAE non sup.

Come si può notare dalla figura 49, i punti *vicini* (rossi) ad una anomalia vengono proiettati nello spazio latente nello stesso cluster ed in sovrapposizione con le anomalie stesse. Infatti nella figure a sinistra si può notare il plot effettuto solo con gli stati anomali/normali, mentre a destra si mostra quello con il dataset manipolato e si evince che le sezioni occupate dai punti anomali e da quelli precedenti sono sovrapposte e distribuite allo stesso modo.

6.1.4 Valutazione dell'errore di ricostruzione su tutti i nodi

Si è pensato di valutare l'autoencoder, con spazio latente a due dimensioni e a tre, tramite l'errore di ricostruzione anche su tutti gli altri dati dei nodi che si hanno a disposizione. Come si può notare dalla tabella seguente, l'errore di ricostruzione sul dataset contenente solo valori relativi a periodi anomali, risulta essere molto più alto rispetto a quello degli altri due set di dati. In generale si può evincere che training_set e test_set_normale hanno sempre valori molto vicini tra loro e questo è giustificato anche dal fatto che le letture valutate come normali sono molto di più rispetto a quelle dei periodi anomali. L'errore di ricostruzione valutato con l'autoencoder VAE, sia con spazio latente a due dimensioni che per quello con spazio latente a tre dimensione, per il training_set e test_set_normale varia tra 0.02 e 0.04, per il test_set_anomalo l'errore di ricostruzione varia tra 0.02 e 0.12, per il test_set_pre_anomalia l'errore di ricostruzione varia tra 0.04 e 0.15. Si evidenzia inoltre che l'autoencoder con spazio latente a due dimensioni risulta essere leggermente più performante rispetto a quello con spazio latente a tre dimensioni.

	Numero letture	**Errore di ricostruzione medio VAE non superv. spazio latente a 3 dim**	**Errore di ricostruzione medio VAE non superv. spazio latente a 2 dim**
Nodo: r205n20 Periodo: 2020-07-02 08:30:00 – 2021-03-03 13:45:00			
Training_set (T_r)	14230	0.02497	0.03252
Test_set_normale (T_e^n)	11556	0.02179	0.02825
Test_set_anomalo (T_e^a)	2674	0.04971	0.05627
Test_set_pre_anomalia	31	0.07496	0.07692
Nodo: r205n02 Periodo: 2020-07-22 15:45:00 – 2021-01-19 09:15:00			
Training_set (T_r)	8384	0.02094	0.03225
Test_set_normale (T_e^n)	6081	0.02545	0.04002
Test_set_anomalo (T_e^a)	2303	0.05215	0.05400
Test_set_pre_anomalia	No data	No data	No data

Nodo: r205n17 Periodo: 2020-05-31 22:15:00 – 2021-02-18 14:45:00			
Training_set (T_r)	5330	0.04081	0.04736
Test_set_normale (T_e^n)	5208	0.04042	0.04713
Test_set_anomalo (T_e^a)	122	0.09247	0.09557
Test_set_pre_anomalia	22	0.09586	0.10272
Nodo: r206n02 Periodo: 2020-07-06 12:30:00 – 2021-03-03 13:45:00			
Training_set (T_r)	10781	0.02582	0.03084
Test_set_normale (T_e^n)	9886	0.02647	0.03111
Test_set_anomalo (T_e^a)	895	0.04757	0.06243
Test_set_pre_anomalia	11	0.15952	0.17605
Nodo: r206n14 Periodo: 2020-05-31 22:15:00 – 2021-03-03 13:45:00			
Training_set (T_r)	7845	0.02360	0.02847
Test_set_normale (T_e^n)	7779	0.02030	0.02618
Test_set_anomalo (T_e^a)	66	0.07244	0.08399
Test_set_pre_anomalia	11	0.14175	0.14805
Nodo: r206n17 Periodo: 2020-05-31 22:15:00 – 2021-03-03 13:45:00			
Training_set (T_r)	11664	0.02517	0.03041
Test_set_normale (T_e^n)	11594	0.02050	0.02797
Test_set_anomalo (T_e^a)	70	0.04988	0.06001
Test_set_pre_anomalia	12	0.10453	0.10903
Nodo: r206n18 Periodo: 2020-05-31 22:00:00 – 2021-03-03 13:45:00			
Training_set (T_r)	11681	0.02070	0.02733
Test_set_normale (T_e^n)	11617	0.02075	0.02738
Test_set_anomalo (T_e^a)	64	0.06665	0.07083
Test_set_pre_anomalia	11	0.11956	0.12525
Nodo: r206n19 Periodo: 2020-05-31 22:00:00 – 2021-03-03 13:45:00			
Training_set (T_r)	11703	0.02115	0.02914
Test_set_normale (T_e^n)	11635	0.01706	0.02513
Test_set_anomalo (T_e^a)	68	0.06056	0.07087
Test_set_pre_anomalia	12	0.09746	0.10307
Nodo: r208n02 Periodo: 2020-05-31 22:00:00 – 2021-03-03 13:45:00			
Training_set (T_r)	11382	0.02308	0.02835
Test_set_normale (T_e^n)	11365	0.02332	0.02860
Test_set_anomalo (T_e^a)	17	0.08470	0.09554
Test_set_pre_anomalia	11	0.13358	0.13649
Nodo: r208n04 Periodo: 2020-05-31 22:00:00 – 2021-03-03 13:45:00			
Training_set (T_r)	6114	0.02149	0.02746
Test_set_normale (T_e^n)	6107	0.02151	0.02748
Test_set_anomalo (T_e^a)	7	0.12782	0.14308
Test_set_pre_anomalia	11	0.12720	0.13521
Nodo: r208n07 Periodo: 2020-07-10 11:30:00 – 2021-03-03 13:45:00			
Training_set (T_r)	10405	0.02343	0.02878
Test_set_normale (T_e^n)	10393	0.02364	0.02898
Test_set_anomalo (T_e^a)	12	0.09880	0.10390
Test_set_pre_anomalia	8	0.15839	0.16728
Nodo: r208n12 Periodo: 2020-05-31 22:15:00 – 2021-03-03 13:45:00			
Training_set (T_r)	13546	0.02618	0.03139
Test_set_normale (T_e^n)	13520	0.02619	0.03139

Test_set_anomalo (T_e^a)	26	0.08016	0.08802
Test_set_pre_anomalia	50	0.04356	0.05022
Nodo: r210n05 Periodo: 2020-05-31 22:15:00 – 2021-03-03 13:45:00			
Training_set (T_r)	13667	0.01744	0.02630
Test_set_normale (T_e^n)	8302	0.02030	0.02715
Test_set_anomalo (T_e^a)	5365	0.03293	0.03666
Test_set_pre_anomalia	39	0.06841	0.07430
Nodo: r211n19 Periodo: 2020-05-31 22:15:00 – 2021-03-03 13:45:00			
Training_set (T_r)	6670	0.03027	0.03535
Test_set_normale (T_e^n)	5588	0.02840	0.03401
Test_set_anomalo (T_e^a)	1082	0.05056	0.06236
Test_set_pre_anomalia	42	0.08192	0.09134
Nodo: r212n06 Periodo: 2020-05-31 22:00:00 – 2021-03-03 13:45:00			
Training_set (T_r)	10244	0.02016	0.02690
Test_set_normale (T_e^n)	10195	0.02016	0.02694
Test_set_anomalo (T_e^a)	49	0.06064	0.06576
Test_set_pre_anomalia	16	0.10820	0.11532
Nodo: r215n05 Periodo: 2020-05-31 22:00:00 – 2021-03-03 13:45:00			
Training_set (T_r)	17576	0.01736	0.02184
Test_set_normale (T_e^n)	12388	0.01717	0.02020
Test_set_anomalo (T_e^a)	5188	0.02663	0.03451
Test_set_pre_anomalia	41	0.05948	0.06400
Nodo: r218n03 Periodo: 2020-05-31 22:00:00 – 2021-03-03 13:45:00			
Training_set (T_r)	8300	0.02236	0.02905
Test_set_normale (T_e^n)	7332	0.02114	0.02806
Test_set_anomalo (T_e^a)	968	0.07054	0.06416
Test_set_pre_anomalia	28	0.07115	0.07906
Nodo: r224n20 Periodo: 2020-05-31 22:15:00 – 2021-03-03 13:45:00			
Training_set (T_r)	10792	0.02749	0.02991
Test_set_normale (T_e^n)	10634	0.02768	0.03015
Test_set_anomalo (T_e^a)	158	0.04818	0.05080
Test_set_pre_anomalia	21	0.10919	0.11288
Nodo: r239n12 Periodo: 2020-05-31 22:00:00 – 2021-03-03 13:45:00			
Training_set (T_r)	13495	0.02325	0.02919
Test_set_normale (T_e^n)	9102	0.02618	0.03085
Test_set_anomalo (T_e^a)	4393	0.04328	0.04579
Test_set_pre_anomalia	42	0.05844	0.06391
Nodo: r243n11 Periodo: 2020-05-31 22:15:00 – 2021-03-03 13:45:00			
Training_set (T_r)	12182	0.01761	0.02361
Test_set_normale (T_e^n)	11717	0.01815	0.02399
Test_set_anomalo (T_e^a)	465	0.04344	0.04189
Test_set_pre_anomalia	48	0.06793	0.07716
Nodo: r249n06 Periodo: 2020-05-31 22:00:00 – 2021-03-03 13:45:00			
Training_set (T_r)	14932	0.01861	0.02306
Test_set_normale (T_e^n)	14477	0.01913	0.02376
Test_set_anomalo (T_e^a)	455	0.02322	0.03587
Test_set_pre_anomalia	34	0.07857	0.08687
Nodo: r253n06 Periodo: 2020-06-16 13:00:00 – 2021-03-03 13:45:00			

Training_set (T_r)	9862	0.01948	0.02539
Test_set_normale (T_e^n)	9684	0.01949	0.02533
Test_set_anomalo (T_e^a)	178	0.04923	0.05737
Test_set_pre_anomalia	No data	No data	No data
Nodo: r254n01 Periodo: 2020-05-31 22:30:00 – 2021-03-03 13:45:00			
Training_set (T_r)	17191	0.02346	0.02913
Test_set_normale (T_e^n)	15873	0.02307	0.02838
Test_set_anomalo (T_e^a)	1318	0.03515	0.04062
Test_set_pre_anomalia	89	0.04740	0.05347
Nodo: r256n05 **Periodo**: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	11140	0.02148	0.02504
Test_set_normale (T_e^n)	11024	0.02153	0.02510
Test_set_anomalo (T_e^a)	116	0.05549	0.05837
Test_set_pre_anomalia	45	0.04110	0.04694

