# ALMA MATER STUDIORUM
# UNIVERSITÁ DI BOLOGNA

## SCUOLA DI INGEGNERIA E ARCHITETTURA

DIPARTIMENTO INFORMATICA - SCIENZA E INGEGNERIA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA


**TESI DI LAUREA**

in

*Intelligent Systems M*

# A First Study of Transferable and Explainable Deep Learning Models for HPC Systems


Candidato:

**Enzo Pio Palmisano**

Relatore:

Prof.ssa **Michela Milano**

Correlatore:

Dott. **Andrea Borghesi**



Anno Accademico 2018-2019

Sessione III

# Indice

# Introduzione

Il lavoro descritto in questa tesi è basato sullo studio di modelli di Deep Learning applicati all'anomaly detection per la rilevazioni di stati anomali nei sistemi HPC (High-Performance Computing). In particolare, l'obiettivo è studiare la trasferibilità di un modello e di spiegarne il risultato prodotto attraverso tecniche dell'Explainable artificial intelligence.
I sistemi HPC sono dotati di numerosi sensori in grado di monitorare il corretto funzionamento in tempo reale. Tuttavia, a causa dell'elevato grado di complessità di questi sistemi, si rende necessario l'uso di tecniche innovative e capaci di prevedere guasti, errori e qualsiasi tipo di anomalia per ridurre i costi di manutenzione e per rendere il servizio sempre disponibile. Negli anni ci sono stati numerosi studi per la realizzazioni di modelli di Deep Learning utili allo scopo, ma ciò che manca a questi modelli è la capacità di generalizzare a condizioni diverse da quelle riscontrate durante la fase di training.
Nella prima parte di questo lavoro andremo, quindi, ad analizzare un modello già sviluppato per studiarne la trasferibilità e la sua generalizzazione per un'applicazione più ampia rispetto al dominio su cui è costruito.
Un ulteriore problema è dato dalle modalità di risposta di un modello ad un determinato input. Molto spesso risulta essere incomprensibile anche per coloro che hanno costruito il modello, la risposta prodotta. Perciò, attraverso le tecniche dell'Explainable artificial intelligence, vengono studiati e analizzati i vari output del modello per comprenderne il funzionamento e motivare i risultati corretti ed errati.

Il sistema HPC che andremo ad analizzare è chiamato MARCONI ed è di proprietà del consorzio no-profit Cineca di Bologna. Il Cineca è composto da 70 università italiane, quattro centri di ricerca nazionali e il Ministero di Università e ricerca (MIUR). Cineca rappresenta uno dei più potenti centri di supercalcolo per la ricerca scientifica in Italia.
La struttura della tesi è cosi definita:

- Il capitolo 1 descrive i sistemi HPC ed in particolare il sistema Marconi.
- Nel capitolo 2 si introduce il tema dell'anomaly detection con le varie tecniche allo stato dell'arte.
- Nel capitolo 3 si descrive il Machine Learning e i suoi algoritmi, quelli applicati all'anomaly detection e quelli utilizzati in questo lavoro.
- Nel capitolo 4 viene presentata la tecnica di estrazione dei dati dal sistema Marconi e gli step per arrivare ad un dataset utile per i modelli.
- Nel capitolo 5 vi è lo studio sulla trasferibilità e la generalizzazione del modello
- Il capitolo 6 è dedicato allo studio delle risposte del modello attraverso l'Explainability.
- Il capitolo 7 è dedicato alle considerazioni sui risultati ottenuti in questo lavoro e alcuni sviluppi futuri.

Capitolo 1

# I sistemi HPC

## 1.1 Introduzione

Con HPC (High Performance Computing) si intende un insieme di infrastrutture e tecnologie utilizzate per l'elaborazione di grandi quantità di dati. I sistemi HPC sono in grado di fornire prestazione dell'ordine di **PetaFLOPS** utilizzando tecniche di cluster computing o di massive parallel processing. Le maggiori applicazioni sono: ricerca scientifica, progettazione tecnica, simulazione di realtà complesse, analisi su grandi quantità di dati, machine learning [1].

## 1.2 Architettura

I sistemi HPC sono generalmente costruiti da molti singoli computer collegati tra loro da un tipo di rete (ad esempio attraverso un ethernet). Ognuno di questi singoli computer viene chiamato **nodo**. In figura 1 si osserva la struttura tipica di un nodo. A seconda del sistema HPC, i nodi di calcolo, anche singolarmente, potrebbero essere molto più potenti di un normale personal computer. Spesso hanno più processori (ognuno è multi-core) e possono avere acceleratori (come le GPU) e altre funzionalità meno comuni sui personal computer. I sistemi HPC includono spesso diversi tipi di nodi, specializzati per scopi diversi. I nodi ***head*** (o front-end o login) sono i punti di accesso

per interagire con il sistema HPC. I nodi di calcolo sono dove viene eseguita la vera elaborazione. Generalmente non si ha accesso diretto ai nodi di calcolo. Infatti l'accesso a queste risorse è controllato da uno **scheduler** o da un **sistema batch.**

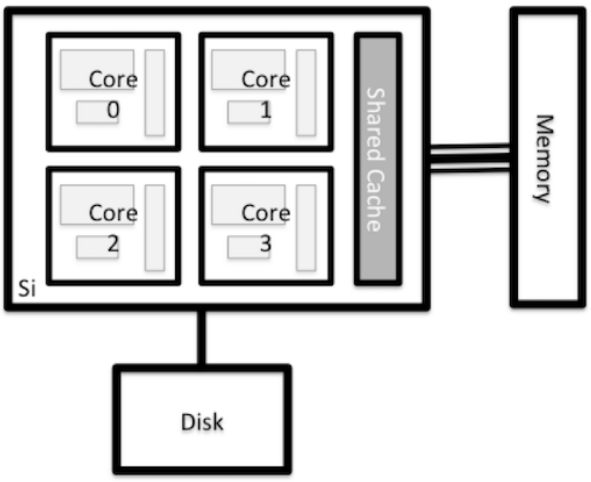


Figura 1.1: Esempio di struttura di un nodo [2]

Il processore contiene più core di calcolo (generalmente abbreviati in core). Ogni core contiene un'unità a virgola mobile (FPU) che è responsabile dell'esecuzione effettiva dei calcoli sui dati e varie cache di memoria veloci che sono responsabili della conservazione dei dati su cui si sta attualmente lavorando. La potenza di calcolo di un processore dipende generalmente da tre cose:

- La velocità del processore.
- La potenza dell'unità a virgola mobile (generalmente più moderno è il processore, più potente è la Floating-point unit).
- Il numero di core disponibili.

Spesso, i nodi HPC hanno processori multipli, quindi il numero di core disponibili su un nodo è raddoppiato. Ogni nodo ha anche una certa quantità di memoria disponibile (nota anche come RAM o DRAM) in aggiunta alle

cache di memoria del processore. I nodi di calcolo moderni hanno in genere un intervallo di memoria compreso tra 64 e 256 GB per nodo. Infine, ogni nodo ha anche accesso al *file-system* per l'archiviazione persistente dei dati. Questa memoria è spesso condivisa tra tutti i nodi e spesso ci sono diversi tipi di archiviazione collegati a un nodo [2].
Una struttura cosí complessa richiede numerosi sensori per avere un monitoraggio in tempo reale dei vari componenti. Tipicamente, i sensori misurano parametri come: temperatura core, temperatura CPU, sistema di raffreddamento (velocità delle ventole della CPU o del sistema di aerazione in generale), utilizzo della memoria RAM o di quella persistente, carichi di lavoro e molti altri.

## 1.3 Il sistema MARCONI

Il sistema Marconi HPC è il più grande supercomputer di proprietà della CINECA consorzio. Al momento è formato da quasi 6.800 nodi di calcolo divisi in due partizioni denominate *KnightLandings* e *SkyLake*. The *KnightLandings* la partizione possiede nodi con un singolo processore a 68 core ciascuno, mentre lo *Skylake* la partizione possiede nodi con due processori a 24 core ciascuno. Il calcolo globale è stimato in 20 PFlop/s. Perciò il MARCONI si colloca come **diciannovesimo supercomputer al mondo per potenza computazionale** [3].

Capitolo 2

# Anomaly Detection

## 2.1 Introduzione

L'anomaly detection è un importante problema studiato in diverse aree di ricerca e domini applicativi. L'obiettivo è trovare **pattern** nei dati che non sono conformi al comportamento previsto. Questi modelli non conformi sono spesso indicati come **anomalie**. In figura 2.1 vi è illustrato un esempio di anomalie in un semplice set di dati bidimensionali: i punti $o_1$, $o_2$ e i punti della regione $O_3$ sono anomalie poichè non hanno lo stesso comportamento dei restanti punti, concentrati nelle due regioni $N_1$ e $N_2$[4].
Il loro rilevamento trova ampio impiego in una vasta gamma di applicazioni come il rilevamento di frodi bancarie, assicurazioni o cure sanitarie, il rilevamento di intrusioni per la *cybersecurity*, il rilevamento di guasti in sistemi critici di sicurezza e nella sorveglianza militare. Nel tempo, sono state sviluppate una varietà di tecniche di rilevazione di anomalie e molte di queste tecniche sono state sviluppate specificamente per alcuni domini applicativi, mentre altre sono più generiche. In generale, le tecniche di rilevamento delle anomalie sono classificate in base ai seguenti criteri [5][6][7]:

- **Tipo di anomalia**: anomalia puntuale, anomalia contestuale e anomalia collettiva.

- **Disponibilità di etichette**: supervisionate, non supervisionate e semi-supervisionate.
- **Tipo di modello impiegato**: modelli lineari, modelli statistici, modelli probabilistici, basati sul clustering, basato sulla densità e basato sul deep learning e molti altri.
- **Applicazioni**: rilevamento di frodi, sorveglianza, rilevamento di danni industriali, anomalie mediche, rilevamento delle intrusionie e molti altri.

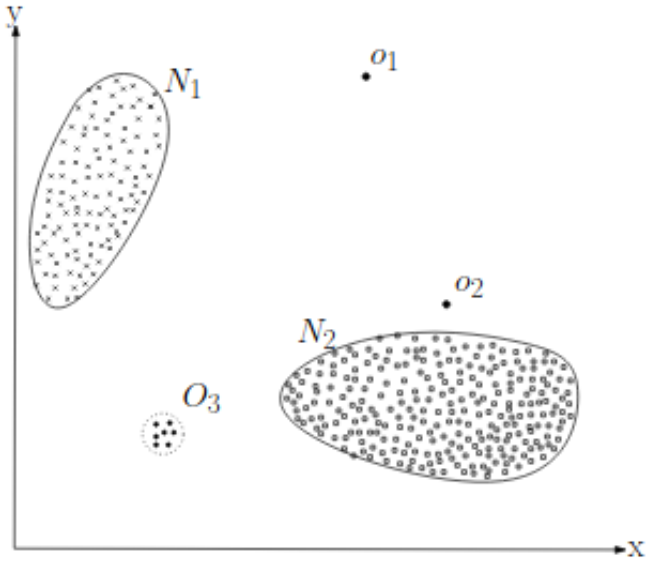


Figura 2.1: Esempio di anomalie: $o_1$, $o_2$, $O_3$ sono anomalie rispetto ai due pattern $N_1$ e $N_2$ [4].

## 2.2 Anomaly Detection negli HPC

Il dominio applicativo dell'anomaly detection di questo lavoro sono gli HPC. Come abbiamo già sottolineato nel capitolo 1, i sistemi di calcolo ad alte prestazioni (HPC) sono complesse macchine con molti componenti che devono funzionare contemporaneamente al meglio delle loro prestazioni teoriche.

Tuttavia molti fattori possono degradare le prestazioni di un sistema HPC: l'hardware può rompersi, le applicazioni potrebbero avere stati imprevisti, i componenti possono essere erroneamente configurati. Un aspetto critico del supercomputer moderno e futuro è la capacità di rilevare lo stato delle condizioni difettose dal comportamento improprio di una o più parti. Al giorno d'oggi, le infrastrutture di monitoraggio sono disponibili in molti sistemi e data center HPC e vengono utilizzati per raccogliere dati grazie allo stato dei sistemi e dei loro componenti e ad una grande varietà di sensori di misura. Dato l'enorme quantità di dati provenienti da un framework di monitoraggio, l'identificazione in tempo reale di problemi e situazioni indesiderate è un compito arduo per gli amministratori di sistema. La scala crescente dei sistemi HPC renderà questo compito ancora più difficile [8].
Perciò vi è il crescente bisogno di utilizzare nuove tecniche capaci di prevedere comportamenti anomali in grado di aiutare gli amministratori di sistema in un intervento rapido e preventivo in modo da poter ridurre al minimo i danni.

## 2.3 Tecniche di Anomaly Detection

Sono state sviluppate diverse tecniche per affrontare il problema. Tuttavia, negli ultimi anni si seguono principalmente due approcci: tecniche statistiche e tecniche basate sul Machine Learning.

### 2.3.1 Modelli Statistici

Il principio alla base di qualsiasi tecnica statistica di anomaly detection è: "*L'anomalia è un'osservazione sospettata di essere parzialmente o totalmente irrilevante perché non generata dal modello stocastico ipotizzato*" [Anscombe e Guttman 1960]. Le tecniche statistiche di rilevamento delle anomalie si basano sul seguente presupposto chiave: *le istanze di dati normali si verificano in regioni ad alta probabilità di un modello stocastico, mentre si verificano anomalie nelle regioni a bassa probabilità del modello stocastico.* Le tecniche statistiche adattano un modello statistico (di solito per un comportamento

normale) ai dati forniti e quindi applicano un test di inferenza statistica per determinare se un'instanza, ancora non vista, appartiene a questo modello o meno. Le istanze che hanno una bassa probabilità di essere generate dal modello appreso, basate sulla statistica del test applicato, sono dichiarate anomalie. Tra le tecniche statistiche ci sono due grandi famiglie: tecniche parametriche e non parametriche.
Le tecniche **parametriche** presuppongono che i dati normali siano generati da una distribuzione parametrica con parametri $\Theta$ e funzione di densità di probabilità $f(x, \Theta)$, in cui si tratta di un'osservazione. Il punteggio di anomalia di un'istanza di prova (o osservazione) è l'inverso della funzione di densità di probabilità, $f(x, \Theta)$. I parametri $\Theta$ sono stimati dai dati forniti.
Le tecniche **non parametriche** utilizzano modelli statistici non parametrici, in modo tale che la struttura del modello non sia definita a priori, ma sia invece determinata da dati specifici. Tali tecniche generalmente fanno meno ipotesi riguardo ai dati, come la fluidità della densità, rispetto alle tecniche parametriche [4].