	Numero letture	**Errore di ricostruzione medio VAE non superv. spazio latente a 3 dim**	**Errore di ricostruzione medio VAE non superv. spazio latente a 2 dim**
Media calcolata su tutti i nodi			
Training_set (T_r)	269116	0.02306	0.02912
Test_set_normale (T_e^n)	243057	0.02296	0.02906
Test_set_anomalo (T_e^a)	26059	0.05925	0.06577
Test_set_pre_anomalia	635	0.09309	0.09979

Anche utilizzando un VAE non supervisionato si è riscontrato che l'errore di ricostruzione durante e prima di una anomalia risulta essere molto più alto rispetto a quello di un periodo normale. La feature label, che indentifica una anomalia quando assume valore 1, viene settata da un sys admin solo dopo che un problema viene riscontrato sul nodo e quindi è molto probabile che un nodo sia in stato anomalo senza che la sua label sia 1. Il vantaggio che si avrebbe utilizzando l'autoencoder è rilevare questa situazione e questo viene confermato dai valori riportati nella tabella precedente, infatti il VAE si accorge che sul nodo sta succedendo qualcosa di strano prima che la label venga impostata su "Anomalia" avendo un errore di ricostruzione sul valori pre_anomalia molto alto.

Per poter analizzare meglio l'errore di ricostruzione sulle singole feature, si è pensato di utilizzare un heatmap, col quale possiamo mettere in evidenza quali feature hanno avuto maggiori difficoltà ad essere ricostruite. Per una visualizzazione migliore si è scelto di mostrare solo alcune delle 232 feature. Osservando la figura seguente si deduce che le feature con errore di ricostruzione maggiore durante un periodo di anomalia (evidenziato dalla linea arancione in

alto) sono: ***avg:ps0_output_volta, avg:load_fifteen, avg:load_five, avg:load_one, avg: part_max_used, avg:proc_total, avg:swap_free, avg:fan0_1, avg:fan1_1, avg:fan2_1, avg:fan3_1, avg:state***. Si può anche notare che spesso prima dell'insorgere di una anomalia l'errore di ricostruzione aumenta, confermando cioè che è stato riportato in precedenza.

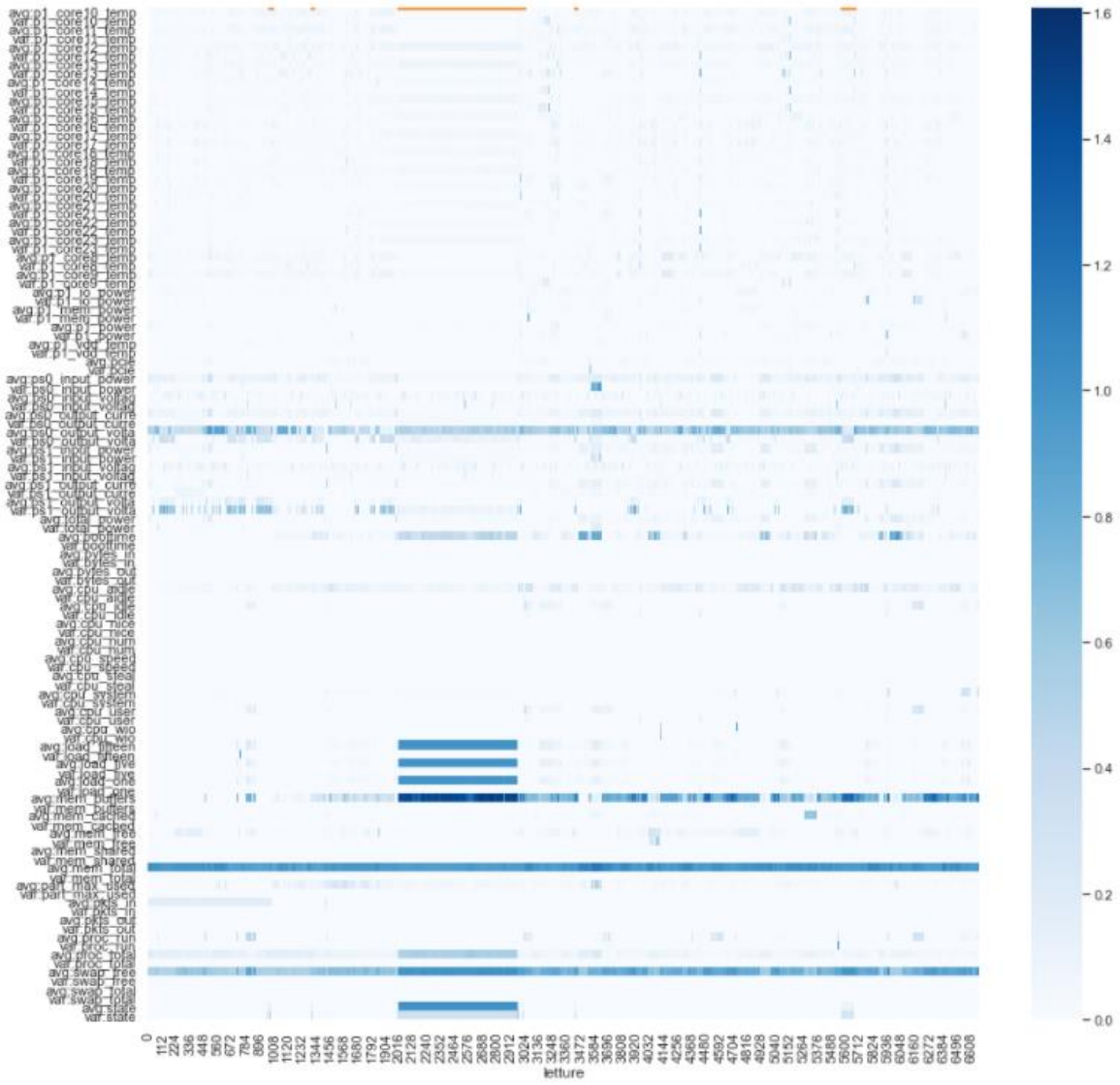

Figura 227 - Heatmap nodo n211n19 VAE non supervisionato con spazio latente a 3 dimensioni

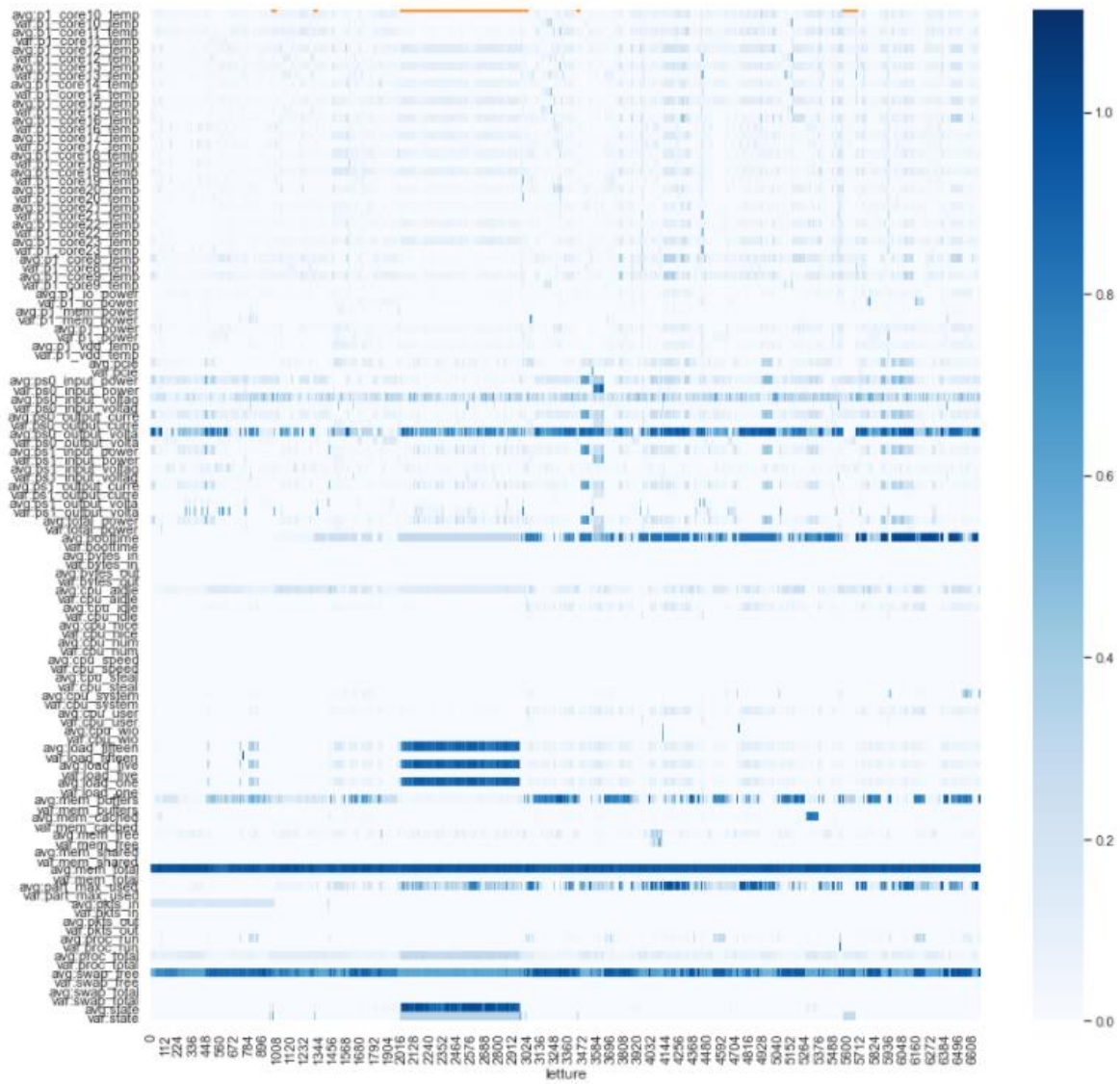

Figura 228 - Heatmap nodo n211n19 VAE non supervisionato con spazio latente a 2 dimensioni

6.2 Autoencoder variazionale semi supervisionato

Si è provato a rieseguire tutti gli esperimenti in modo semi supervisionato, quindi utilizzando per i training solo dati etichettati come normali. Osservando l'andamento dell'encoder si può notare che il valore di partenza della perdita è pari a circa 0.07 ed arriva, andando avanti con le epoche a circa 0.005, mentre l'errore sul validation set parte da 0.06 e arriva a 0.02 (Figura 229).

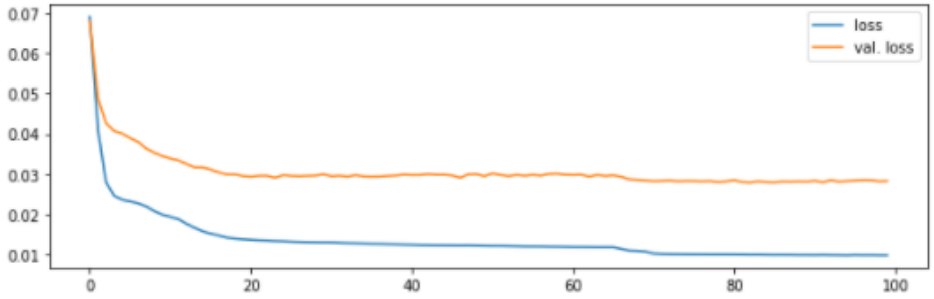

Figura 229 - Training con VAE semi supervisionato

Si è poi provato ad usare l'autoencoder su dati *nuovi* per osservare il suo comportamento tramite il raggruppamento in cluster in base alla variabile "label". Il nodo preso in analisi è il **r239n12**. Come si può osservare dalle figure seguenti, che rappresentano come si comportano le classi nello spazio latente, si può notare che le due classi individuate (anomalie in giallo, normali in azzurro) risultano essere raggruppate e distribuite nella stessa zona.

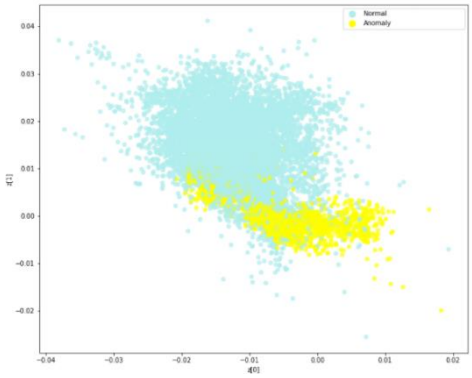

Figura 230 - Proiezione su spazio latente della variabile label VAE semi sup.

6.2.1 VAE con spazio latente a tre dimensioni semi supervisionato

Dopo aver eseguito queste prove, si è pensato di provare a plottare i punti in uno spazio latente a 3 dimensioni. Osservando l'andamento dell'encoder si può notare che il valore di partenza della perdita è pari a circa 0.07 ed arriva, andando avanti con le epoche a circa 0.005, mentre l'errore sul validation set parte da 0.05 e arriva a 0.02 (Figura 231).

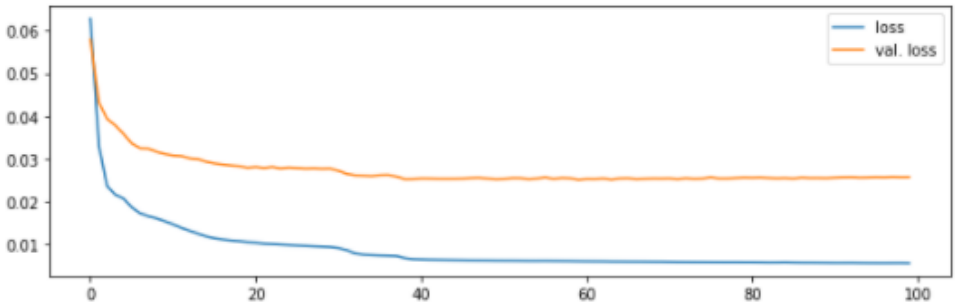

Figura 231 – Training con VAE semi sup. con spazio latente a tre dimensioni semi supervisionato

Si è poi provato ad usare l'autoencoder su dati "nuovi" per osservare il suo comportamento tramite il raggruppamento in cluster in base alla variabile "label". Il nodo preso in analisi è il **r239n12**.

Come si può osservare dalla figura seguente, si può notare che le due classi individuate (anomalie in giallo, normali in azzurro) risultano essere raggruppate e distribuite nella stessa zona.

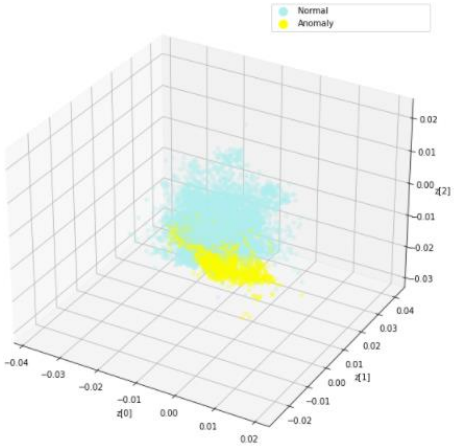

Figura 232 - Proiezione su spazio latente a tre dimensioni VAE semi sup. della variabile label

6.2.2 Proiezione dei dati pre-anomalia nello spazio latente

Si è provato a valutare se, immediatamente prima all'insorgenza di un'anomalia, i dati vengano proiettati vicini nello spazio latente. Per fare questo si modifica la feature "label" dei punti che precedono l'anomalia di 2h e 30m con un valore 2. Il risultato ottenuto sia con VAE con spazio latente a due dimensioni che con VAE con spazio latente a tre dimensioni è il seguente:

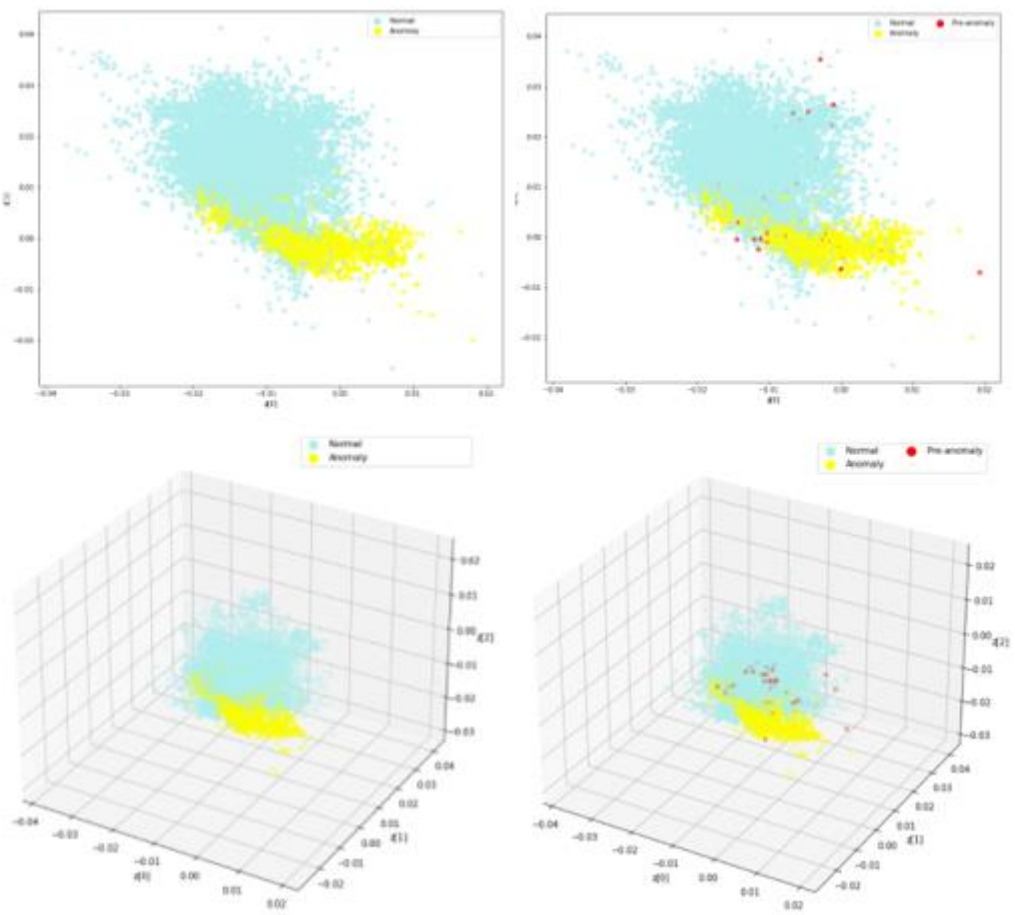

Figura 233 - Proiezione su spazio latente a tre dimensioni della variabile label VAE semi sup.

Come si può notare dalla figura 233, i punti vicini ad una anomalia vengono proiettati nello spazio latente nello stesso cluster ed in sovrapposizione con le anomalie stesse oppure nelle immediate vicinanze. Infatti nella figure a sinistra si può notare il plot effettuto solo con gli stati anomali/normali, mentre a destra si mostra quello con il dataset manipolato e si evince che le sezioni occupate dai punti anomali e da quelli precedenti sono sovrapposte e distribuite allo stesso modo.

6.2.3 Distribuzione dell'errore di ricostruzione

Per poter effettuare una migliore valutazione sul comportamento dell'autoencoder si è pensato di dividere il data set del nodo r205n20 che contiene i dati relativi al periodo dal 2020-07-02 08:30:00 al 2021-03-03 13:45:00, in *training_set(T_r), test_set con soli punti normali (T_e^n), test_set con solo punti anomali (T_e^a) e test_set_pre_anomalia* che contiene i dati di 2h e 30m precedenti alle anomalie e calcolare l'errore medio di ricostruzione su ognuno di essi. Di seguito viene mostrata la distribuzione dell'errore di distribuzioni nei vari casi presi analisi, prima per

il VAE con spazio latente a due dimensioni e in seguito per il VAE con spazio latente a tre dimensioni. Si evidenzia che nel caso del test set con anomalie i valori risultanti sono molto più alti, rispetto agli altri dataset, confermando che durante i periodi anomali il valore dell'errore di ricostruzione risulta essere più alto. Inoltre se si osserva la distribuzione dell'errore di ricostruzione sul dataset dei dati pre anomalia, si evince che i suoi valori sono più alti rispetto alla maggior parte dei valori dei dataset normale e di training.