### 2.3.2 Modelli di Machine Learning

Il termine Machine Learning (apprendimento automatico) si riferisce al rilevamento automatico di modelli significativi nei dati. Negli ultimi due decenni è diventato uno strumento comune in quasi tutte le attività che richiedono l'estrazione di informazioni da grandi insiemi di dati [9]. Con l'avvento dei Big Data, gli HPC moderni si sono dotati di strumenti tecnologicamente avanzati per la raccolta dati sul loro funzionamento. Il Machine Learning, quindi, va incontro all'esigenza di sfruttare questi grandi quantitativi di dati per lo sviluppo di modelli predittivi per l'anomaly detection. L'apprendimento è un dominio molto ampio. Di conseguenza, il campo del machine learning si è ramificato in diversi sottocampi che si occupano di diversi tipi di attività di apprendimento [9]. In figura 2.2 sono raffigurate le differenti tecniche di Machine Learning e i tipi di dati richiesti nelle loro applicazioni. Nel prossimo capitolo sono introdotte le due più importanti tecniche, l'**apprendimento supervisionato** (Supervised Learning) e

**non supervisionato** (Unsupervised Learning), per poi introdurre la tecnica dell'**apprendimento semi-supervisionato** (semi-supervised learning) e dell'**apprendimento per rinforzo**(reinforcement learning).

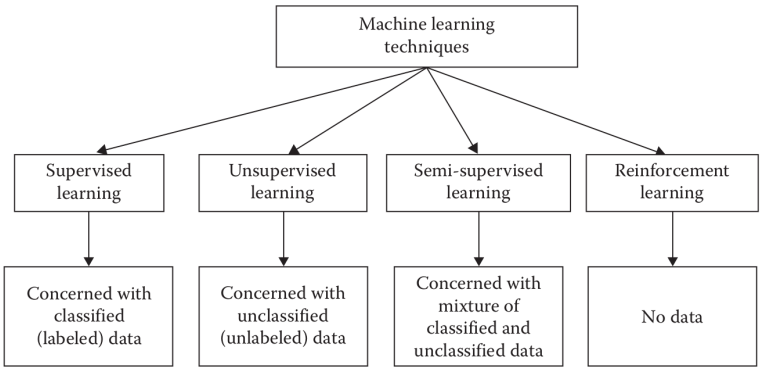


Figura 2.2: Le differenti tecniche di Machine Learning e i tipi di dati richiesti [10].

Capitolo 3

# Machine Learning

Il termine Machine Learning (Apprendimento automatico) fu coniato nel 1959 da Arthur Samuel [11]. Tom M. Mitchell ha fornito una definizione più citata e più formale degli algoritmi studiati nel campo dell'apprendimento automatico: "*Si dice che un programma per computer apprenda dall'esperienza E rispetto ad alcune classi di attività T e che misurino le prestazioni P se le sue prestazioni in attività in T, come misurato da P, migliora con l'esperienza E.*" [12]. Questa definizione dei compiti in cui l'apprendimento automatico è interessato, offre una definizione fondamentalmente operativa piuttosto che definire il campo in termini cognitivi. Ciò segue la proposta di Alan Turing nel suo documento "*Computing Machinery and Intelligence*", in cui la domanda "*Le macchine possono pensare*?" è sostituito dalla domanda "*Le macchine possono fare ciò che noi (come entità pensanti) possiamo fare?*" [13]. Nella proposta di Turing sono esposte le varie caratteristiche che potrebbero essere possedute da una macchina pensante e le varie implicazioni nel costruirne una. Oggi l'apprendimento automatico è lo studio scientifico di algoritmi e modelli statistici che i sistemi informatici utilizzano per eseguire un compito specifico senza utilizzare istruzioni esplicite, basandosi invece su schemi e inferenze. È visto come un sottoinsieme dell'**intelligenza artificiale**. Gli algoritmi di machine learning costruiscono un modello matematico basato su dati di esempio, noti come "**dati di training**", al fine di effettuare previsioni o decisioni senza essere esplicitamente programmati

per eseguire l'attività. Gli algoritmi di machine learning vengono utilizzati in un ampia varietà di applicazioni, in cui è difficile o impossibile sviluppare un algoritmo convenzionale per eseguire efficacemente l'attività. L'apprendimento automatico si divide in due grandi categorie: l'**apprendimento supervisionato** (*Supervised Learning*) e l'**apprendimento non supervisionato**(*Unsupervised Learning*). Un altro campo che negli ultimi anni sta dando un enorme contributo è l'**apprendimento per rinforzo** (Reinforcement Learning) [14]. Nelle prossime sezioni saranno spiegati i maggiori algoritmi utilizzati in questi campi per poi esporre l'algoritmo utilizzato in questo lavoro .

## 3.1 Supervised Learning

Nell'apprendimento supervisionato, l'obiettivo è inferire una funzione o mappatura dai dati di allenamento (***training data***) etichettati. I dati di allenamento sono costituiti dal vettore di input $X$ e dal vettore di output $Y$ di etichette o tag. Un'etichetta o tag dal vettore $Y$ è la spiegazione del suo rispettivo esempio di input dal vettore $X$. Insieme formano un ***training example***. In altre parole, i dati di addestramento comprendono i *training examples*. Il il vettore di output $Y$ è costituito da etichette per ciascun *training examples* presente nel *training set*. Per la realizzazione di un modello di machine learning, in generale, viene utilizzato un *training set* opportunamente preso dal dataset su cui costruire il modello. Per provare il risultato di un modello e le sue performance, si utilizza un altro sottoinsieme di dati dal dataset chiamato *test set*. *Training set* e *test set* sono quindi sottoinsiemi dei dati iniziali.
Le etichette per il vettore di output sono forniti dal supervisore. Spesso, questi supervisori sono umani, esperti del dominio su cui si sta costruendo il modello, ma possono essere utilizzati anche metodi automatici per tale etichettatura. I dati etichettati manualmente sono un preziosa risorsa intelligente e affidabile per l'apprendimento supervisionato [10]. Due gruppi o categorie di algoritmi rientrano nell'apprendimento supervisionato: la **classificazione** e la **regressione**. Nella classificazione l'obiettivo è quello di

costruire un modello conciso della distribuzione delle etichette di **classe** in termini di *features* predittive. Il classificatore risultante viene quindi utilizzato per assegnare le etichette di classe alle istanze di test in cui sono noti i valori delle *features* predittive, ma il valore dell'etichetta di classe è sconosciuto. In figura 2.3 è illustrato un esempio di training set per un task di classificazione. La ***class*** è l'obiettivo di un algoritmo supervisionato per la classificazione che apprende in un primo momento grazie al *training set* [15]. La regressione è un processo statistico che cerca di stabilire una relazione tra due o più variabili. Fornito a un modello di regressione un valore $x$, questo restituirà il corrispondente valore $y$ generato dall'elaborazione di $x$. La regressione lineare si differenzia nettamente dalla classificazione, poiché quest'ultima si limita a discriminare gli elementi in una determinato numero di classi (label), mentre nella prima l'input è un dato e il sistema ci restituisce un output reale (a differenza della classificazione che riceve in input un dato e restituisce in ouput una label, un'etichetta a cui il dato appartiene e in generale quindi un valore discreto) [16]. Nelle prossime sezioni sono descritti i più importanti algoritmi per la classificazione: decision tree (alberi decisionali), support vector machines (SVM), classificatori Naive Bayes e le reti neurali artificiali.

| Data in standard format | | | | | |
|---|---|---|---|---|---|
| case | Feature 1 | Feature 2 | ... | Feature n | Class |
| 1 | xxx | x | | xx | good |
| 2 | xxx | x | | xx | good |
| 3 | xxx | x | | xx | bad |
| ... | | | | | ... |

Figura 3.1: Esempio di dati per la classificazione: la colonna "Class" rappresenta l'etichetta assegnata alle feature [15].

### 3.1.1 Decision Tree

Gli alberi decisionali (Decision Tree) sono alberi che classificano le istanze ordinandole in base ai valori delle ***features***. Ogni nodo in un albero de-

cisionale rappresenta una ***feature*** in un'istanza da classificare e ogni ramo rappresenta un valore che il nodo può assumere. Le istanze sono classificate a partire dal nodo principale e ordinate in base ai loro valori delle features (figura 2.4) [15].

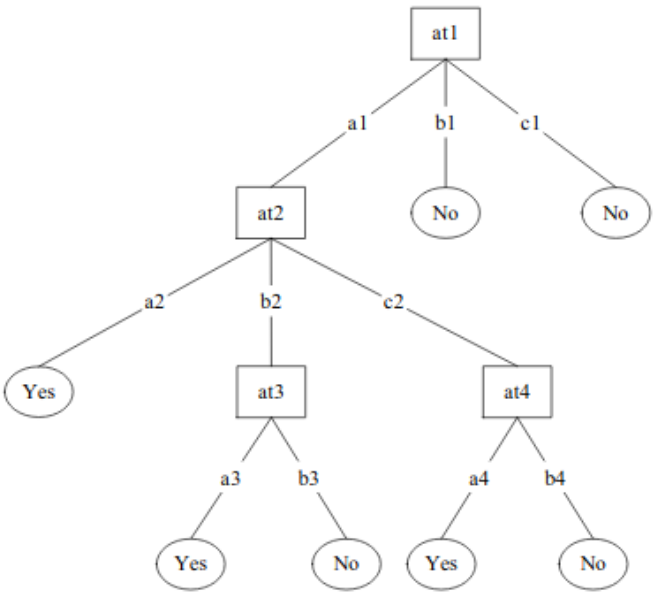


Figura 3.2: Esempio di un albero di decisione [15].

### 3.1.2 Support Vector Machines

Il support vector machines (SVM) è un algoritmo che ruota attorno alla nozione di "margine", ai lati di un iperpiano che separa due classi di dati. È stato dimostrato che l'ottimizzazione del margine e la creazione della massima distanza possibile tra l'**iperpiano** di separazione e le istanze su entrambi i lati, riducono il limite superiore dell'errore di generalizzazione previsto. Se è possibile separare linearmente due classi, è possibile trovare un iperpiano di separazione ottimale minimizzando la norma quadrata dell'iperpiano di separazione. Nel caso di dati separabili linearmente, una volta trovato l'iperpiano di separazione ottimale, i punti dati che giacciono sul suo margine sono noti come **punti vettore di supporto** e la soluzione è rappresentata

come una combinazione lineare di solo questi punti (figura 2.3). Altri punti dati vengono ignorati [15].

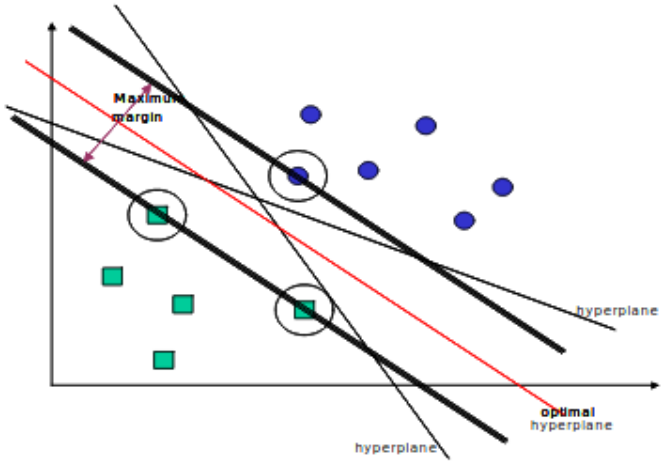


Figura 3.3: L'algoritmo SVM: in verde e in blu i dati linearmente separabili, i punti cerchiati sono i vettori di supporto [15].

### 3.1.3 Classificatori Naive Bayes

Il classificatore Naive Bayes è uno dei più rappresentativi algoritmi di ***statistical learning***. Le Naive Bayes network (NB) sono reti bayesiane molto semplici che sono composte da grafici aciclici diretti con un solo genitore (che rappresenta il nodo non osservato) e diversi nodi figli (corrispondenti a nodi osservati) con una forte assunzione di indipendenza tra i nodi figli nel contesto del loro genitore. Il modello di indipendenza (Naive Bayes) si basa sulla stima:

$$R = \frac{P(i|X)}{P(j|X)} = \frac{P(i)P(X|i)}{P(j)P(X|j)} = \frac{P(i)\prod P(X_r|i)}{P(j)\prod P(X_r|j)} \tag{3.1}$$

Confrontando queste due probabilità, la probabilità maggiore indica il valore dell'etichetta di classe più probabile che sia l'etichetta effettiva (se $R > 1$: predice $i$ altrimenti predice $j$). Poiché questo algoritmo di classificazione utilizza una moltiplicazione per calcolare le probabilità $P(X, i)$, è particolar-

mente incline a essere influenzato da una probabilità 0.
L'ipotesi di indipendenza tra i nodi figli è chiaramente quasi sempre sbagliata e per questo motivo i classificatori Bayesiani sono di solito meno accurati rispetto ad altri algoritmi di apprendimento [15].

### 3.1.4 Reti Neurali Artificiali

Una rete neurale artificiale (ANN) è un modello di calcolo ispirato alla struttura delle **reti neurali** nel cervello. L'idea alla base delle reti neurali artificiali è che molti neuroni possono essere uniti tramite collegamenti di comunicazione per eseguire calcoli complessi. È comune descrivere la struttura di una rete neurale come un **grafo** i cui nodi sono i neuroni e ogni arco (diretto) nel grafo collega l'output di alcuni neuroni all'ingresso di un altro neurone. Si prende come esempio una rete ***feed-farward*** per una descrizione generale delle ANN: essa è descritta da un grafo aciclico diretto, $G = (V, E)$ e una funzione di peso sugli archi, $w : E \to R$. I nodi del grafo corrispondono ai neuroni. Ogni singolo neurone è modellato come una semplice funzione scalare, $\sigma : R \to R$. Ci concentreremo su tre possibili funzioni $\sigma$ : la funzione sign, $\sigma(a) = sign(a)$, la funzione di soglia, $\sigma(a) = 1$ se $a > 0$ e la funzione sigmoid, $\sigma(a) = \frac{1}{(1+e^{-a})}$, che è un'approssimazione uniforme alla funzione di soglia. Chiamiamo queste funzioni, le **funzioni di attivazione** del neurone. Ciascun arco del grafo collega l'uscita di alcuni neuroni all'ingresso di un altro neurone. L'input di un neurone si ottiene prendendo una somma ponderata delle uscite di tutti i neuroni ad esso collegati [10]. Una rete è solitamente divisa in **strati** (***layer***) in cui ci sono un numero prefissato di neuroni. La struttura è, in generale, uno strato di input, un numero predefinito di strati interni (***hidden layer***) e uno strato di output (figura 2.5). Per questo motivo si parla di apprendimento profondo (**deep learning**).

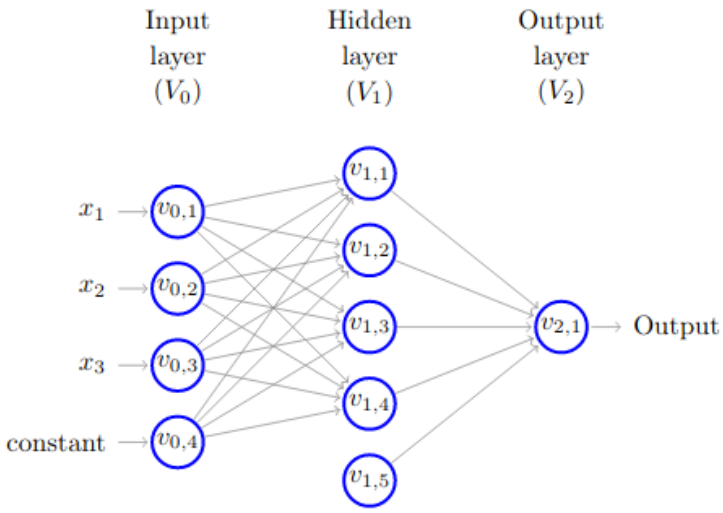


Figura 3.4: Esempio di una ANN feed-forward: $V_0$,$V_1$,$V_2$ sono i tre strati della rete, $v_{i,j}$ rappresenta il neurone $j$-esimo dello strato $i$ e $x_n$ i valori d'ingresso all'input layer [10].

L'apprendimento di una ANN si divide in due parti: una **fase di training** e una di **inferenza**. Nella fase di training la rete è addestrata su una serie di dati per determinare la mappatura input-output. I pesi delle connessioni tra i neuroni vengono quindi fissati e la rete viene utilizzata per determinare le classificazioni di un nuovo set di dati. Durante la classificazione, il segnale sulle unità di input si propaga completamente attraverso la rete per determinare i valori di attivazione in tutte le unità di output. Pertanto, la ANN dipende da tre aspetti fondamentali: funzioni di input e attivazione dell'unità, architettura di rete e peso di ciascuna connessione di input. Dato che i primi due aspetti sono fissi, il comportamento della ANN è definito dagli attuali valori dei pesi. I pesi della rete da addestrare vengono inizialmente impostati su valori casuali, quindi le istanze del set di allenamento vengono ripetutamente esposte alla rete. I valori per l'input di un'istanza vengono posizionati sulle unità di input e l'output della rete viene confrontato con l'output desiderato per questa istanza utilizzando una **funzione errore**. Quindi, tutti i pesi nella rete vengono regolati leggermente nella direzione

che avvicinerebbe i valori di uscita della rete ai valori per l'uscita desiderata. Esistono diversi algoritmi con cui è possibile addestrare una rete. Tuttavia, l'algoritmo di apprendimento più noto e ampiamente utilizzato per stimare i valori dei pesi è l'algoritmo di ***Back Propagation*** (BP). Generalmente, l'algoritmo BP include i seguenti sei passaggi [15]:

- Presentare un campione (batch) del training set alla rete neurale.
- Confrontare l'output della rete con l'output desiderato da quel campione. Calcola l'errore in ciascun neurone di output.
- Per ciascun neurone, calcolare 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 "responabilità" per l'errore locale ai neuroni al livello precedente, dando maggiore responsabilità ai neuroni collegati da pesi più forti.
- Ripeti i passaggi precedenti sui neuroni al livello precedente, usando la "responsabilità" di ognuno come errore.

L'algoritmo BP dovrà eseguire una serie di modifiche dei pesi prima di raggiungere una buona configurazione. Nella fase di inferenza, invece, la rete addestreata viene utilizzata per la classificazioni di nuove instanze dei dati, presentando ad essa un nuovo set di dati.
I parametri di una rete come il numero di neuroni, numero di strati, funzione di errore, dimensione del batch e la funzione di attivazioni sono scelti dal progettista della rete. Esistono diversi tipi di ANN oltre alle feed-forward descritte in questa sezione. Tuttavia, esse seguono tutte lo stesso approccio. Tra le reti ANN troviamo:

- Le **reti neurali convoluzionali** che sono un tipo di rete specializzata per l'elaborazione di dati multidimensionali (***grid data***). Esse utiliz-

zano la **convoluzione** con una matrice chiamata kernel al posto della matrice generale moltiplicativa in almeno uno dei loro strati [17].

- Le ***reti neurali ricorrenti*** che sono una classe di rete neurale artificiale in cui i valori di uscita di uno strato di un livello superiore vengono utilizzati come ingresso ad uno strato di livello inferiore [18]. Quest'interconnessione tra strati permette l'utilizzo di uno degli strati come **memoria di stato**, e consente, fornendo in ingresso una sequenza temporale di valori, di modellarne un comportamento dinamico temporale dipendente dalle informazioni ricevute agli istanti di tempo precedenti [19].

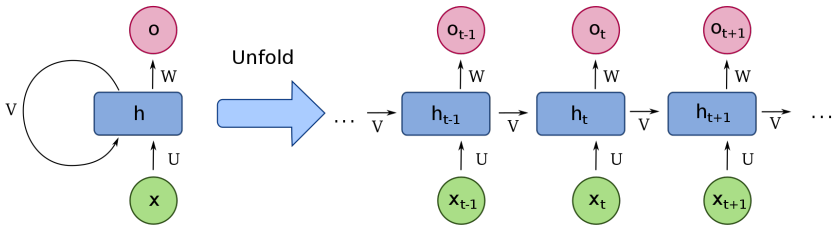


Figura 3.5: Architettura di una rete neurale ricorrente [20].

Tra le reti neurali artificiali troviamo anche gli **autoencoder** per l'apprendimento non supervisionato, ma ne parlaremo in una sezione apposita, in modo approfondito, essendo una delle tecniche utilizzate in questo lavoro.

### 3.1.5 Supervised Learning per l'Anomaly Detection

L'apprendimento supervisionato nell'anomaly detection permette di realizzare sistemi efficaci ed efficenti. Tuttavia, un'anomalia è un evento molto raro. Pertanto, avere a disposizione set di dati con un'etichetta che distingue punti normali e punti anomali è tutt'ora un problema sia per la realizzazione di sistemi capaci di individuare un comportamento anomalo ed etichettarlo sia per la natura imprevedibile di un'anomalia. Infatti per garantire un buon

processo di apprendimento, l'insieme dei dati di addestramento dovrebbe essere imparziale ed equilibrato e deve contenere più o meno lo stesso numero di esempi per ogni classe. Questi requisiti complicano notevolmente l'attività di training. Nel nostro caso, gli HPC, le condizioni di guasto non vengono sempre archiviati in registri o database, ma il solo compito di assegnare etichette a un set di dati è di responsabilità degli amministratori di sistema. Pertanto, non è sempre possibile disporre dei set di dati con etichetta corretti necessario con approcci supervisionati [21].
Tuttavia, nel nostro lavoro, grazie ai sistemi avanzati di raccolta dati negli HPC del MARCONI, abbiamo a disposizione set di dati con un'etichetta indicante uno stato anomalo. Ciò ci permette, quindi, di seguire la strada dell'apprendimento supervisionato.

| timestamp | value | name | node | state |
|---|---|---|---|---|
| 2019-10-08 09:00:00 | DOWN+DRAIN matches a critical state | plugin_output | r183c12s04 | 2 |
| 2019-10-08 09:15:00 | DOWN+DRAIN matches a critical state | plugin_output | r183c12s04 | 2 |
| 2019-10-31 13:45:00 | ALLOCATED+DRAIN matches a critical state | plugin_output | r183c12s04 | 2 |
| 2019-11-01 21:07:00 | ERROR, the command timeouted: /sbin/runuser -c '/usr/bin/ssh master01 /cinecalocal/nagios/passive/mhsc_nodehealth_a3.pl' root | plugin_output | r183c12s04 | 2 |
| 2019-11-02 17:45:00 | C({topwaiter: monitored, PCacheMsgMgrThread: SINCE 603.7923 sec.}[/marconi,/marconi_work]) W() O() | plugin_output | r183c12s04 | 2 |

Figura 3.6: Esempio del nostro set di dati: la colonna *state* ci indica se c'è un'anomalia o comportamento normale [22].

## 3.2 Unsupervised Learning

L'apprendimento non supervisionato studia come i sistemi possono imparare a rappresentare particolari modelli di input in un modo che rifletta la struttura statistica della raccolta complessiva di pattern di input. Diversamente dall'apprendimento supervisionato o dall'apprendimento rafforzato, **non vi sono output target espliciti o valutazioni associate a ciascun input** [23]. Tra gli algoritmi non supervisionati vi sono gli tecniche di clustering, tecniche statistiche e le reti neurali artificiali.

### 3.2.1 Tecniche di Clustering: k-Means, Hierarchical e DBSCAN

I metodi di clustering prevedono la decomposizione diretta di un set di dati in una partizione composta da $k$ **cluster** disgiunti, indicata con $C = C_1, ..., C_k$. Questi metodi generalmente cercano di produrre un'approssimazione locale a un globale funzione obiettiva, che viene identificata perfezionando iterativamente una soluzione iniziale. Il k-means è l'algoritmo di clustering partizionale più utilizzato. Impiega uno schema di trasferimento iterativo per produrre un clustering k-vie che minimizza localmente la distorsione tra gli oggetti dati e un set di $k$ rappresentanti dei cluster. Ogni rappresentante, indicato come centroide, è calcolato come vettore medio di tutti gli oggetti assegnati a un determinato cluster. Nella versione classica dell'algoritmo, la distorsione viene misurata usando la distanza euclidea, in modo che l'obiettivo del processo di clustering diventi la minimizzazione dell'errore di somma dei quadrati (SSE) tra gli oggetti ($x_i$) e i centroidi del cluster $\mu_1, ..., \mu_k$:

$$SSE(C) = \sum_{c=1}^{k} \sum_{x_i \in C_c} \|x_i - \mu_c\| \tag{3.2}$$

dove:

$$\mu_c = \frac{\sum_{x_i \in C_c} X_i}{|C_c|} \tag{3.3}$$

Tuttavia, l'algoritmo si divide in due fasi. Nel primo passaggio, ogni oggetto viene riassegnato al cluster con il più vicino centroide. Una volta elaborati tutti gli oggetti, i vettori del centroide vengono aggiornati per riflettere le nuove assegnazioni di cluster. Il processo di affinamento iterativo è ripetuto fino a quando non viene soddisfatto un determinato criterio di stop. In genere questo si verifica quando l'assegnazione di oggetti ai cluster non cambia più da un'iterazione ad un'altra.
Invece di generare una partizione piatta di dati, può spesso essere utile strutturare una gerarchia di concetti producendo un insieme di cluster nidificati che potrebbero essere disposti a formare una struttura ad albero attraverso gli algoritmi gerarchici. Essi sono generalmente organizzati in due categorie

distinte:

- **Agglomerativo**: iniziare con ciascun oggetto assegnato a un cluster singleton. Viene applicata una strategia dal basso verso l'alto in cui, ad ogni passo, la coppia di cluster più simile è unita.
- **Divisive**: inizia con un singolo cluster contenente tutti gli $n$ oggetti. Applica un strategia verso il basso in cui, ad ogni passaggio, un cluster scelto viene suddiviso in due sotto cluster.

In entrambi i casi, la gerarchia risultante può essere presentata visivamente usando una struttura ad albero chiamata **dendrogramma**, che contiene nodi per ciascuno cluster costruito dall'algoritmo, insieme a cluster-relazione che illustrano le operazioni di unione o divisione eseguite durante il processo di clustering. La figura 2.8 fornisce un semplice esempio di un processo agglomerativo di clustering, applicato a un set di cinque oggetti dati, insieme a le corrispondenti assegnazioni di cluster. Vale la pena notare che ad ogni fusione la somiglianza tra la coppia scelta di cluster diminuisce. A differenza del requisito, nella maggior parte degli algoritmi partizionali, di specificare un valore per il numero di cluster $k$ in anticipo, gli algoritmi gerarchici supportano la costruzione di un albero con il quale un utente può selezionare manualmente il valore $k$ esaminando il dendrogramma risultante e individuando un punto di interruzione appropriato [24].

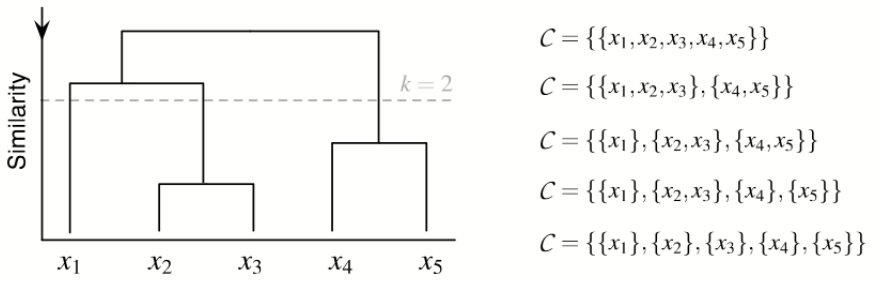


Figura 3.7: Esempio di algoritmo di clustering gerarchico agglomertivo [24].

Gli algoritmi di clustering basati sulla densità (DBSCAN) sono progettati per rilevare cluster di forma arbitraria. Un cluster viene definito come una

regione ad alta densità partizionata da regioni a bassa densità nello spazio dei dati. DBSCAN può scoprire cluster di forma arbitraria ma è sensibile ai parametri di input, specialmente quando la densità dei dati non è uniforme. Un tipico algoritmo **DBSCAN** richiede due parametri: $\epsilon$ e il numero minimo di punti richiesti per formare un cluster. Inizia con un punto di partenza arbitrario che non è stato visitato. Questo punto di $\epsilon$-vicinato viene recuperato e, se contiene abbastanza punti, viene avviato un cluster, altrimenti il punto è etichettato come **rumore**. Si noti che questo punto potrebbe essere successivamente trovato in un $\epsilon$-ambiente sufficientemente dimensionato di un punto diverso e quindi essere inserito in un cluster. Se si trova che un punto è una parte densa di un cluster, anche il suo $\epsilon$-vicinato fa parte di quel cluster . Quindi, vengono aggiunti tutti i punti che si trovano all'interno del $\epsilon$-vicinato, così come il loro stesso vicinato quando sono anche densi. Questo processo continua fino a quando non viene trovato completamente il cluster collegato alla densità. Quindi, un nuovo punto non visitato viene recuperato ed elaborato, portando alla scoperta di un ulteriore cluster o rumore [25].
I vantaggi del DBSCAN sono i seguenti [25]:

- Il DBSCAN non richiede di specificare a priori il numero di cluster nei dati, a differenza di $k$-means.
- Il DBSCAN cerca cluster di forma arbitraria.
- È robusto al rumore dei dati.
- Il DBSCAN richiede due parametri ed è per lo più insensibile all'ordinamento dei punti nel dataset.