6.2.3.1 Distribuzione dell'errore di ricostruzione su VAE con spazio latente a due dimensioni

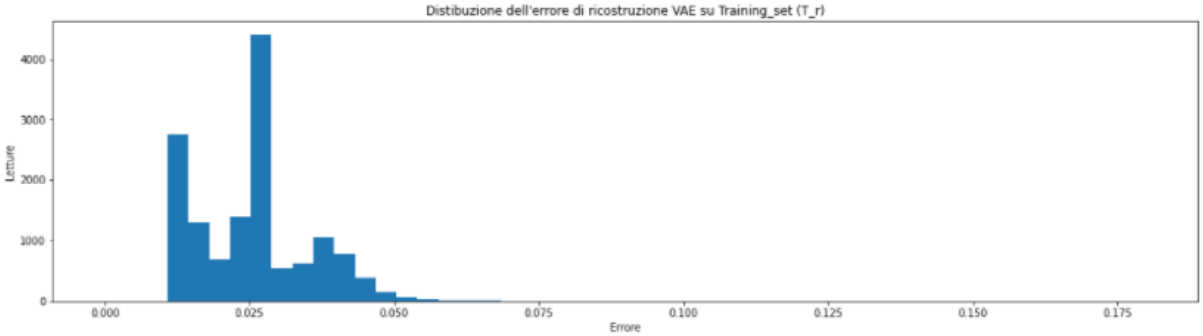

Figura 234 - Distribuzione dell'errore di ricostruzione su Training_set (T_r) VAE semi sup. a spazio latente a 2 dim

Figura 235 - Distribuzione dell'errore di ricostruzione su Test_set_normale (T_e^n) VAE semi sup. a spazio latente a 2 dim

Figura 236 - Distribuzione dell'errore di ricostruzione su Test_set_anomalo (T_e^a) VAE semi sup. a spazio latente a 2 dim

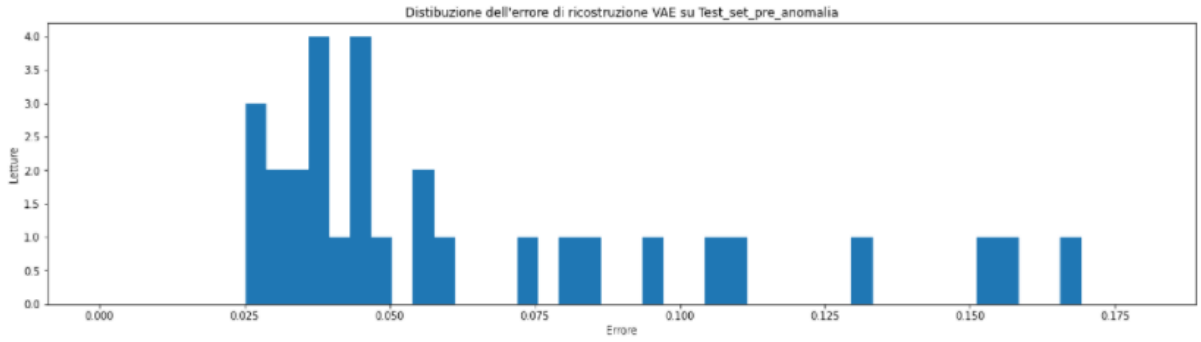

Figura 237 - Distribuzione dell'errore di ricostruzione su Test_set_pre_anomalia VAE semi sup. a spazio latente a 2 dim

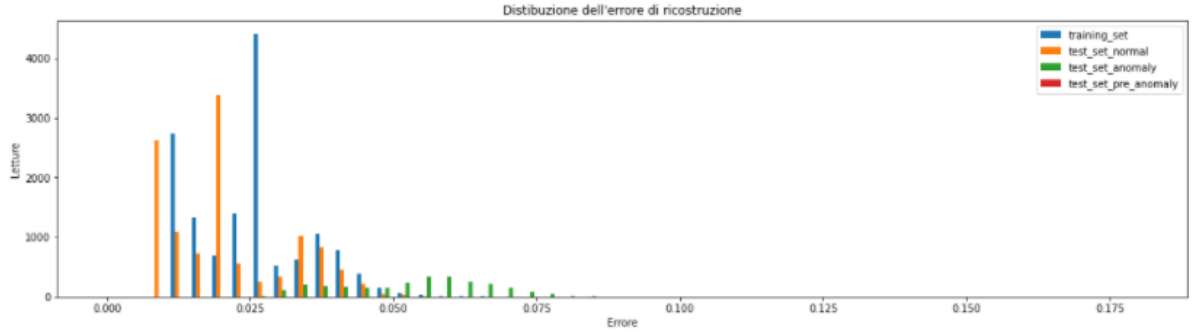

Figura 238 - Distribuzione errore di ricostruzione VAE semi sup. a spazio latente a 2 dim

6.2.3.2 Distribuzione dell'errore di ricostruzione su VAE con spazio latente a tre dimensioni

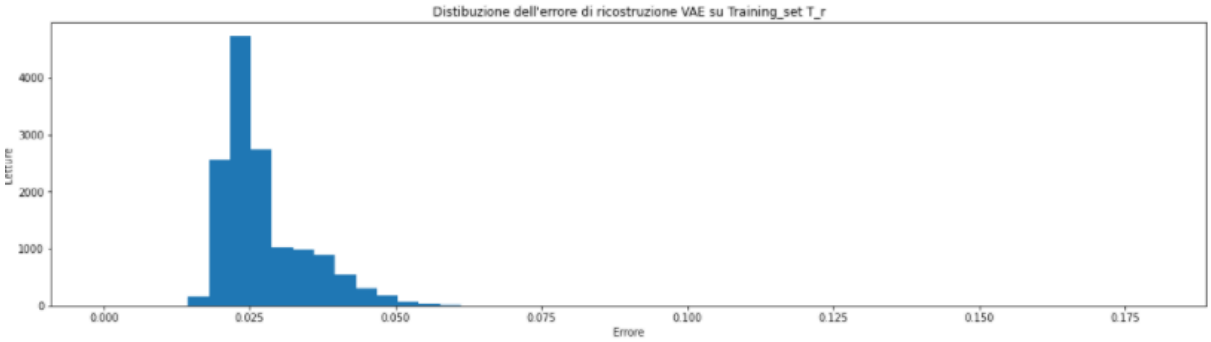

Figura 239 - Distribuzione dell'errore di ricostruzione su Training_set (T_r) VAE semi sup. a spazio latente a 3 dim

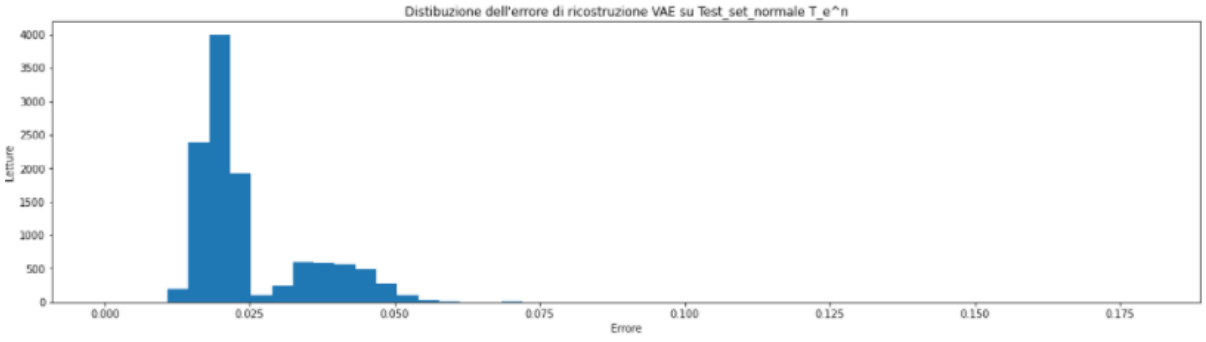

Figura 240 - Distribuzione dell'errore di ricostruzione su Test_set_normale (T_e^n) VAE semi sup. a spazio latente a 3 dim

Figura 241 - Distribuzione dell'errore di ricostruzione su Test_set_anomalo (T_e^a) VAE semi sup. a spazio latente a 3 dim

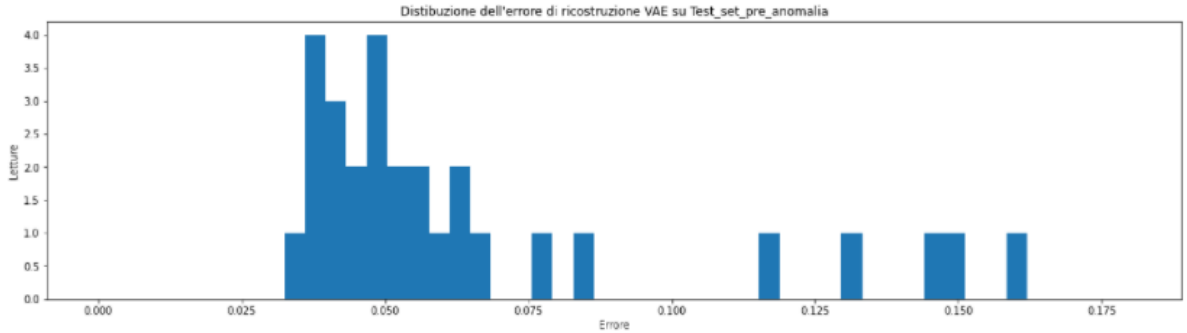

Figura 242 - Distribuzione dell'errore di ricostruzione su Test_set_pre_anomalia VAE semi sup. a spazio latente a 3 dim