Per quanto riguarda gli svantaggi, il DBSCAN dipende dalla misura della distanza utilizzata nella funzione Query $(P, )$. La metrica di distanza più comune utilizzata è la **distanza euclidea**. Soprattutto per i dati ad alta dimensione, questa metrica può essere resa quasi inutile a causa della cosiddetta "cura della dimensionalità", rendendo difficile trovare un valore adeguato per . Questo effetto, tuttavia, è presente anche in qualsiasi altro algoritmo basato sulla distanza euclidea. Inoltre l'algoritmo non può raggruppare bene i set di

dati con grandi differenze di densità, poiché la combinazione dei punti minini (all'interno di un cluster) non può quindi essere scelta in modo appropriato per tutti i cluster [25].

### 3.2.2 Tecniche Statistiche

È spesso conveniente ed efficace presumere che i dati siano stati generati a seguito di un processo statistico e quindi descrivere i dati trovando il modello statistico che si adatta meglio ai dati, in cui il modello è descritto in termini di distribuzione e un insieme di parametri per quella **distribuzione**. Ad alto livello, questo processo prevede la decisione su un modello statistico per i dati e la stima i parametri di quel modello dai dati. Queste tecniche si basano su modelli di ***Mixtures***, che modellano i dati utilizzando una serie di distribuzioni statistiche. Ogni distribuzione corrisponde a un cluster e i parametri di ciascuna distribuzione forniscono una descrizione di cluster corrispondente, in genere in termini di centro e diffusione. Tra le tecniche statistiche troviamo: il ***maximum likelihood estimation*** (MLE) e l'algoritmo ***Expectation-Maximization*** (EM) [26].
I modelli "*Mixtures*" visualizzano i dati come un insieme di osservazioni da una mix di distribuzioni di probabilità differenti. Le distribuzioni di probabilità possono essere qualsiasi cosa, ma sono spesso considerati normali multivariati, poiché questo tipo di distribuzione è ben compreso, matematicamente facile da lavorare ed è stato mostrato per produrre buoni risultati in molti casi. Concettualmente, i modelli *Mixtures* corrispondono al seguente processo di generazione dei dati. Date diverse distribuzioni, di solito dello stesso tipo, ma con parametri diversi, selezionare casualmente una di queste distribuzioni e generare un oggetto da esso. Si Ripete il processo $m$ volte, dove $m$ è il numero di oggetti.
Utilizzando metodi statistici, possiamo stimare il parametro di queste distribuzioni dai dati e quindi descrivere queste distribuzioni (cluster). Possiamo anche identificare quali oggetti appartengono a quali cluster. Tuttavia, la modellazione mista non produce una chiara assegnazione di oggetti a cluster, ma piuttosto dà la probabilità a cui appartiene un oggetto specifico a un

particolare cluster.
Dato un modello statistico per i dati, è necessario stimare i parametri di quel modello. Un approccio standard utilizzato per questa attività è il ***maximum likelihood esimation*** (MLE).
Il metodo è chiamato maximum likelihood perché, dato un insieme di dati, la probabilità dei dati, considerata come una funzione del parametro, è chiamata funzione di ***likelihood***:

$$likelihood(\Theta, X) = L(\Theta, X) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{(x_i-\mu)^2}{2\sigma^2}} \tag{3.4}$$

per ragioni pratiche si utilizza il logaritmo della funzione:

$$\log(likelihood(\Theta, X)) = l(\Theta, X) = -\sum_{i=1}^{m} \frac{(x_i - \mu)^2}{2\sigma^2} - 0.5m \log 2\pi - m \log \sigma \tag{3.5}$$

dove $\mu$ e $\sigma$ sono i parametri della **distribuzione gaussiana** utilizzata come modello di mixture (***Univariate Gaussian Mixture***). Si nota che i valori dei parametri che massimizzano la probabilità della funzione $\log likelihood$ massimizzano anche la probabilità poiché il *log* è una funzione monotona crescente. Possiamo anche utilizzare l'approccio MLE per stimare il modello parametri per un modello di mixture. Nel caso più semplice, sappiamo quali dati provengono da quali distribuzioni e la situazione si riduce a stimare i parametri di una singola distribuzione data da quella distribuzione. In una situazione più generale e più realistica, non sappiamo quali punti sono stati generati da quale distribuzione. Pertanto, non possiamo direttamente calcolare la probabilità di ciascun punto dati e, quindi, sembrerebbe che non è possibile utilizzare il MLE per stimare i parametri. La soluzione a questo problema è l'algoritmo EM.
Data un'ipotesi per i valori dei parametri, l'algoritmo EM calcola la probabilità che ciascun punto appartenga a ciascuna distribuzione e quindi utilizza queste probabilità per calcolare una nuova stima per i parametri. (Questi parametri sono quelli che massimizzano la probabilità). Questa iterazione continua finché le stime dei parametri non cambiano o cambiano molto poco. Pertanto, utilizziamo la stima della massima probabilità, ma tramite un

ricerca iterativa [26]. L'algoritmo EM è fatto in questo modo [26]:

1. Seleziona un set iniziale di parametri del modello. (Come nel k-means, questo può essere fatto in modo casuale o in vari modi).

2. **Expection Step**: per ogni oggetto, calcolare la probabilità che ogni oggetto appartiene a ciascuna distribuzione, ovvero calcola $prob(distributionj|x_i, \Theta)$.

3. **Maximization Step**: date le probabilità dall'Expectation Step, trova le nuove stime dei parametri che massimizzano la probabilità prevista.

4. I parametri non cambiano. (In alternativa, interrompere se la modifica dei parametri è inferiore a un valore specificato di soglia).

Trovare cluster modellando i dati usando modelli di mixture e applicando l'algoritmo EM per stimare i parametri di questi modelli ha una varietà di vantaggi e svantaggi. Sul lato negativo, l'algoritmo EM può essere lento, non è pratico per i modelli con un gran numero di componenti e non funziona bene quando i cluster contengono solo pochi punti dati o se il i punti dati sono quasi lineari. Sul lato positivo, questi modelli sono più generici del k-means o altre tecniche di clustering. Pertanto, questi modelli (basati su distribuzioni gaussiane) possono trovare cluster di differenti dimensioni e forme ellittiche [26].

### 3.2.3 Le Reti Neurali Artificiali per l'apprendimento non supervisionato: gli Autoencoder

Gli autoencoder giocano un ruolo fondamentale per l'apprendimento non supervisionato e soprattutto nell'anomaly detection. Un autoencoder è **una rete neurale addestrata a tentare di copiare il suo input al suo output**. Internamente, ha uno strato nascosto $h$ chiamato ***encoder layer*** il quale descrive una codifica dell'input. La rete può essere vista come composta da due parti: un funzione **encoder** $h = f(x)$ e una **decoder** che produce una ricostruzione $r = g(h)$ (in figura 3.9 vi è illustrato un esempio di architettura

tipica di un autoencoder). Pertanto, gli autoencoder sono progettati per non essere in grado di imparare a copiare perfettamente. Di solito lo sono in modi che consentono loro di copiare solo approssimativamente e solo di copiare l'input che assomiglia ai dati di allenamento. Perché il modello è costretto a stabilire delle priorità su quali aspetti dell'input debbano essere copiati, questo fa si che spesso apprenda proprietà utili dei dati [17].

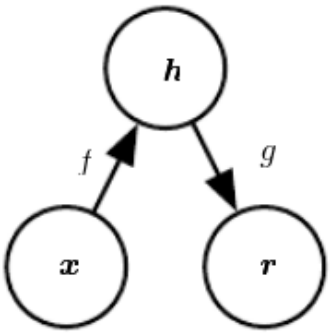


Figura 3.8: La struttura generale di un autoencoder, $f$ trasforma l'input $x$ nella codifica $h$ mentre la funzone $g$ prende la codifica per ricostruire il dato $r$ [17].

Esistono diversi tipi di autoencoder:

- Un autoencoder la cui dimensione del code layer è inferiore della dimensione dell'input viene chiamato **undercomplete**. L'obiettivo è acquisire le caratteristiche più salienti dei dati di allenamento [17].
- Un autoencoder, invece, che ha il code layer maggiore dell'input, è chiamato **overcomplete**.
- Uno **Sparse Autoencoder** è un autoencoder il cui criterio di addestramento prevede una penalità di sparsità $\Omega(h)$ sul layer code $h$ oltre all'errore di ricostruzione:

$$L(x, g(f(x))) + \Omega(h) \tag{3.6}$$

dove $g(h)$ è il decoder output e $h = f(x)$ l'input. Essi vengono in genere utilizzati per apprendere le funzionalità di un altro task, ad esempio come classificazione [17].

- Un **Denoising Autoencoder** permette di ricostruire dati da quelli corrotti, minimizzando la ***loss function*** $L(x, g(f(x')))$ dove $x'$ è nella stessa forma di $x$ corrotta, però, dal rumore [17].

- Un Varational Autoencoder (VAE) è un autoencoder composto sia da un **encoder** che da un **decoder** e che viene addestrato per ridurre al minimo l'errore di ricostruzione tra i dati codificati e decodificati e i dati iniziali. Tuttavia, al fine di introdurre una certa regolarizzazione dello spazio latente, procediamo a una leggera modifica del processo di codifica-decodifica: invece di codificare un input come un singolo punto, lo codifichiamo come distribuzione nello spazio latente (figura 2.10) [27]:

  1. l'input viene codificato come distribuzione nello spazio latente.
  2. un punto dallo spazio latente viene campionato da quella distribuzione.
  3. il punto campionato viene decodificato e l'errore di ricostruzione può essere calcolato.
  4. infine, l'errore di ricostruzione viene riprogrammato attraverso la rete.

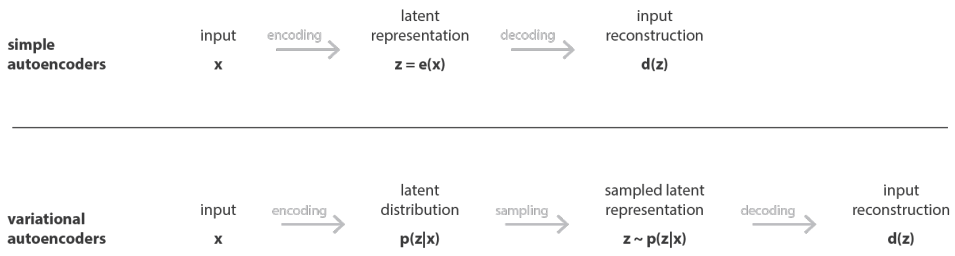


Figura 3.9: Differenze tra un normale autoencoder e un VAE [27].

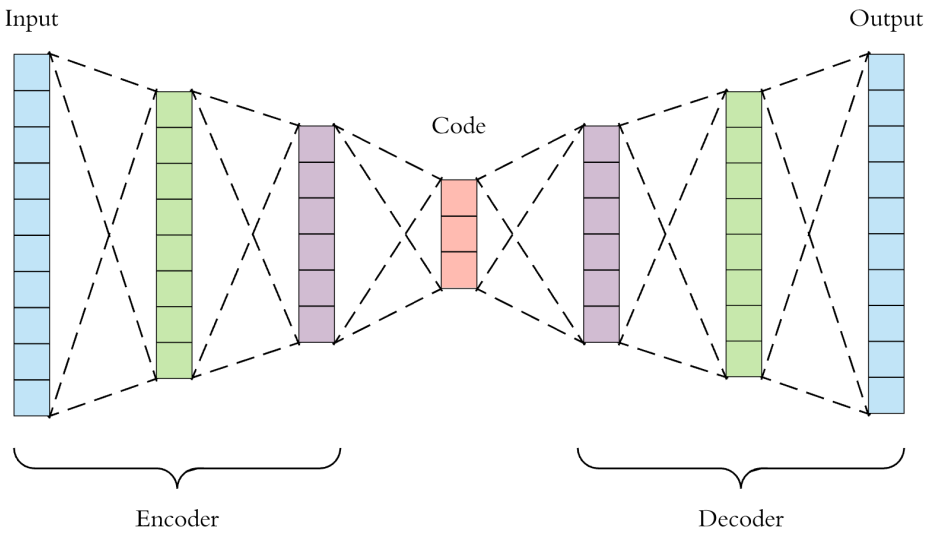


Figura 3.10: Architettura di un autoencoder [28].

## 3.3 Semi-Supervised Learning

L'apprendimento semi-supervisionato è il processo per trovare un classificatore migliore tra i dati etichettati e quelli senza etichetta. La metodologia di apprendimento semi-supervisionato può fornire elevate prestazioni di classificazione utilizzando dati senza etichetta. La metodologia può essere utilizzata per adattarsi a una varietà di situazioni identificando invece di specificare una relazione tra dati etichettati e non etichettati dai dati. Può produrre un miglioramento quando i dati senza etichetta possono ricostruire il limite di classificazione ottimale. Alcuni popolari modelli di apprendimento semi-supervisionato includono self-training, modelli di mixture, metodi basati su grafici, co-training e apprendimento multiview [29].

### 3.3.1 Apprendimento Semi-Supervisionato nell'Anomaly Detection

Il rilevamento di anomalie basato sull'apprendimento semi-supervisionato può essere costruito mediante gli autoencoder, utilizzando l'**errore di rico-**

**struzione come punteggio di anomalia**. Infatti i punti dati con elevata ricostruzione sono considerati anomalie poichè solo i dati con istanze normali vengono utilizzati per addestrare l'autoencoder. Dopo l'allenamento, l'autoencoder ricostruirà bene i dati normali, ma non riuscirà a farlo con i dati di anomalia che non ha riscontrato in fase di addestramento. Data una soglia, quindi, se l'errore di ricostruzione del dato, in fase di inferenza, è superiore alla soglia allora possiamo classificare quel dato come anomalo altrimenti come normale [30](figura 3.9). La soglia può essere scelta con molti metodi. Ad esempio il valore può essere deciso osservando l'n-esimo percentile della distribuzione degli errori del normale set di dati.Infatti l'n-esimo percentile è una statistica che indica il valore al di sotto del quale una determinata percentuale di osservazioni in un gruppo di dati cadono le osservazioni (prendendo, ad esempio, il 90esimo percentile è il valore al di sotto del quale è possibile trovare il 90% delle osservazioni). [21].

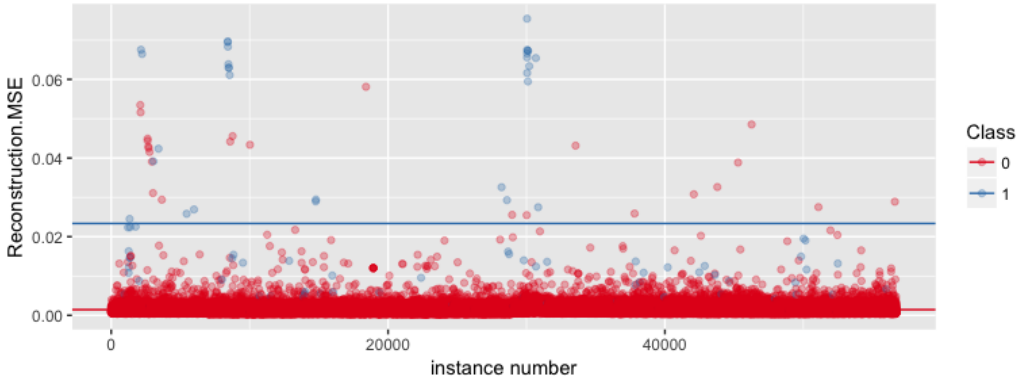


Figura 3.11: Esempio di classificazione secondo l'errore di ricostruzione (rappresentato sull'asse $y$). Tutti i punti sopra la soglia fissata (linea blu) possono essere classificati come anomali (alto errore di ricostruzione). Al contrario, quelli sotto soglia sono punti normali (basso errore di ricostruzione) [31].

Nel caso degli HPC, un metodo di apprendimento semi-supervisionato basato sugli autoencoder viene utilizzato attraverso l'apprendimento dello stato normale dei nodi di calcolo del sistema; una volta addestrati, gli autoencoders possono identificare situazioni anomale nei sistemi HPC e le etichette sono necessarie solo per ottenere il normale set di dati; successivamente, il

training procede in modo non supervisionato. In questo modo è possibile rilevare anche guasti non inclusi nel set di training. Inoltre,questo approccio non necessità di iniettare anomalie durante la fase di addestramento (considerando che le tecniche supervisionate hanno questo requisito), un'azione che non può essere possibile nella maggior parte dei sistemi di produzione HPC senza incorrere in gravi inattività [21].

## 3.4 Reinforcement Learning

L'apprendimento per rinforzo risale ai primi tempi della cibernetica , della statistica, psicologia, neuroscienze e informatica. Negli ultimi dieci anni ha attratto crescente interesse per le comunità di machine learning e intelligenza artificiale. L'obiettivo è la realizzazione di un modo di programmare gli agenti[1] con **ricompensa e punizione** senza necessità di specificare come deve essere realizzato il compito. Pertanto, l'apprendimento per rinforzo è **il problema affrontato da un agente che deve imparare il comportamento attraverso interazioni di prova ed errore con un ambiente dinamico**. Esistono due strategie principali per risolvere i problemi di apprendimento di rinforzo. Il primo è cerca nello spazio dei comportamenti per trovare quello che si comporta bene nell'ambiente.Il secondo è usare tecniche statistiche e metodi di programmazione dinamica per stimare l'utilità dell'assunzione azioni negli stati del mondo. Nel modello standard di apprendimento del rinforzo, un agente è collegato al suo ambiente tramite percezione e azione. Ad ogni fase di interazione l'agente riceve come input, $i$, qualche indicazione dello stato o degli stati correnti dell'ambiente; l'agente quindi sceglie un'azione, $a$, da generare come output. L'azione cambia lo stato di ambiente e il valore di questa transizione di stato viene comunicato all'agente tramite un segnale di rinforzo scalare, $r$. Il comportamento dell'agente, $B$, dovrebbe scegliere le azioni che tendono per aumentare la somma a lungo termine dei valori del segnale di rinforzo [32].

[1]L'agente è una qualunque entità in grado di percepire l'ambiente che lo circonda attraverso dei sensori e di eseguire delle azioni attraverso degli attuatori.

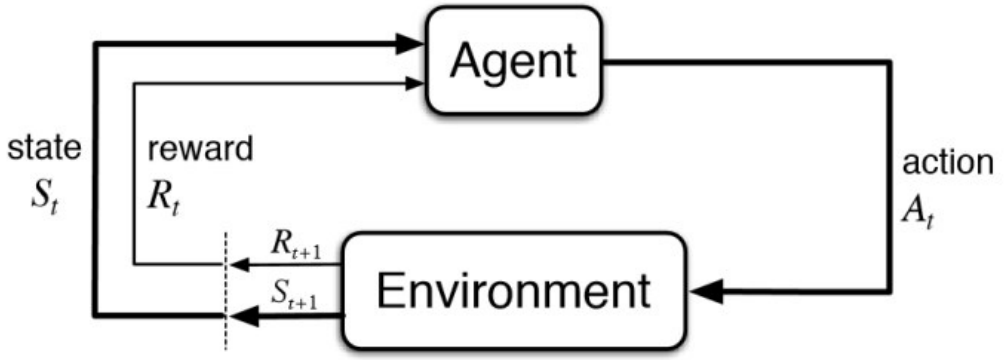


Figura 3.12: Architettura tipica dell'apprendimento per rinforzo [33].

Uno dei più conosciuti algoritmi di apprendimento per rinforzo è il **Q-Learning**. Esso fa parte della famiglia di algoritmi adottati nella tecniche delle differenze temporali, adottate nel caso di modelli a informazione incompleta. Uno dei suoi maggiori punti di rilievo consiste nell'abilità di comparare l'utilità aspettata delle azioni disponibili senza richiedere un modello dell'ambiente. Il suo obiettivo è quello di permettere ad un sistema di apprendimento automatico di adattarsi all'ambiente che lo circonda migliorando la scelta delle azioni da eseguire. Per giungere a questo obiettivo, cerca di massimizzare il valore del successivo premio per sconto[2]. Pertanto, dato un insieme di stati $S$, un insieme di azioni per stato $A$,per ogni azione $a \in A$ l'agente va da uno stato all'altro. Ogni stato fornisce all'agente una ricompensa. L'obiettivo dell'agente è massimizzare la ricompensa appena ricevuta. L'agente fa questo apprendendo quali sono le azioni ottimali associate ad ogni stato. Quindi l'algoritmo è provvisto di una funzione per calcolare la Qualità di una certa coppia stato-azione: $Q : S \times A \rightarrow \mathbb{R}$. Prima che l'apprendimento inizi, $Q$ restituisce un valore fisso, scelto dal progettista. Poi, ogni volta che l'agente riceve una ricompensa (lo stato è cambiato) vengono calcolati nuovi valori per ogni combinazione stato-azione. Il cuore dell'algoritmo fa uso di un processo iterativo di aggiornamento e correzione basato sulla nuova informazione

[2]Rinforzo con premio scontato: il rinforzo è distribuito per tutti gli istanti temporali ma aumenta a seconda di un parametro legato agli istanti temporali in cui viene applicato.

[34]:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + a_t(s_t, a_t) \times (r_t + \lambda * max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{3.7}$$

dove $Q(s_t, a_t)$ alla destra dell'equazione è il vecchio valore, $a_t(s_t, a_t)$ il tasso di apprendimento, $r_t$ la ricompensa, $\lambda$ il fattore di sconto, $max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$ il valore futuro massimo. L'algoritmo termina quando lo stato $s_{t+1}$ è uno stato finale (o stato di assorbimento). Una semplice implementazione di Q-learning usa tabelle per memorizzare i dati. Tuttavia questo approccio perde fattibilità al crescere del livello di complessità del sistema. Una possibile soluzione a questo problema prevede l'uso di una rete neurale artificiale come approssimatore di funzione [34].

## 3.5 Metodologie e tecniche utilizzate

In questo lavoro sono state utilizzate alcune tecniche analizzate nelle precedenti sezioni. In particolare, il nostro approccio è stato quello dell' **apprendimento supervisionato** utilizzando le **reti neurali artificiali**. Inoltre, a causa dell'elevata dimensionalità del nostro set di dati, sono stati utilizzati gli **autoencoder** per estrarre le **features più rilevanti**. Infatti, l'autoencoder è stato allenato per ricostruire, in modo più fedele possibile, l'input. In seguito sono stati utilizzati i layer dell'encoder allenato, per costruire un classificatore, aggiungedo un ulteriore hidden layer e un neurone di output per la classificazione . In figura 3.13 vi è illustrato il modello appena descritto.

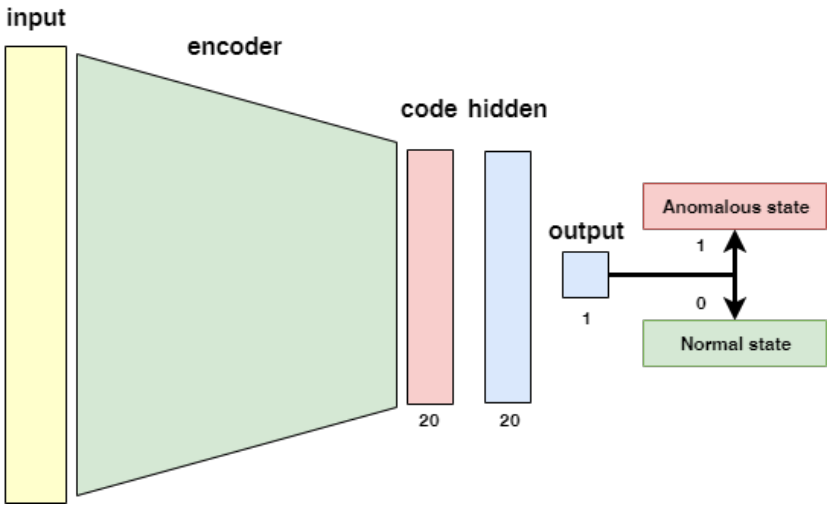


Figura 3.13: Íl modello utilizzato in questo lavoro.

La classificazione avviene mediante una **soglia** essendo il valore di output del neurone è un valore compreso tra 0 e 1. Impostando una soglia a 0.5, classifichiamo come anomalo l'input che ha un valore di uscita superiore alla soglia, invece, se il valore di uscita è inferiore alla soglia, allora viene classificato come normale. Il valore di soglia può essere modificato per ottenere il risultato voluto come si nota in figura 3.14.

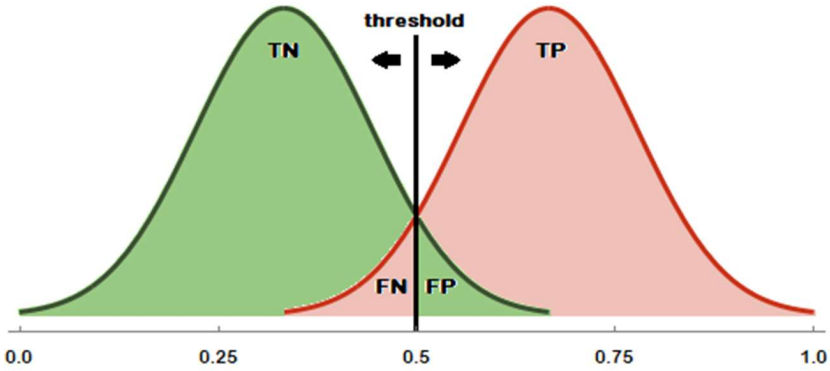


Figura 3.14: La soglia discrimina i punti anomali e i punti normali. Il valore di soglia può essere modificato in base al tipo di applicazione.

In un secondo momento il modello è stato modificato per un task di

classificazione **multi-etichetta**. In quest'ultimo caso il modello è simile al precedente tranne nei neuroni di output. Infatti, per un task di questo tipo, il numero di neuroni di output deve essere uguale al numero delle classi da individuare. I vari parametri impostati ai modelli verranno discussi nelle sezioni apposite. Per la realizzazione dei modelli di deep learning è stato utilizzato la libreria software scritta in Python, **Keras**. Keras è un'API di reti neurali di alto livello in grado di funzionare su TensorFlow, CNTK o Theano. È stato sviluppato con l'obiettivo di consentire una rapida sperimentazione [35]. Il backend utilizzato è **Tensorflow**, una piattaforma open source end-to-end per l'apprendimento automatico [36].

Capitolo 4

# Estrazione ed Elaborazione dei Dati

L'infrastruttura di raccolta dati implementata negli HPC del Marconi utilizza il *framework real-time* chiamato **Examon**. Grazie ad esso, i dati originati da fonti eterogenee (ad esempio da sensori fisici e moduli software) vengono raccolti e archiviati in un'unica struttura con un formato di accesso uniforme. Questo aiuta notevolmente la creazione di un set di dati con una visione d'insieme dell'intero cluster e del suo stato. I componenti principali dell'infrastruttura Examon sono diversi *daemon* software a basso costo, lungo i nodi di calcolo, strettamente accoppiati con essi. Questi daemon monitorano molte prestazioni e metriche di utilizzo e consumo energetico di ciascun nodo [21].

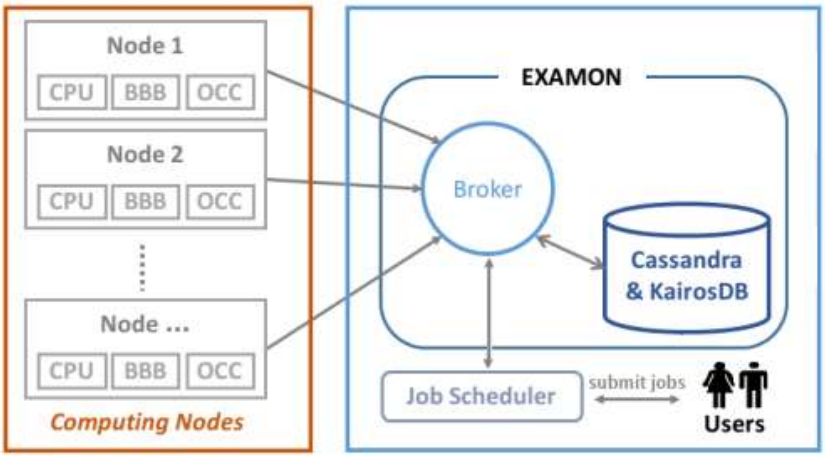


Figura 4.1: Architettura di Examon [21].

Le metriche estratte sono raggruppate in "plugin" in base alla loro origine. Ogni plugin fornisce informazioni su un diverso aspetto del nodo. I tre plugin da cui sono stati estratti i dati per questo lavoro sono: **Ganglia** per l'estrazione delle metriche relative alla CPU e al sistema (RAM, memoria libera in memoria e molti altri), **Confluent** per l'estrazione delle metriche relative alla comunicazione, temperatura e consumo ed infine **Nagios** che ci fornisce lo stato dell'intero sistema. In particolare, quando un'anomalia viene rilevata, manualmente è registrato qui con un valore numerico che va da 0 a 3 e una descrizione testuale. Il punteggio numerico rappresenta l'urgenza livello per intervento tecnico. Se è basso (0 o 1) rappresenta normale comportamenti o loro piccole variazioni che non influiscono sul nodo prestazione. Invece, un valore più alto (2 o 3) indica uno stato critico che deve essere risolto al più presto (ad esempio, è possibile che rimanga un nodo disconnesso dalla rete perché si verifica un errore di collegamento) [22].