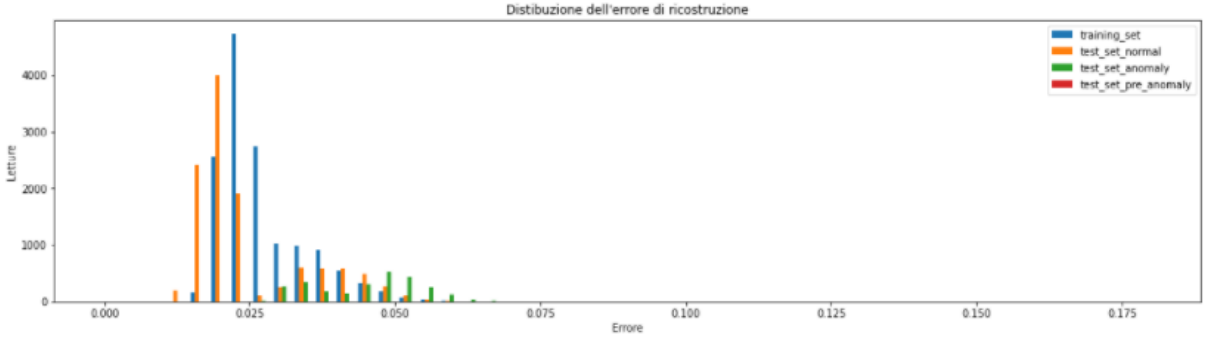

Figura 243 - Distribuzione errore di ricostruzione VAE semi sup. a spazio latente a 3 dim

6.2.4 Valutazione dell'errore di ricostruzione su tutti i nodi

Si è pensato di valutare l'autoencoder, con spazio latente a due dimensioni e a tre, tramite l'errore di ricostruzione anche su tutti gli altri dati dei nodi che si hanno a disposizione. Come si può notare dalla tabella seguente, l'errore di ricostruzione sul dataset contenente solo valori relativi a periodi anomali, risulta essere molto più alto rispetto a quello degli altri due set di dati. In generale si può evincere che training_set e test_set_normale hanno sempre valori molto vicini tra loro e questo è giustificato anche dal fatto che le letture valutate come normali sono molto di più rispetto a quelle dei periodi anomali. L'errore di ricostruzione valutato con l'autoencoder VAE, sia con spazio latente a due dimensioni che per quello con spazio latente a tre dimensione, per il training_set e test_set_normale varia tra 0.05 e 0.03, per il

test_set_anomalo l'errore di ricostruzione varia tra 0.04 e 0.16, per il test_set_pre_anomalia l'errore di ricostruzione varia tra 0.06 e 0.19. Si evidenzia inoltre che l'autoencoder con spazio latente a tre dimensioni risulta essere leggermente più performante rispetto a quello con spazio latente a due dimensioni.

	Numero letture	Errore di ricostruzione medio VAE semi superv. spazio latente a 3 dim	Errore di ricostruzione medio VAE semi superv. spazio latente a 2 dim
Nodo: r205n20 Periodo: 2020-07-02 08:30:00 - 2021-03-03 13:45:00			
Training_set (T_r)	14230	0.02723	0.02564
Test_set_normale (T_e^n)	11556	0.02472	0.02111
Test_set_anomalo (T_e^a)	2674	0.04539	0.05345
Test_set_pre_anomalia	31	0.07527	0.06957
Nodo: r205n02 Periodo: 2020-07-22 15:45:00 - 2021-01-19 09:15:00			
Training_set (T_r)	8384	0.01886	0.02366
Test_set_normale (T_e^n)	6081	0.02564	0.02729
Test_set_anomalo (T_e^a)	2303	0.04942	0.05273
Test_set_pre_anomalia	No data	No data	No data
Nodo: r205n17 Periodo: 2020-05-31 22:15:00 - 2021-02-18 14:45:00			
Training_set (T_r)	5330	0.03120	0.03305
Test_set_normale (T_e^n)	5208	0.03257	0.03252
Test_set_anomalo (T_e^a)	122	0.08862	0.07677
Test_set_pre_anomalia	22	0.09363	0.08767
Nodo: r206n02 Periodo: 2020-07-06 12:30:00 - 2021-03-03 13:45:00			
Training_set (T_r)	10781	0.02326	0.02651
Test_set_normale (T_e^n)	9886	0.02389	0.02739
Test_set_anomalo (T_e^a)	895	0.04853	0.05478
Test_set_pre_anomalia	11	0.14405	0.14974
Nodo: r206n14 Periodo: 2020-05-31 22:15:00 - 2021-03-03 13:45:00			
Training_set (T_r)	7845	0.02177	0.02989
Test_set_normale (T_e^n)	7779	0.01848	0.02268
Test_set_anomalo (T_e^a)	66	0.06545	0.07769
Test_set_pre_anomalia	11	0.13679	0.15007
Nodo: r206n17 Periodo: 2020-05-31 22:15:00 - 2021-03-03 13:45:00			
Training_set (T_r)	11664	0.02101	0.02743
Test_set_normale (T_e^n)	11594	0.01751	0.02221
Test_set_anomalo (T_e^a)	70	0.05006	0.05489
Test_set_pre_anomalia	12	0.10065	0.11204
Nodo: r206n18 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	11681	0.01736	0.02219
Test_set_normale (T_e^n)	11617	0.01740	0.02224
Test_set_anomalo (T_e^a)	64	0.06464	0.08614
Test_set_pre_anomalia	11	0.11605	0.12133
Nodo: r206n19 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	11703	0.02012	0.02746

Test_set_normale (T_e^n)	11635	0.01643	0.02371
Test_set_anomalo (T_e^a)	68	0.05461	0.07114
Test_set_pre_anomalia	12	0.09131	0.09838
Nodo: r208n02 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	11382	0.02189	0.02703
Test_set_normale (T_e^n)	11365	0.02206	0.02720
Test_set_anomalo (T_e^a)	17	0.07958	0.08115
Test_set_pre_anomalia	11	0.12974	0.13343
Nodo: r208n04 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	6114	0.01743	0.02053
Test_set_normale (T_e^n)	6107	0.01744	0.02055
Test_set_anomalo (T_e^a)	7	0.12987	0.12642
Test_set_pre_anomalia	11	0.12753	0.12423
Nodo: r208n07 Periodo: 2020-07-10 11:30:00 - 2021-03-03 13:45:00			
Training_set (T_r)	10405	0.02001	0.02787
Test_set_normale (T_e^n)	10393	0.02022	0.02811
Test_set_anomalo (T_e^a)	12	0.09131	0.09331
Test_set_pre_anomalia	8	0.15362	0.15505
Nodo: r208n12 Periodo: 2020-05-31 22:15:00 - 2021-03-03 13:45:00			
Training_set (T_r)	13546	0.02328	0.03373
Test_set_normale (T_e^n)	13520	0.02329	0.03376
Test_set_anomalo (T_e^a)	26	0.07788	0.07950
Test_set_pre_anomalia	50	0.03515	0.03546
Nodo: r210n05 Periodo: 2020-05-31 22:15:00 - 2021-03-03 13:45:00			
Training_set (T_r)	13667	0.02085	0.02387
Test_set_normale (T_e^n)	8302	0.02344	0.02834
Test_set_anomalo (T_e^a)	5365	0.03965	0.02700
Test_set_pre_anomalia	39	0.06252	0.07015
Nodo: r211n19 Periodo: 2020-05-31 22:15:00 - 2021-03-03 13:45:00			
Training_set (T_r)	6670	0.03001	0.03064
Test_set_normale (T_e^n)	5588	0.02742	0.02839
Test_set_anomalo (T_e^a)	1082	0.05736	0.06794
Test_set_pre_anomalia	42	0.08537	0.07885
Nodo: r212n06 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	10244	0.01723	0.02303
Test_set_normale (T_e^n)	10195	0.01722	0.02306
Test_set_anomalo (T_e^a)	49	0.05789	0.06107
Test_set_pre_anomalia	16	0.10377	0.11835
Nodo: r215n05 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	17576	0.01440	0.01786
Test_set_normale (T_e^n)	12388	0.01345	0.01819
Test_set_anomalo (T_e^a)	5188	0.03136	0.02376
Test_set_pre_anomalia	41	0.05611	0.05919
Nodo: r218n03 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	8300	0.02760	0.03234
Test_set_normale (T_e^n)	7332	0.02624	0.03200
Test_set_anomalo (T_e^a)	968	0.06407	0.07915
Test_set_pre_anomalia	28	0.06821	0.07609

Nodo: r224n20 Periodo: 2020-05-31 22:15:00 - 2021-03-03 13:45:00			
Training_set (T_r)	10792	0.02348	0.02867
Test_set_normale (T_e^n)	10634	0.02366	0.02895
Test_set_anomalo (T_e^a)	158	0.04635	0.05055
Test_set_pre_anomalia	21	0.10424	0.10624
Nodo: r239n12 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	13495	0.01943	0.02519
Test_set_normale (T_e^n)	9102	0.02229	0.03068
Test_set_anomalo (T_e^a)	4393	0.04167	0.05120
Test_set_pre_anomalia	42	0.05150	0.05777
Nodo: r243n11 Periodo: 2020-05-31 22:15:00 - 2021-03-03 13:45:00			
Training_set (T_r)	12182	0.01523	0.02308
Test_set_normale (T_e^n)	11717	0.01566	0.02375
Test_set_anomalo (T_e^a)	465	0.04004	0.04197
Test_set_pre_anomalia	48	0.06233	0.06666
Nodo: r249n06 Periodo: 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	14932	0.02565	0.03133
Test_set_normale (T_e^n)	14477	0.01639	0.02097
Test_set_anomalo (T_e^a)	455	0.03675	0.04592
Test_set_pre_anomalia	34	0.08188	0.08466
Nodo: r253n06 Periodo: 2020-06-16 13:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	9862	0.01911	0.02394
Test_set_normale (T_e^n)	9684	0.01915	0.02406
Test_set_anomalo (T_e^a)	178	0.05083	0.05457
Test_set_pre_anomalia	No data	No data	No data
Nodo: r254n01 Periodo: 2020-05-31 22:30:00 - 2021-03-03 13:45:00			
Training_set (T_r)	17191	0.02203	0.02687
Test_set_normale (T_e^n)	15873	0.02192	0.02667
Test_set_anomalo (T_e^a)	1318	0.03809	0.05091
Test_set_pre_anomalia	89	0.04412	0.05035
Nodo: r256n05 **Periodo:** 2020-05-31 22:00:00 - 2021-03-03 13:45:00			
Training_set (T_r)	11140	0.01593	0.02179
Test_set_normale (T_e^n)	11024	0.01603	0.02189
Test_set_anomalo (T_e^a)	116	0.05780	0.06359
Test_set_pre_anomalia	45	0.04067	0.04680

	Numero letture	**Errore di ricostruzione medio VAE semi superv. spazio latente a 3 dim**	**Errore di ricostruzione medio VAE semi superv. spazio latente a 2 dim**
Media calcolata su tutti i nodi			
Training_set (T_r)	269116	0.02135	0.02640
Test_set_normale (T_e^n)	243057	0.02115	0.02581
Test_set_anomalo (T_e^a)	26059	0.05863	0.06356
Test_set_pre_anomalia	635	0.08929	0.09327

Anche con l'utilizzo di VAE semi supervisionato, l'errore di ricostruzione durante e prima di una anomalia risulta essere molto più alto rispetto a quello di un periodo normale. La feature label, che indentifica una anomalia quando assume valore 1, viene settata da un sys admin solo dopo che un problema viene riscontrato sul nodo e quindi è molto probabile che un nodo sia in stato anomalo senza che la sua label sia 1. Il vantaggio che si avrebbe utilizzando l'autoencoder è rilevare questa situazione e questo viene confermato dai valori riportati nella tabella precedente, infatti il VAE si accorge che sul nodo sta succedendo qualcosa di strano prima che la label venga impostata su "Anomalia" avendo un errore di ricostruzione sul valori pre_anomalia molto alto.