Listing 4.1: Esempio di query ad Examon-client

```
data = sq.SELECT("state","description").FROM("*")
    .WHERE(plugin="nagios_pub",node="r183c13s04")
    .TSTART("13-10-2019 00:00:00")
    .TSTOP("15-10-2019 00:00:00")
    .execute()
```

| timestamp | value | name | node | state |
|---|---|---|---|---|
| 2019-10-08 09:00:00 | DOWN+DRAIN matches a critical state | plugin_output | r183c12s04 | 2 |
| 2019-10-08 09:15:00 | DOWN+DRAIN matches a critical state | plugin_output | r183c12s04 | 2 |
| 2019-10-31 13:45:00 | ALLOCATED+DRAIN matches a critical state | plugin_output | r183c12s04 | 2 |
| 2019-11-01 21:07:00 | ERROR, the command timeouted: /sbin/runuser -c '/usr/bin/ssh master01 /cinecalocal/nagios/passive/mhsc_nodehealth_a3.pl' root | plugin_output | r183c12s04 | 2 |
| 2019-11-02 17:45:00 | C({topwaiter: monitored, PCacheMsgMgrThread: SINCE 603.7923 sec.][/marconi,/marconi_work]) W() O() | plugin_output | r183c12s04 | 2 |

Figura 4.2: Esempio di dati provenienti da Nagios

## 4.1 Estrazione dei dati

L'estrazione dei dati è avvenuta attraverso *Examon-client plugin* sviluppato da Francesco Beneventi [37]. Il plugin ha permesso di richiedere i dati attraverso un linguaggio **SQL-like** (listing 4.1). Inoltre, è stato utilizzato anche parte del lavoro di estrazione dei dati, sviluppato in "*Anomaly Detecion in HPC Systems*" di Kevin Leto [22]. I dati estratti dai plugin **Ganglia** e **Confluent** forniscono oltre alle varie metriche dei sensori, anche il timestamp. I dati sono,inoltre, campionati ogni 5 minuti. Invece, i dati estratti dal plugin **Nagios**, sono etichettati in vari modi. I più interessanti, poichè rappresentano uno stato anomalo, sono quelli con etichetta "***DOWN+DRAIN***", ed, in particolare, quelli con lo stato 2 che rappresenta, appunto, un comportamento anomalo e urgente da risolvere per l'amministratore del sistema. Tuttavia, i plugin non ci forniscono i dettagli sull'anomalia rilevata. Questo sarà, infatti, uno degli obiettivi di questo lavoro, studiare le anomalie attraverso il comportamento del modello. Infine i dati estratti dai vari plugin

vengono uniti attraverso il timestamp e normalizzati in valori compresi tra 0 e 1, per poi essere salvati in formato *Comma-separated values* (CSV) per essere analizzati.

| timestamp | value | name | node | state |
|---|---|---|---|---|
| 2019-10-07 00:45:00.100 | C() W() O({mmgetstate-active:rdma-on:no-relevant-waiters][/marconi,/marconi_work]) | plugin_output | r183c12s04 | 0 |
| 2019-10-07 00:45:00.100 | DOWN+DRAIN matches a critical state | plugin_output | r183c12s04 | 2 |
| 2019-10-07 01:00:00.014 | C() W() O(/,/boot,/boot/efi,/dev,/dev/shm,/opt,/scratch_local,/tmp,/usr,/var) | plugin_output | r183c12s04 | 0 |
| 2019-10-07 01:00:00.014 | C() W() O(/[19%],/boot[14%],/boot/efi[4%],/dev[0%],/dev/shm[0%],/opt[59%],/scratch_local[1%],/tmp[1%],/usr[24%],/var[1… | plugin_output | r183c12s04 | 0 |
| 2019-10-07 01:00:00.014 | OK, total memory 197537984 greater than 196000000 | plugin_output | r183c12s04 | 0 |
| 2019-10-07 01:00:00.014 | SSH OK - OpenSSH_6.6.1 (protocol 2.0) | plugin_output | r183c12s04 | 0 |

Figura 4.3: Esempio di dati provenienti da Nagios: in evidenza i dati a noi interessanti cioè quelli con etichetta "DOWN+DRAIN" e "state" 2 [22]

In questo lavoro sono stati estratti i dati dai nodi: r183c09s01, r183c09s02, r183c09s03, r183c09s04, r183c10s01, r183c10s02, r183c10s03, r183c10s04, r183c11s01, r183c11s02, r183c11s03, r183c11s04, r183c12s01, r183c12s02, r183c12s03, r183c12s04, r183c13s01, r183c13s02, r183c13s03, r183c13s04, r183c14s01, r183c14s02, r183c14s03, r183c14s04, r183c15s01, r183c15s03, r183c15s04, r183c16s03, r183c16s04. Il periodo considerato è dal 31/10/2019 al 12/11/2019.

### 4.1.1 Dataset Generale

Il dataset generale comprende i nodi appena citati, i cui dati sono opportunamente divisi per l'80 % in training set e il restante test set. Il training set di ogni nodo viene ,quindi, unito in un unico dataset.
Per la classificazione multi-etichetta è stato costruito un nuovo dataset in cui abbiamo aggiunto altre due etichette per distinguere le anomalie dei nodi r183c10s04 e r183c16s04 da quelle degli altri nodi. Infatti, questi due nodi presentavano un periodo molto anomalo quindi è stato costruito un dataset e un modello differente. In entrambi i dataset sono state considerate le fea-

tures in comune a tutti i nodi, eliminando quelle che non lo sono. Pertanto, il dataset finale contiene 124 features e 66358 records.

### 4.1.2 Strumenti Software

Per quanto riguarda le librerie software utilizzate, dopo l'estrazione dei dati attraverso Examon-plugin, la trasformazioni e l'elaborazioni sono state effettuate con **Pandas**, uno strumento di analisi e manipolazione dei dati open source costruito sulla base del linguaggio di programmazione Python [38].

Capitolo 5

# Transfer Learning

Molti metodi di apprendimento automatico funzionano bene solo sotto un presupposto comune: i dati di training e test sono costruiti dallo stesso spazio delle caratteristiche e dalla stessa distribuzione (figura 5.1). Quando la distribuzione cambia, la maggior parte dei modelli statistici deve essere ricostruito da zero utilizzando i dati di allenamento appena raccolti. In molte applicazioni del mondo reale, è costoso o impossibile raccogliere nuovamente i dati di addestramento necessari e ricostruire i modelli. Pertanto, l'obiettivo è ridurre la necessità e gli sforzi per raccogliere nuovamente i dati di addestramento (figura 5.2) [39].

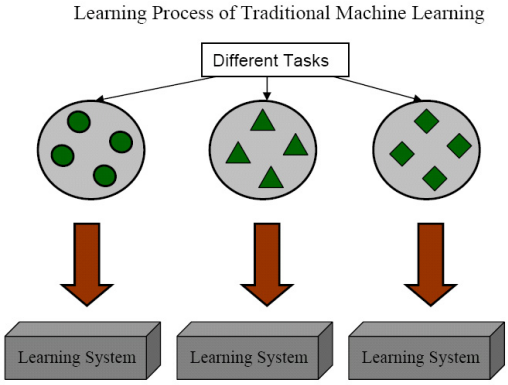


Figura 5.1: Processo di apprendimento di un normale task di Machine Learning [39].

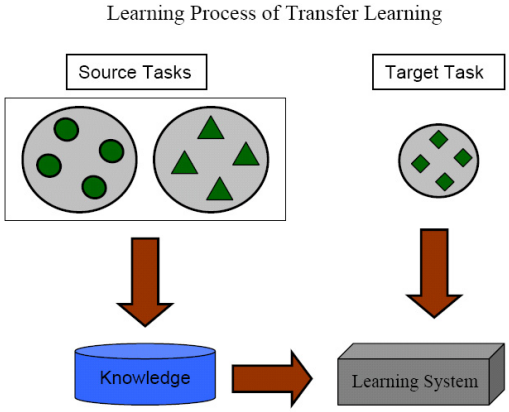


Figura 5.2: Processo di apprendimento in un task di Transfer Learning [39].

L'obiettivo in questo capitolo è quello di **trasferire la conoscenza** del modello realizzato in *Anomaly Detection in HPC Systems* di K.Leto [22], nel quale è stato realizzato un modello di machine learning per il nodo r183c12s04, ad altri nodi del sistema HPC analizzato. In un primo momento viene utilizzato il modello per analizzarne il comportamento sugli altri nodi per poi costruire un modello generale per il rilevameno delle anomalie su tutti i nodi testati. Il training set per la realizzazione del modello generale comprende i dati provenienti dai nodi sotto esame, scelti opportunamente e uniti in un unico dataset mentre il test set è stato realizzato con un altra porzione di dati in modo che training set e test set non abbiano gli stessi

esempi.

## 5.1 Metodologia

Il modello (da ora in poi chiamato Modello Uno) utilizzato in *Anomaly Detection in HPC Systems*[22] è stato addestrato sul nodo r183c12s04. In un primo momento, per ridurre la dimensionalità del set di dati di addestramento, quindi, per estrarre le features più rilevanti, è stato utilizzato un autoencoder. L'autoencoder è composto da layer che contengono, in successione, 100, 80, 60, 40 e 20 neuroni. Il code layer, quindi, è formato da 20 neuroni. La parte di decoder segue la stessa struttura dell'encoder ma ovviamente al contrario. Ogni layer ha la funzione di attivazione, *ReLu*, ad eccezione dell'ultimo layer in cui è stata utilizzata una *sigmoid*. Con 200 epoche, un *batch size* di 75 e un *validation-split* del 10% del training set la loss ottenuta è la seguente:

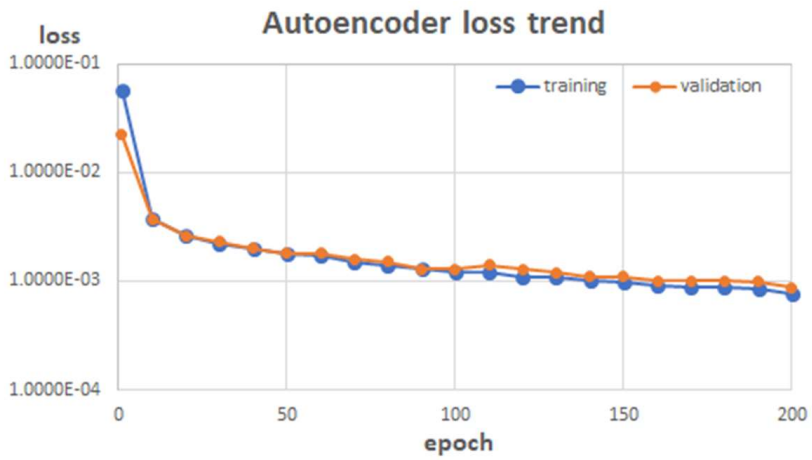


Figura 5.3: Andamento della Loss e la Loss del validation set dell'autoencoder [22].

La parte di encoder, già addestrata, viene aggiunta al modello finale per la classificazione dell'anomalia, formando il modello come si osserva in figura

5.4. Il classificatore è stato addestrato con 110 epoche, un batch size di 75 e validation-split[1] del 10% del training set.

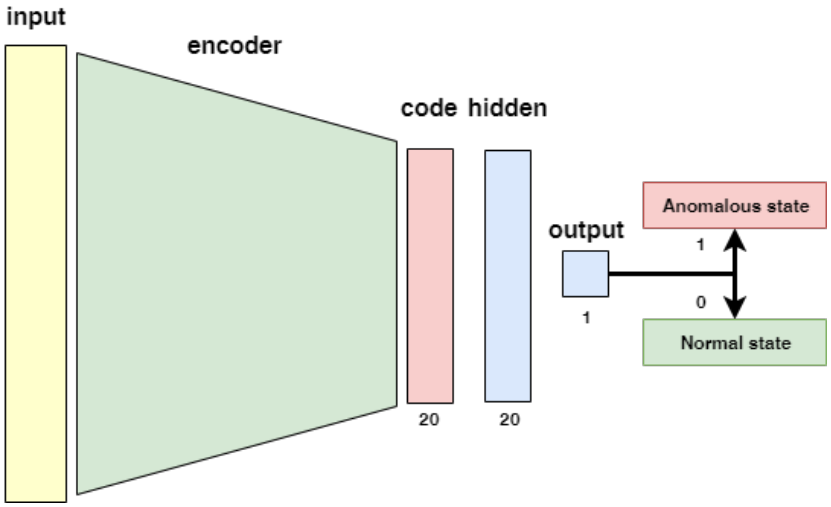


Figura 5.4: Modello finale composto dall'encoder e il classificatore [22].

Il secondo modello (che chiameremo Modello Due), realizzato in questo lavoro, apporta alcune modifiche a quello appena descritto. Innanzitutto viene aggiunto un layer con 200 unità come primo e ultimo layer dell'autoencoder. In seguto per l'addestramento siamo passati a 300 **epoche**, un ***validation-split*** impostato al 20% del training set e un **batch size** di 75, ottenendo il seguente risultato per la loss:

[1]Il validation split è un ulteriore segmentazione del dataset oltre alla divisione in training set e test set. Viene utilizzato per misurare le performance del modello durante l'addestramento; è effettuata in modo automatico da Keras, la divisione, impostandone la percentuale di split.

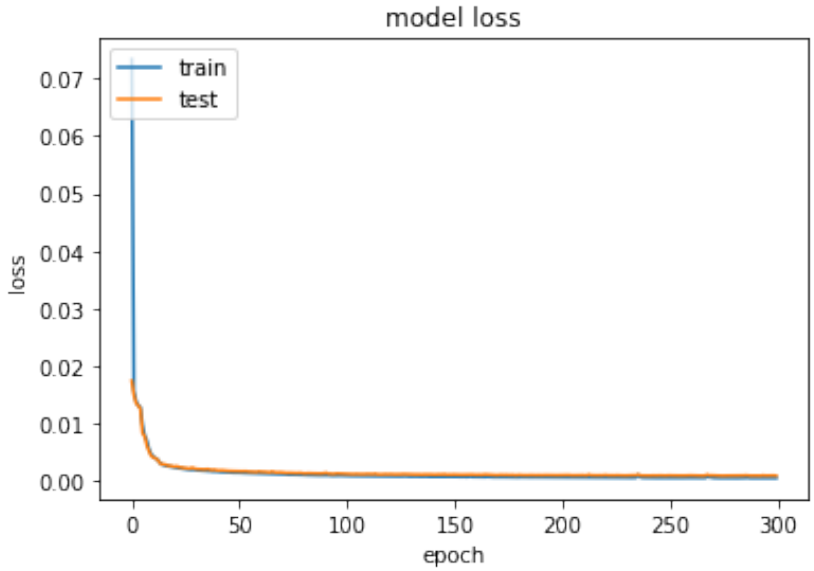


Figura 5.5: Loss e Validation Loss del secondo modello.