Per poter analizzare meglio l'errore di ricostruzione sulle singole feature, si è pensato di utilizzare un heatmap, col quale possiamo mettere in evidenza quali feature hanno avuto maggiori difficoltà ad essere ricostruite. Per una visualizzazione migliore si è scelto di mostrare solo alcune delle 232 feature. Osservando la figura seguente si deduce che le feature con errore di ricostruzione maggiore durante un periodo di anomalia (evidenziato dalla linea arancione in alto) sono: ***avg:ps0_output_volta, avg:load_fifteen, avg:load_five, avg:load_one, avg:part_max_used, avg:proc_total, avg:swap_free, avg:fan0_1, avg:fan1_1, avg:fan2_1, avg:fan3_1, avg:state***. Si può anche notare che spesso prima dell'insorgere di una anomalia l'errore di ricostruzione aumenta, confermando cioè che è stato riportato in precedenza.

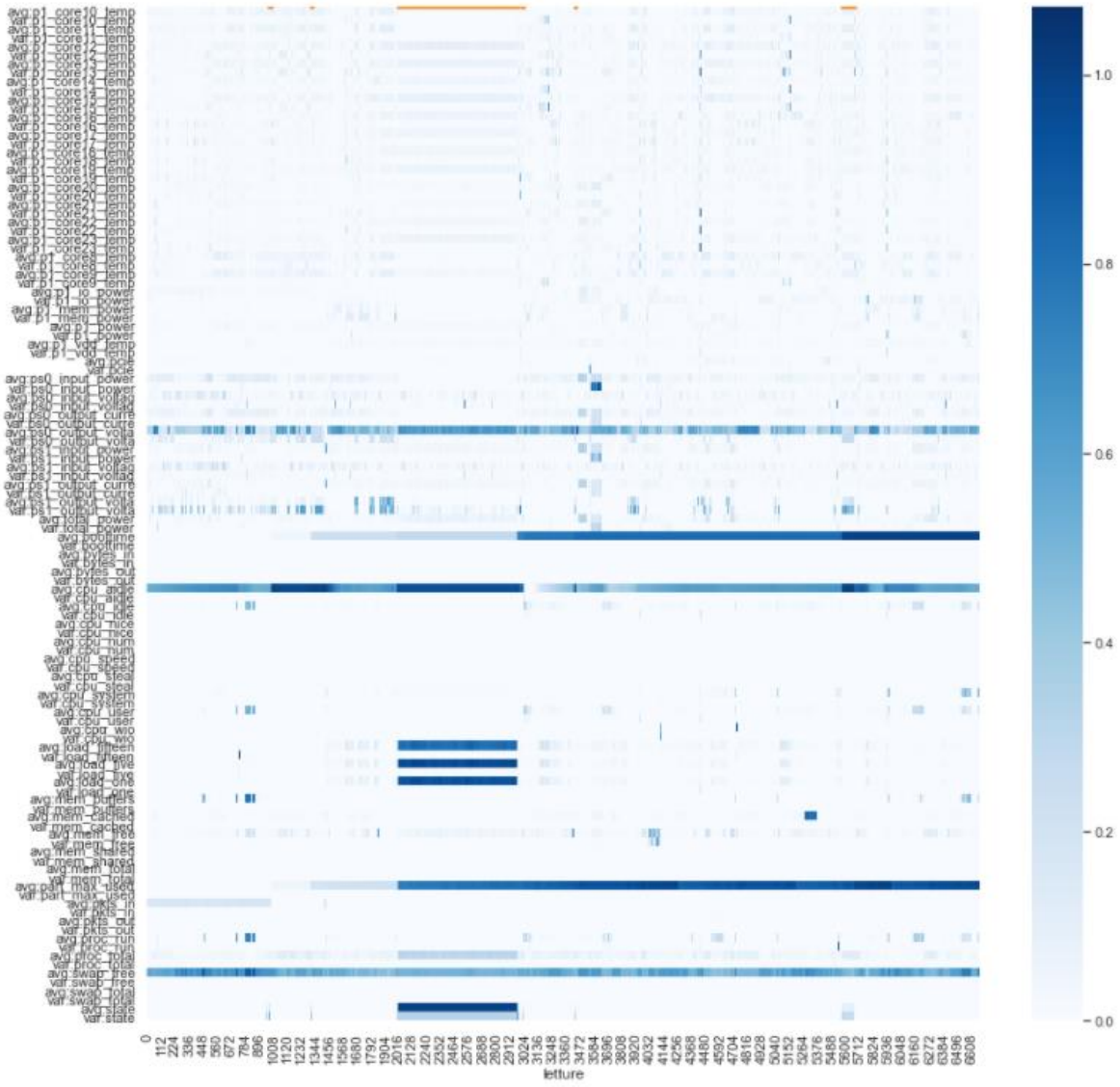

Figura 244 - Heatmap nodo n211n19 VAE non supervisionato con spazio latente a 3 dimensioni

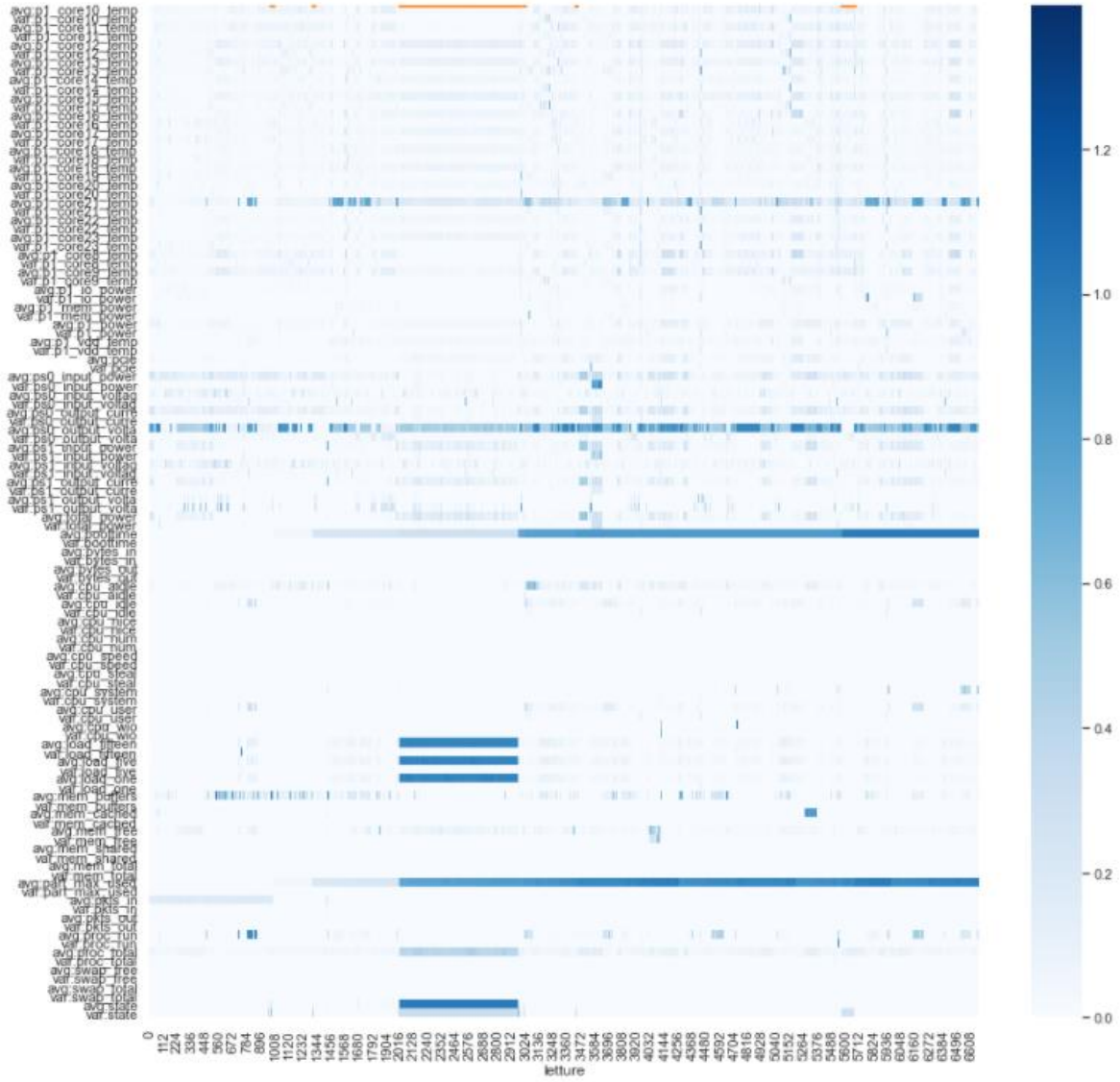

Figura 245 - Heatmap nodo n211n19 VAE non supervisionato con spazio latente a 2 dimensioni

CAPITOLO 7

Confronto tra i modelli realizzati

In questa sezione si andrà ad analizzare quale degli autoencoder presi in analisi si comporta meglio con i dati dei nodi che si hanno a disposizione. Il confronto si baserà sul valore dell'errore di ricostruzione dei vari autoencoder, in particolare, in un primo momento si avrà un paragone grafico attraverso degli istogrammi che rappresentano la distribuzione dell'errore di ricostruzione, mettendo in evidenza con colori diversi i valori relativi a periodi normali (blu) e quelli relativi a periodi anomali (arancione), poi si passerà ad un confronto quantitativo attraverso una tabella in cui sono riportati i valori numerici relativi all'errore di ricostruzione.

7.1 Confronto grafico

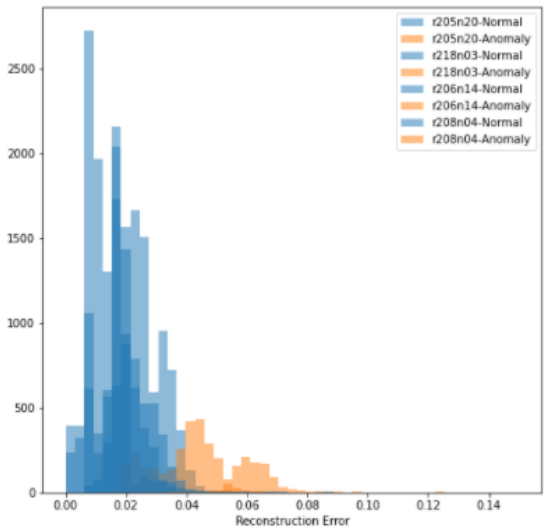

Figura 247 - Autoencoder semplice con MSE

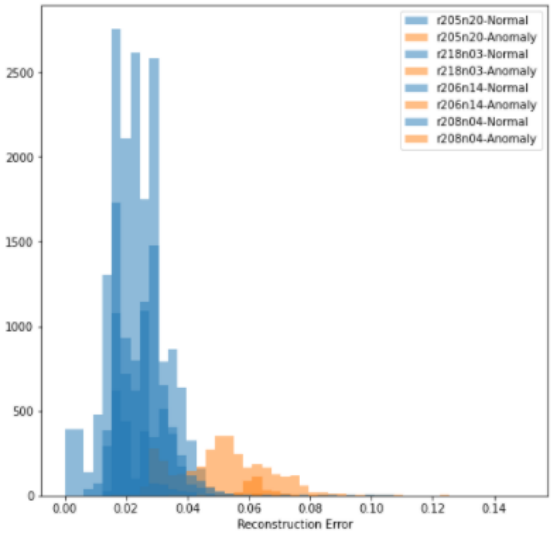

Figura 246 - Autoencoder semplice con MAE

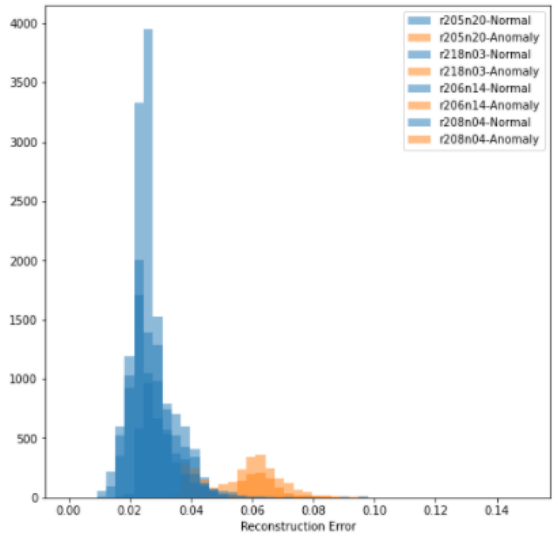

Figura 250 - VAE non super. con spazio latente a 2 dim.

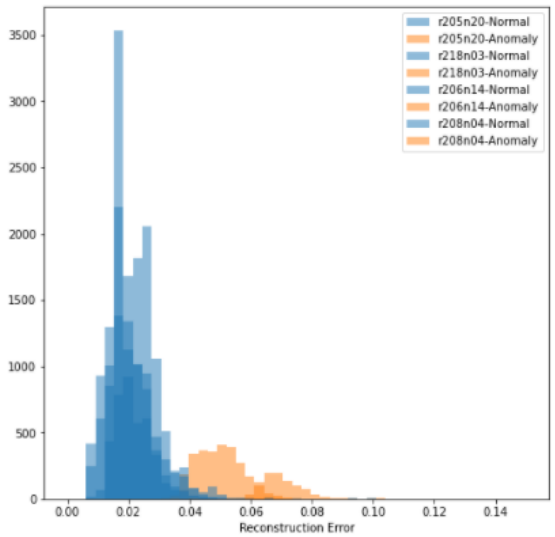

Figura 251 - VAE non super. con spazio latente a 3 dim

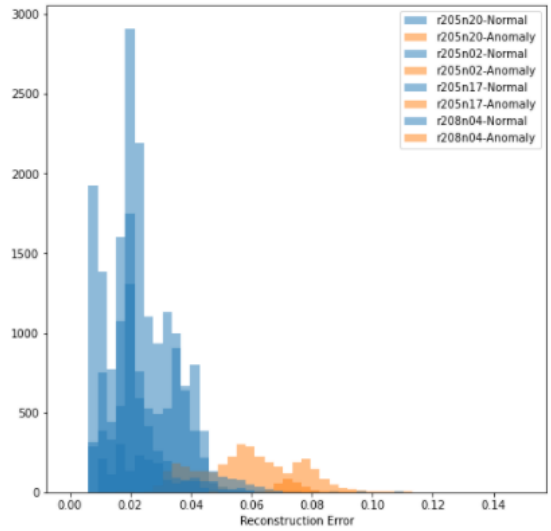

Figura 248 - VAE super. con spazio latente a 2 dim

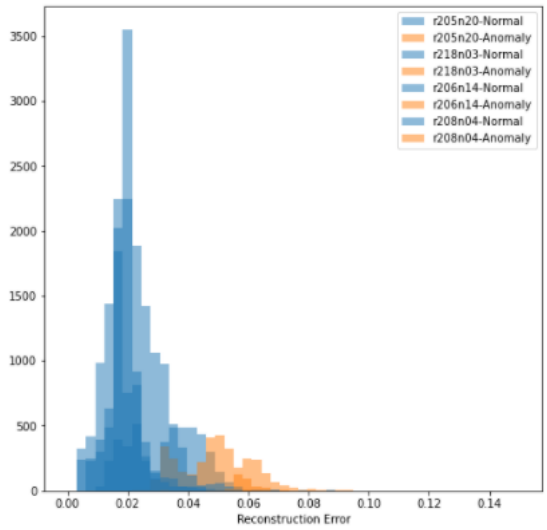

Figura 249 - VAE super. con spazio latente a 3 dim

Nelle figure precedenti si analizza l'errore di ricostruzione in tutti gli autoencoder allenati nella sezione precedente per valutare quale di questi risulti più performate e utile allo scopo di effettuare anomaly detection in un HPC system. Quello che subito si può notare è che con tutti gli approcci usati, si riesce ad avere una netta divisione tra i valori normali e quelli anomali in quanto l'area blu (valori normali) è compresa tra i valori 0 e 0.04, mentre l'area arancione (valori anomali) è compresa tra 0.04 e 0.14; mentre con l'approccio AE semi supervisionato (figure 246-247) e con VAE non sipervisonato (figure 249-250) questa suddivisone risulta molto netta, con VAE semi supervisionato le due aree sono spesso sovrapposte e solo dopo il valore 0.06 si ottiene una distinzione netta. Se si osserva la perdita della fase di training, i vari approcci risultano essere quasi equivalenti, solo l'AE allenato con una funzione di perdita MAE è meno performate.

7.2 Confronto quantitativo

	Perdita train_set	Perdita validation_set
AE con MSE	da 0.07 a 0.005	Da 0.05 a 0.02
AE con MAE	da 0.14 a 0.02	da 0.13 a 0.07
VAE non sup. con spazio latenete a 2 dim (MSE)	da 0.07 a 0.007	da 0.04 a 0.02
VAE non sup. con spazio latenete a 3 dim (MSE)	da 0.07 a 0.007	da 0.04 a 0.02
VAE sup. con spazio latenete a 2 dim (MSE)	da 0.07 a 0.005	da 0.06 a 0.02
VAE sup. con spazio latenete a 3 dim (MSE)	da 0.07 a 0.005	da 0.05 a 0.02

Con tutti e tre gli autoencoder, l'errore di ricostruzione durante e prima di una anomalia risulta essere molto più alto rispetto a quello di un periodo normale, questo significa che si accorgono che sul nodo sta succedendo qualcosa di strano prima che la label venga impostata su "Anomalia" avendo un errore di ricostruzione sui valori pre_anomalia molto alto. Confrontando i vari AE in analisi si evidenzia che l'approccio AE semi supervisionato con funzione di perdita MSE, ha un errore di ricostruzione medio calcolato sui vari nodi minore rispetto agli autoencoder VAE, come si può notare nella tabella seguente.

Errore di ricostruzione medio calcolato su tutti i nodi							
	Numero letture	**AE MSE**	**AE MAE**	**VAE non superv. spazio latente a 3 dim (MSE)**	**VAE non superv. spazio latente a 2 dim (MSE)**	**VAE semi superv. spazio latente a 3 dim (MSE)**	**VAE semi superv. spazio latente a 2 dim (MSE)**
Training_set (T_r)	269116	0.01617	0.02280	0.02306	0.02912	0.02135	0.02640
Test_set_normale (T_e^n)	243057	0.01582	0.02233	0.02296	0.02906	0.02115	0.02581
Test_set_anomalo (T_e^a)	26059	0.04515	0.05608	0.05925	0.06577	0.05863	0.06356
Test_set_pre_anomalia	635	0.07497	0.09175	0.09309	0.09979	0.08929	0.09327

Nella tabella seguente si identificano le feature che risultano avere un errore di ricostruzione maggiore durante un periodo di anomalia confrontate per i vari modelli.