La parte per la classificazione è rimasta invariata. L'addestramento è stato effettuato su un training set che comprende i nodi, descritti nel capitolo precedente.
Il terzo modello (Modello Tre), per la classificazione multi-etichetta delle anomalie, utilizza l'autoencoder simile al Modello Uno, addestrato sul dataset generale, mentre il classificatore è stato addestrato di volta in volta su ogni nodo (la parte dell'encoder rimane comunque invariata, essendo già addestrata sul dataset generale) in modo da analizzare la trasferibilità dell' autoencoder. Il classificatore per la classificazione mult-etichetta presenta 4 neuroni di output e la loss *"categorical_crossentropy"* [35]. Per l'addestramento dell'autoencoder abbiamo utilizzato un numero di epoche pari a 1000 con un batch size di 256 e validation split del 20% ottenendo la seguente loss:

```
Epoch 1000/1000
64631/64631 [==============================] - 1s 18us/step - loss: 3.4368e-04 - val_loss: 3.6814e-04
```

Figura 5.6: Loss e Validation Loss del terzo modello.

Le metriche utilizzate per misurare le performance del modello sono l'a-

curatezza (accuracy), recall, precision e f1-score, le quali si calcolano nei seguenti modi (TP indica i veri positivi, TN indica i veri negativi, FP indica i falsi positivi e FN indica i falsi negativi):

- $Accuracy = \frac{Numero\,di\,predizioni\,corrette}{Numero\,di\,predizioni\,totali} = \frac{TP+TN}{TP+FP+FN+TN}$
- $Recall = \frac{TP}{TP+FN}$
- $Precision = \frac{TP}{TP+FP}$
- $F1Score = 2\frac{precision*recall}{precision+recall}$

## 5.2 Risultati Modello Uno

I primi test sono stati effettuati con il Modello Uno per studiarne la trasferibilità su altri nodi. Le metriche utilizzate per misurare il modello sono state l'accuratezza e la F1Score. Queste due metriche sono state misurate considerando solo le anomalie poichè il set di dati non è bilanciato tra punti normali e punti anomali. Pertanto, i risultati sono riportati nella tabella 5.1.

Tabella 5.1: Risultati del Modello Uno

| Nodo | Accuracy | F1Score Anomalies |
|---|---|---|
| r183c09s03 | 0% | 0.0 |
| r183c09s04 | 9% | 0.17 |
| r183c10s01 | 25% | 0.4 |
| r183c10s02 | 34% | 0.51 |
| r183c10s03 | 0% | 0.0 |
| r183c11s01 | 0% | 0.0 |
| r183c11s02 | 0% | 0.0 |
| r183c11s03 | 1.4 % | 0.0 |

### 5.2.1 Considerazioni

Come si nota dai risultati nella tabella 5.1, il Modello Uno non è in grado di prevedere le anomalie al di fuori del nodo su cui è stato addestrato

(r183c12s04). Mentre nel nodo di addestramento ha ottenuto un accuracy del 99.8% , precision del 97.3 %, recall del 100% e f1 score di 0.98 [22], sugli altri nodi ottiene risultati molto scarsi. Pertanto, questo modello non sembra essere trasferibile, ne possiamo costruire un modello generale con il solo addestramento effettuato sul nodo r183c12s04. Il prossimo passo è costruire un modello generale utilizzando un set di data per l'addestramento che comprenda tutti i nodi.

## 5.3 Risultati Modello Due

Come si è osservato nella sezione precedente, con un modello addestrato su un solo nodo non è possibile costruire un modello generale e capace di poter essere trasferibile. Perciò, come abbiamo visto nel capitolo 4, è stato costruito un set di dati che comprende 27 nodi. In seguito, abbiamo testato il modello su ogni test set di un nodo e non su un test set generale, in modo da vedere il comportamento locale del modello. Il modello addestrato su questo dataset presenta questi risultati (tabella 5.2 e tabella 5.3):

Tabella 5.2: Risultati del Modello Due

| Nodo | Accuracy | F1Score Normale | F1Score Anomalie | F1Score Totale |
|---|---|---|---|---|
| r183c09s01 | 97.95% | 0.99 | 0.61 | 0.98 |
| r183c09s03 | 97.63% | 0.99 | 0.48 | 0.97 |
| r183c09s04 | 98.23% | 0.99 | 0.29 | 0.98 |
| r183c10s01 | 97.84% | 0.99 | 0.50 | 0.97 |
| r183c10s02 | 99.05% | 1.00 | 0.80 | 0.99 |
| r183c10s03 | 98.27% | 0.99 | 0.67 | 0.98 |
| r183c11s01 | 97.95% | 0.99 | 0.56 | 0.98 |
| r183c12s04 | 99.79 % | 1.00 | 0.99 | 1.00 |
| r183c11s03 | 98.40 % | 0.99 | 0.36 | 0.98 |
| r183c11s04 | 98.28 % | 0.39 | 0.98 | 0.98 |
| r183c15s01 | 98.32 % | 0.99 | 0.51 | 0.98 |
| r183c15s02 | 98.32 % | 0.99 | 0.34 | 0.98 |
| r183c15s03 | 98.42 % | 0.99 | 0.53 | 0.98 |
| r183c15s04 | 97.73 % | 0.99 | 0.57 | 0.98 |
| r183c16s03 | 98.01 % | 0.99 | 0.45 | 0.98 |
| r183c09s02 | 97.84 % | 0.99 | 0.50 | 0.97 |
| r183c12s01 | 99.68 % | 1.00 | 0.95 | 1.00 |
| r183c12s02 | 97.33 % | 0.99 | 0.37 | 0.97 |
| r183c12s03 | 97.52 % | 0.99 | 0.51 | 0.97 |
| r183c13s01 | 99.70 % | 1.00 | 0.94 | 1.00 |
| r183c13s02 | 98.81 % | 0.99 | 0.78 | 0.99 |
| r183c13s03 | 98.40 % | 0.99 | 0.40 | 0.98 |
| r183c13s04 | 97.52 % | 0.99 | 0.51 | 0.97 |
| r183c14s01 | 98.40 % | 0.99 | 0.44 | 0.98 |
| r183c14s02 | 97.30 % | 0.99 | 0.82 | 0.97 |
| r183c14s03 | 99.14 % | 1.00 | 0.87 | 0.99 |
| r183c14s04 | 98.85 % | 0.99 | 0.75 | 0.99 |

L'accuratezza è calcolata sul dataset con punti normali e punti anomali mentre la F1 score è divisa in *Normal, Anomalies, Average* che rispettivamente indicano la metrica sul dataset con soli punti normali, dataset con soli punti

anomali e dataset totale. Per quanto riguarda la recall e precision otteniamo il seguente risultato medio (tabella 5.3):

Tabella 5.3: Risultati del Modello Due

| Media Accuray(Totale) | Media Recall(Solo Anomalie) | Media F1Score(Solo Anomalie) | Media Precision(Solo Anomalie) |
|---|---|---|---|
| 98.32% | 55% | 0.58 | 84.33% |

Anche in questo caso sono state considerate le misure sul dataset con solo anomalie.
Infine sono mostrate le matrici di confusione:

Tabella 5.4: Matrici di Confusione del Modello Due

Predetti - r183c09s01

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 893 | 5 | 898 |
| | Anomalie | 14 | 15 | 29 |
| | Total | 907 | 20 | 927 |

Predetti - r183c09s03

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 895 | 3 | 898 |
| | Anomalie | 19 | 10 | 29 |
| | Total | 916 | 13 | 929 |

Predetti - r183c09s04

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 1104 | 0 | 1104 |
| | Anomalie | 20 | 4 | 24 |
| | Total | 1124 | 4 | 1128 |

Predetti - r183c10s02

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 919 | 1 | 920 |
| | Anomalie | 8 | 18 | 26 |
| | Total | 927 | 19 | 946 |

Predetti - r183c10s03

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 895 | 3 | 898 |
| | Anomalie | 13 | 16 | 29 |
| | Total | 908 | 19 | 927 |

Predetti - r183c11s01

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 896 | 2 | 898 |
| | Anomalie | 17 | 12 | 29 |
| | Total | 925 | 14 | 939 |

Predetti - r183c11s04

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 1079 | 4 | 1083 |
| | Anomalie | 15 | 6 | 21 |
| | Total | 1094 | 10 | 1104 |

Predetti - r183c12s04

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 1317 | 2 | 1319 |
| | Anomalie | 1 | 108 | 109 |
| | Total | 1318 | 110 | 1428 |

Predetti - r183c11s03

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 1105 | 0 | 1105 |
| | Anomalie | 18 | 5 | 23 |
| | Total | 1123 | 5 | 1128 |

Predetti - r183c15s01

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 987 | 1 | 988 |
| | Anomalie | 16 | 9 | 25 |
| | Total | 1003 | 10 | 1113 |

Predetti - r183c15s02

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 1104 | 0 | 1104 |
| | Anomalie | 19 | 5 | 24 |
| | Total | 1123 | 5 | 1128 |

Predetti - r183c15s03

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 988 | 0 | 988 |
| | Anomalie | 16 | 9 | 25 |
| | Total | 1004 | 9 | 1013 |

Predetti - r183c15s04

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 892 | 6 | 898 |
| | Anomalie | 15 | 14 | 29 |
| | Total | 907 | 20 | 927 |

Predetti - r183c16s03

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 1073 | 5 | 1078 |
| | Anomalie | 17 | 9 | 26 |
| | Total | 1090 | 14 | 1104 |

Predetti - r183c09s02

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 897 | 1 | 898 |
| | Anomalie | 19 | 10 | 29 |
| | Total | 916 | 11 | 927 |

Predetti - r183c11s04

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 977 | 11 | 988 |
| | Anomalie | 1 | 24 | 25 |
| | Total | 978 | 35 | 1013 |

Predetti - r183c12s01

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 896 | 2 | 898 |
| | Anomalie | 1 | 28 | 29 |
| | Total | 897 | 30 | 927 |

Predetti - r183c12s02

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 978 | 10 | 988 |
| | Anomalie | 17 | 8 | 25 |
| | Total | 995 | 18 | 1013 |

Predetti - r183c12s03

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 892 | 4 | 896 |
| | Anomalie | 17 | 12 | 29 |
| | Total | 909 | 16 | 925 |

Predetti - r183c13s01

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 987 | 1 | 988 |
| | Anomalie | 2 | 23 | 25 |
| | Total | 989 | 24 | 1013 |

Predetti - r183c13s02

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 897 | 1 | 898 |
| | Anomalie | 10 | 19 | 29 |
| | Total | 907 | 20 | 927 |

Predetti - r183c13s03

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 1104 | 0 | 1104 |
| | Anomalie | 18 | 6 | 24 |
| | Total | 1122 | 6 | 1128 |

Predetti - r183c13s04

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 892 | 6 | 898 |
| | Anomalie | 17 | 12 | 29 |
| | Total | 1122 | 6 | 1128 |

Predetti - r183c14s01

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 1102 | 1 | 1103 |
| | Anomalie | 17 | 7 | 24 |
| | Total | 1119 | 8 | 1127 |

Predetti - r183c14s02

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 844 | 9 | 853 |
| | Anomalie | 16 | 58 | 74 |
| | Total | 860 | 67 | 927 |

Predetti - r183c14s03

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 892 | 6 | 898 |
| | Anomalie | 2 | 27 | 29 |
| | Total | 894 | 33 | 927 |

Predetti - r183c14s04

| | | Normali | Anomalie | Totale |
|---|---|---|---|---|
| Reali | Normali | 1095 | 9 | 1104 |
| | Anomalie | 4 | 20 | 24 |
| | Total | 1099 | 29 | 1128 |

### 5.3.1 I nodi r183c10s04 e r183c16s04

Come abbiamo già accennato nel capitolo 4, i nodi r183c10s04 e r183c16s04 presentano, nel periodo considerato, molte anomalie. Provando questo modello che è stato addestrato su questi due nodi, otteniamo il seguente risultato:

Tabella 5.5: Recall e F1Score del Modello Due sui nodi r183c10s04 e r183c16s04

| Nodo | Recall(Solo Anomalie) | F1Score(Solo Anomalie) |
|---|---|---|
| r183c10s04 | 26% | 0.41 |
| r183c16s04 | 5% | 0.10 |

### 5.3.2 Considerazioni

I risultati sono migliorati sensibilmente coinvolgendo nel modello tutti i nodi. Purtroppo ancora non è soddisfacente. Infatti, la recall media è di solo 55% e il f1 score è di 0.58. Tuttavia, con le anomalie di alcuni nodi, r183c11s02, r183c12s04, r183c12s01,r183c14s02, r183c14s03, r183c11s04, r183c10s02, otteniamo buoni risultati (f1 score maggiore di 0.8). Per quanto riguarda i nodi r183c10s04 e r183c16s04 otteniamo risultati molto scarsi quindi il learning di questo modello non è trasferibile per determinare le anomalie di questi due nodi. L'idea nella prossima sezione sarà quella di separare le anomalie di questi due nodi con un'altra etichetta (ora utilizziamo 0 per normale, 1 per anomalia) e provare un modello di multiclasse.

## 5.4 Modello Tre: classificazione multi-etichetta

Per questo task, abbiamo aggiunto al dataset il set di dati del nodi r183c10s04 distinguendo le sue anomalie con un ulteriore etichetta. La stessa modifica è stata effettuata nel dataset del nodo r183c16s04. Per l'addestramento il dataset è stato diviso in training set e test set e sono stati ottenuti **un'accuratezza al 95% e un f1 score (pesato tra le varie classi) a 0.95** sul test set. La matrice di confusione risulta essere:

Tabella 5.6: Matrici di Confusione del Modello Tre

| | | Predetti | | |
|---|---|---|---|---|
| | | **Normali** | **Anomalia 1** | **Anomalia 2** |
| **Reali** | **Normali** | 18381 | 68 | 0 |
| | **Anomalia 1** | 167 | 488 | 487 |
| | **Anomalia 2** | 0 | 160 | 447 |

Osservando la matrice di confusione e le metriche, il modello ottiene buoni risultati. Tuttavia quando viene utilizzato sul nodo r183c16s04 ottiene risultati molto scarsi: un'**accuratezza del 1.9% e uno f1 score di 0.03**. Anche questo modello non sembra essere trasferibile sui nodi che presentano molte anomalie.