	AE semplice	VAE non sup.	VAE semi sup.
avg:dimm15_temp	SI	NO	NO
avg:dimm3_temp	SI	NO	NO
avg:p0_core15_temp	SI	NO	NO
avg:p0_core21_temp	SI	NO	NO
avg:p0_core22_temp	SI	NO	NO
avg:p0_core23_temp	SI	NO	NO
avg:p0_core2_temp	SI	NO	NO
avg:p0_core3_temp	SI	NO	NO
avg:p0_core8_temp	SI	NO	NO
avg:p0_core9_temp	SI	NO	NO
avg:ps0_output_volta	SI	SI	SI
avg:load_fifteen	SI	SI	SI
avg:load_five	SI	SI	SI
avg:load_one	SI	SI	SI
avg: part_max_used	SI	SI	SI
avg:proc_total	SI	SI	SI
avg:swap_free	SI	SI	SI
avg:state	SI	SI	SI
avg:fan0_1	NO	SI	SI
avg:fan1_1	NO	SI	SI
avg:fan2_1	NO	SI	SI
avg:fan3_1	NO	SI	SI

Da questa analisi si può affermare che se si vuole ottenere una divisione in ***cluster*** dei dati, l'approccio migliore da usare è sicuramente il **VAE** poiché anomalie e situazioni normali vengono plottate nello spazio in gruppi separati, mentre se si vuole utilizzare ***l'errore di ricostruzione per classificare i dati***, sicuramente più performante è l'approccio con un deep **AE con funzione di perdita MSE**.

CAPITOLO 8

Conclusioni e sviluppi futuri

Il rilevamento delle anomalie e la predizione dei guasti fornisce un grande vantaggio nell'ottica dei sistemi HPC e non solo per questi, in quanto aiuta ad ottenere prestazioni alte, ad effettuare gli interventi sulle infrastrutture quando è necessario, a ridurre i fermi e quindi ad ottenere anche un beneficio di tipo economico. In questo lavoro si è effettuata un'analisi preliminare che ha portato a risultati accettabili e che possono essere la base per ulteriori studi e approfondimenti sul funzionamento sistema.

I risultati ottenuti nella valutazione iniziale dei dati hanno portato in risalto le feature del sistema che subiscono delle variazioni significative durante un periodo di anomalia e che quindi potrebbero essere usate come segnale di allarme per identificare eventuali malfunzionamenti. Inoltre altro aspetto significativo derivato da questa fase preliminare è che spesso le anomalie si presentano nei vari nodi del sistema negli stessi intervalli temporali, in particolare se i nodi in questione appartengono allo stesso rack.

Nella seconda fase si sono creati diversi modelli di Machine Learning con lo scopo di addestrali per apprendere il comportamento del sistema e di valutarne le capacità di riproduzione dell'output . I diversi approcci usati si sono mostrati tutti abbastanza promettenti, in particolare perché andando ad analizzare l'errore di ricostruzione tra i valori di input e quelli predetti dai modelli, esso in ogni caso, risulta essere molto più alto in corrispondenza di valori provenienti da periodi anomali. Un altro dato che può essere utile allo scopo di manutenzione predittiva è che anche i valori precedenti all'anomalia presentano un errore di ricostruzione molto alto e quindi i modelli riescono a capire che qualcosa nel sistema sta iniziando a non funzionare. Quest'ultima osservazione è da verificare anche con altri test e valutazioni poiché nel dataset fornito, non sempre si avevano i dati precedenti alle anomalie. Inoltre, andando a valutare i risultati dell'errore di ricostruzione sulle singole feature del sistema, si può notare quali di esse hanno più difficoltà ad essere ricostruite e si evidenza anche che per alcune, il valore dell'errore risulta essere molto più altro in prossimità e durante l'anomalia.

In questo lavoro di tesi è stata eseguita una prima analisi sui dati del MARCONI100, in futuro questa potrà essere espansa prendendo in considerazione anche altri aspetti. Inoltre si potranno ottimizzare i modelli creati in modo tale da migliorare le prestazioni. Un altro sviluppo futuro potrebbe essere andare ad addestrare dei modelli di Machine Learning differenti e confrontare i risultati ottenuti con quelli presi in analisi in questo elaborato. L'obiettivo finale di tutti questi sviluppi è arrivare a creare, un algoritmo per effettuare predictive maintenance e anomaly prediction sul supercomputer MARCONI100 e quindi per poter individuare con anticipo anomalie sul sistema.

ELENCO DELLE FIGURE

BIBLIOGRAFIA

[1] M. K. Pektürk e M. Ünal, «Performance-Aware High-Performance Computing for Remote Sensing Big Data Analytics».

[2] C. W. Italia, «Cos'è l'High Performance Computing (HPC),» 2016. [Online]. Available: https://www.cwi.it/data-center/high-performance-computing-hpc/cose-lhigh-performance-computing-hpc-83881.

[3] G. Pfister, In Search of Clusters, 2nd Edition, NJ: Prentice Hall PTR, 1998.

[4] M. Bakery e R. Buyya, «Cluster Computing at a Glance».

[5] D. Quintero, M. Gonzalez, A. Hussein e J. Myklebust, «IBM High-Performance Computing Insights with IBM Power System AC922 Clustered Solution,» 2019.

[6] «Cineca,» [Online]. Available: https://www.hpc.cineca.it/content/about-us .

[7] [Online]. Available: https://www.top500.org/lists/top500/list/2020/11/?page=1.

[8] [Online]. Available: https://jupyter.org/.

[9] S. Agrawal e J. Agrawal, Survey on Anomaly Detection using Data Mining Techniques, Procedia Computer Science,Volume 60, 2015.

[10] V. Chandola, A. Banerjee e V. Kumar, « Anomaly Detection: A Survey,» ACM Comput. Surv., 2009.

[11] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano e L. Benini., «A Semisupervised Autoencoder-based Approach for Anomaly Detection in High Performance Computing Systems,» 2019.

[12] J. L. Hennessy e D. A. Patterson, «Computer Architecture, Sixth Edition: A Quantitative Approach, 6th».

[13] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano e L. Benini, «Anomaly detection using autoencoders in high performance computing systems,» Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 9428–9433., 2019.

[14] F. Tempia e G. S., «Manutenzione predittiva: dati e AI per aumentare l'efficienza nel manifatturiero,» 2020.

[15] T. M. Mitchell, Machine Learning, McGraw-Hill,, 1997.

[16] I. Goodfellow, Y. Bengio e A. Courville, Deep learning, MIT press, 2016.

[17] M. Mohssen, M. Khan e E. Bashier, Machine Learning: Algorithms and Applications, 2016.

[18] [Online]. Available: https://www.ibm.com/cloud/learn/supervised-learning.

[19] I. Muhammad e Z. Yan, «SUPERVISED MACHINE LEARNING APPROACHES: A SURVEY,» *International Journal of Soft Computing*, 2015.

[20] J. Breuker, R. Dieng-Kuntz, N. Guarino, J. Kok, J. Liu, R. Lopez de Mántaras, R. Mizoguchi, M. Musen e N. Zhong, Frontiers in Artificial Intelligence and Applications, Vol. 160, 2007.

[21] Y. Chen e L. Hung, «Using decision trees to summarize associative classification rules,» *Expert Systems with Applications, Vol. 36,* 2009.

[22] I. Good, «Probability and the Weighing of Evidence,» *The University of Wisconsin - Madison: Charles Griffin*, 1950.

[23] S. Kotsiantis, Supervised Machine Learning: A Review of Classification Techniques, Informatica, Vol. 31, 2007.

[24] D. Bzdok, M. Krzywinski e N. Altman, «Machine learning: supervised methods,» 2018. [Online]. Available: https://doi.org/10.1038/nmeth.4551.

[25] N. Gupta, «Artificial Neural Network, Vol.3,,» *Selected from International Conference on Recent Trends in Applied Sciences with Engineering Applications*, 2013.

[26] A. Dongare, R. Kharde e A. Kachare., «Introduction to Artificial Neural Network., Issue 1,» *International Journal of Engineering and Innovative Technology (IJEIT) Volume 2,* 2012.

[27] J. McGonagle, J. García e S. Mollick., « Feedforward Neural Networks,» *Brilliant.org*, 2021.

[28] E. Palmisano, *A First Study of Transferable and Explainable Deep Learning Models for HPC Systems*, Master's thesis, Università di Bologna, 2019.

[29] Great Learning Team, *Types of Neural Networks and Definition of Neural Network,*, 2020.

[30] S. Saha, A Comprehensive Guide to Convolutional Neural Networks, 2018.

[31] M. Garbade, Understanding K-means Clustering in Machine Learning, 2018.

[32] [Online]. Available: https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning.

[33] E. Pegoraro, «Statistica per Data Science con R - V. 03,» 2019, p. Capitolo 9.

[34] Chire - Own work, «CC BY-SA 3.0,» [Online]. Available: https://commons.wikimedia.org/w/index.php?curid=17045963.

[35] [Online]. Available: https://www.intelligenzaartificiale.it/machine-learning/.

[36] L. Kaelbling, M. Littman e A. Moore, «Reinforcement Learning: A Survey.,» *Journal of Artificial Intelligence Research 4*, 1996.

[37] O. Chapelle, B. Schölkopf e A. Zien, Semi-Supervised Learning (Adaptive Computation and Machine Learning series), The MIT Press, 2006.

[38] N. Hubens, Introduzione agli autoencoder, DLI, 2018.

[39] J. Rocca, Understanding Variational Autoencoders (VAEs), 2019.

[40] A. Dertat, «Applied Deep Learning - Part 3: Autoencoders,» 2017. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798#3f72.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot e E. Duchesnay, «Scikit-learn: Machine Learning in Python,» *Journal of Machine Learning Research, vol.12,* 2011.

[42] NN-SVG. [Online]. Available: https://alexlenail.me/NN-SVG.

[43] [Online]. Available: https://netai.it/guida-rapida-alle-funzioni-di-attivazione-nel-deep-learning/#page-content.

[44] datascience.eu. [Online]. Available: https://datascience.eu/it/apprendimento-automatico/funzione-di-attivazione-relu/.

[45] D. Kingma e J. Ba, Adam: A method for stochastic optimization, ICLR, 2015.

[46] Keras. [Online]. Available: https://keras.io/api/losses.