## 5.5 Un ulteriore esperimento

Un ulteriore prova è stata effettuata con un modello avente l'encoder layer, quindi l'autoencoder, addestrato sul nodo su cui si è ottenuto il miglior risultato e il classificatore addestrato su ogni nodo e testato su di esso. Pertanto, i modelli realizzati sono pari al numero dei nodi. In questo modo è stata testata la trasferibilità prendendo solo l'autoencoder per l'estrazione delle features rilevanti. L'autoencoder, quindi, è stato addestrato sul nodo r183c12s04. I risultati sono stati i seguenti:

Tabella 5.7: Risultati Modello Quattro

| Nodo | Accuracy | F1Score Normale | F1Score Anomalie | F1Score Totale |
|---|---|---|---|---|
| r183c09s01 | 97.74% | 0.99 | 0.32 | 0.97 |
| r183c09s03 | 98.54% | 0.99 | 0.62 | 0.97 |
| r183c09s04 | 99.07% | 1.00 | 0.82 | 0.99 |
| r183c10s01 | 97.21% | 0.99 | 0.00 | 0.96 |
| r183c10s02 | 98.67% | 0.99 | 0.50 | 0.98 |
| r183c10s03 | 98.80% | 0.99 | 0.71 | 0.99 |
| r183c11s01 | 98.26% | 0.99 | 0.59 | 0.98 |
| r183c10s04 | 99.79 % | 0.00 | 1.00 | 1.00 |
| r183c11s03 | 99.59 % | 1.00 | 0.90 | 1.00 |
| r183c11s04 | 99.35 % | 1.00 | 0.87 | 0.99 |
| r183c15s01 | 98.96 % | 0.99 | 0.70 | 0.99 |
| r183c15s02 | 99.70 % | 1.00 | 0.92 | 1.00 |
| r183c15s03 | 100 % | 1.00 | 1.00 | 1.00 |
| r183c15s04 | 98.81 % | 0.99 | 0.60 | 0.99 |
| r183c16s03 | 99.11 % | 1.00 | 0.80 | 0.99 |
| r183c09s02 | 98.06 % | 0.99 | 0.45 | 0.98 |
| r183c12s01 | 99.35 % | 1.00 | 0.88 | 0.99 |
| r183c12s02 | 98.22 % | 0.99 | 0.52 | 0.98 |
| r183c12s03 | 98.38 % | 0.99 | 0.55 | 0.98 |
| r183c13s01 | 98.60 % | 0.99 | 0.81 | 0.98 |
| r183c13s02 | 98.06 % | 0.99 | 0.40 | 0.97 |
| r183c13s03 | 97.90 % | 0.99 | 0.48 | 0.98 |
| r183c13s04 | 98.38 % | 0.99 | 0.55 | 0.98 |
| r183c14s01 | 98.54 % | 0.99 | 0.64 | 0.98 |
| r183c14s02 | 99.03 % | 1.00 | 0.77 | 0.99 |
| r183c14s03 | 99.19 % | 1.00 | 0.83 | 0.99 |
| r183c14s04 | 98.22 % | 0.99 | 0.52 | 0.98 |

Per quanto riguarda la Recall e la Precision media, considerando solo punti anomali:

Tabella 5.8: Recall, F1Score, Precision del Modello Quattro

| Media Recall(Solo Anomalie) | Media F1Score(Solo Anomalie) | Media Precision(Solo Anomalie) |
|---|---|---|
| 57% | 0.66 | 86% |

### 5.5.1 Considerazioni

I risultati espressi nelle tabelle 5.7 e 5.8 indicano un leggero miglioramento con questo modello. Tuttavia, i risultati sono altalenanti: su alcuni nodi i risultati sono scarsi,su altri, invece, si riescono a predire tutti i punti anomali. In particolare, nel nodo r183c10s04 (il nodo che presenta tutto il periodo anomalo) si riesce ad ottenere una recall del 100%. Quest'ultimo risultato indica che le anomalie in questo nodo sono diverse rispetto alle altre, perciò deve essere effettuata una maggiore indagine per rilevare la natura di queste anomalie. Ciò verrà effettuato nel prossimo capitolo.

## 5.6 Conclusioni

Il lavoro in questo capitolo ha come obiettivo il trasferimento della "*knowledge*", acquisita dal modello in "*Anomaly Detection on HPC Systems*" [22], su altri nodi del stesso sistema HPC. I risultati dimostrano che il transfer learning può essere effettuato in parte. Infatti è dimostrato che bisogna costruire, ogni volta, un nuovo dataset con nuove anomalie per essere rilevate. In particolare, le difficoltà sono maggiori con alcuni nodi come il r183c10s04 e r183c16s04. Pertanto, non è stato possibile un transfer learning completo. Le cause sono riconducibili alle diverse forme di anomalie che possono ricorrere in un sistema HPC ma anche ai pochi punti anomali a disposizione. I modelli illustrati, infatti, riescono a classificare molto bene i punti normali anche grazie all'enorme quantità di essi rispetto a quelli anomali.
I possibili approcci per evitare i problemi riscontrati sono già disponibili in letteratura: lo *Zero-shot learning* e l'apprendimento semi-supervisionato.

### 5.6.1 Zero-shot Learning

Lo Zero-shot learning è un metodo di apprendimento molto promettente, in cui le classi coperte dalle istanze di addestramento e le classi che intendia-

mo classificare sono disgiunte. In altre parole, l'apprendimento Zero-shot riguarda l'utilizzo dell'apprendimento supervisionato senza ulteriori dati di allenamento. L'apprendimento zero-shot si riferisce a un caso d'uso specifico dell'apprendimento automatico in cui si desidera che il modello classifichi i dati sulla base di pochissimi o addirittura nessun esempio etichettato, il che significa classificare al volo. In breve questo approccio si basa sull'esistenza di un set di addestramento etichettato di classi viste e classi non viste. Entrambe le classi viste sono correlate in uno spazio vettoriale ad alta dimensione, chiamato spazio semantico, in cui la conoscenza delle classi viste può essere trasferita in quelle non viste. Solitamente, gli algoritmo Zero-shot prima mappano le istanze su attributi intermedi, che possono essere le classi viste (quelli con dati etichettati), attributi specificati dall'uomo o dipendenti dai dati. Quindi gli attributi previsti vengono mappati su un gran numero di classi non viste attraverso le basi di conoscenza. In questo modo, la previsione di classi non viste diventa possibile e non sono necessari dati di addestramento per tali classi. Il processo in due fasi: training e inference. Nella fase di training, viene acquisita la conoscenza degli attributi e, nella fase di inference, questa conoscenza viene utilizzata per classificare le istanze in una nuova serie di classi [40].

### 5.6.2 Approccio Semi-Supervisionato

Un altro approccio possibile è l'apprendimento semi-supervisionato. Il gran numero di dati di punti normali può suggerire l'uso di questa tecnica di apprendimento. In particolare, come abbiamo già visto nella sezione 3.2.4, con l'utilizzo di un autoencoder, addestrato sui punti normali, è possibile grazie all'errore di ricostruzione riuscire ad individuare i punti anomali. Questo approccio è molto promettente dal momento che le anomalie sono molto rare in sistemi come gli HPC e in generale in tutti i sistemi critici.

Capitolo 6

# Explainability

Con la rapida diffusione del Machine Learning e dei suoi modelli, in particolare, le reti neurali artificiali, è nato il problema dell'interpretabilità di questi modelli. Infatti molto spesso sono considerati delle ***black box*** che ricevono degli input per produrre un risultato senza sapere come è stato prodotto e perchè. Se si pensa che questi modelli sono impiegati anche in ambiti commerciali non prettamente scientifici, nei processi decisionali, questi modelli devono essere facilmente interpretabili da chiunque e non solo da esperti. Inoltre accade che neanche il progettista riesce a spiegarsi del perchè di un determinato risultato. Interpretare questi modelli significa dare o fornire il significato o spiegare e presentare in termini comprensibili alcuni concetti. Pertanto, nel data mining e nell'apprendimento automatico, l'interpretazione è definita come l'abilità di spiegare (*Explainability*) o fornire il significato in termini comprensibili a un essere umano. Queste definizioni assumono implicitamente che i concetti espressi in termini comprensibili che compongono una nozione non necessita di ulteriori spiegazioni. In sostanza, un "*Explanation*" è un "***Interfaccia***" tra uomo e decisore che è allo stesso tempo un *proxy* accurato del decisore e comprensibile per l'uomo. Tuttavia, un explanation potrebbe non essere necessaria se non ci sono decisioni da prendere sull'esito della predizione. Ad esempio, se vogliamo sapere se un'immagine contiene o meno un gatto e queste informazioni non è tenuto a prendere alcuna decisione cruciale o non ci sono conseguenze per inaccettabili risultati, quindi non

abbiamo bisogno di un modello interpretabile e possiamo accettare qualsiasi black box. Infine, un altro aspetto rilevante è la "ragione" per cui è necessaria una spiegazione: un'interpretazione del modello può essere richiesto sia per rivelare i risultati nei dati che spiegano la decisione, sia per spiegare come la black box stessa funzioni. Queste ragioni altrettanto importanti e complementari richiedono differenti metodi di analisi [41].
Nell'analisi dell'interpretazione dei modelli predittivi, possiamo identificare un insieme di dimensioni prese in considerazione e che caratterizzano l'interpretazione del modello:

- **Interpretabilità globale e locale**: un modello può essere completamente interpretabile, ovvero siamo in grado di comprendere l'intera logica di un modello e seguire l'intero ragionamento che porta a tutti i diversi possibili esiti. In questo caso, stiamo parlando di interpretabilità globale. Invece, indichiamo con interpretabilità locale la situazione in cui è possibile comprendere solo le ragioni di una decisione specifica: solo la singola previsione e/o decisione è interpretabile.

- **Limitazione del tempo**: un aspetto importante è il tempo che l'utente è disponibile o è autorizzato a spendere sulla comprensione di una spiegazione. La disponibilità di tempo dell'utente è strettamente correlata allo scenario in cui il modello predittivo deve essere utilizzato. Pertanto, in alcuni contesti in cui l'utente deve rapidamente prendere la decisione (ad esempio, se un disastro è imminente), è preferibile avere una spiegazione semplice da capire. Mentre in contesti in cui il tempo di decisione non è un vincolo (ad esempio, durante una procedura per rilasciare un prestito) si potrebbe preferire una spiegazione più complessa ed esaustiva.

- **Natura delle competenze degli utenti**: gli utenti di un modello predittivo possono avere conoscenze di base diverse ed esperienza nel compito: decisori, scienziati, ingegneri della conformità e della sicurezza, scienza dei dati e così via. Conoscere l'esperienza dell'utente nell'attività è un aspetto chiave della percezione di interpretabilità di

un modello. Gli esperti di dominio potrebbero preferire un modello più ampio e sofisticato rispetto a uno più piccolo e talvolta più opaco [41].

La ***Black Box Explanation*** può essere suddivisa in tre grandi problemi: ***model explanation, model inspection e outcome explanation*** (figura 6.1).

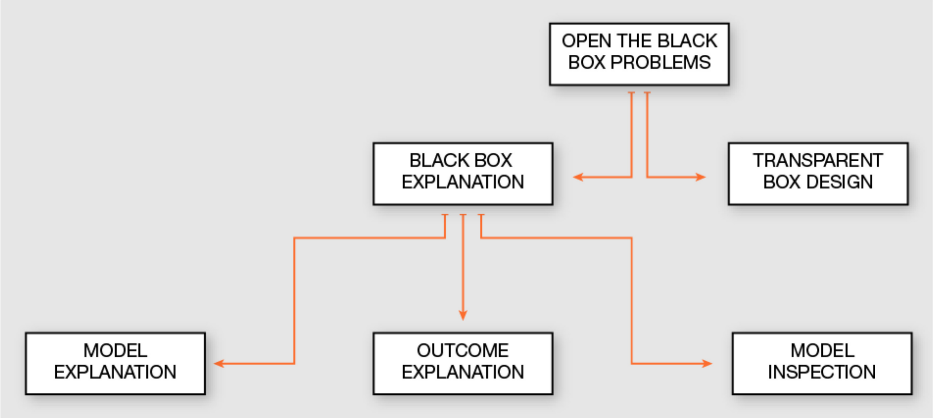


Figura 6.1: Tassonomia del problema nell'aprire una black box [41].

Il problema del model explanation consiste nel fornire una spiegazione globale del modello black box attraverso un modello interpretabile e trasparente. Questo modello dovrebbe essere in grado di imitare il comportamento della black box e dovrebbe anche essere comprensibile per l'uomo. In altre parole, il il modello interpretabile che si avvicina alla black box deve essere interpretabile a livello globale. Il problema del outcome explanation consiste nel fornire una spiegazione per l'esito della black box su quell'istanza. Non è necessario spiegare tutta la logica alla base della black box ma solo la ragione della previsione su un input specifico. Il problema del model inspection consiste nel fornire una rappresentazione (visiva o testuale) per alcune proprietà specifiche del modello della black box o delle sue previsioni. Alcune proprietà interessanti includono la sensibilità alle modifiche degli attributi e l'identificazione dei componenti del black box box (ad es. neuroni in ANN) responsabile di decisioni specifiche [41] .

## 6.1 Metodologia utilizzata

In questo lavoro sono state utilizzate due tecniche per due finalità diverse: il ***model inspection*** e per l'***outcome inspection***. Per il primo problema è stato costruito un algoritmo per la visualizzazione grafica delle attivazioni all'interno del nostro modello di rete neurale artificiale. In particolare, similmente alla tecnica dell'***Activation Maximization***, sono state visualizzate le attivazioni per osservare la formazione di pattern all'interno della rete. Per il secondo problema è stato utilizzato un ***Agnostic Explanator***. Un Agnostic Explanator implementa una funzione $f$ tale da poter spiegare qualsiasi tipo di black box. Questi approcci non restituiscono un predittore globale comprensibile specifico, quindi il tipo di spiegazione cambia rispetto alla funzione e al predittore globale. Per definizione, tutti questi approcci sono generalizzabili. In particolare è stato utilizzato LIME (Local Interpretable Model-agnostic Explanations) [42]. L'approccio di LIME ($f$) non dipende dal tipo di dati, né dal tipo di black box $b$ da aprire, né su un particolare tipo di predittore locale comprensibile $c_l$ o spiegazione $\epsilon_l$. L'intuizione principale di LIME è che la spiegazione può essere derivata localmente dai record generati casualmente nel vicinato del record da spiegare e ponderato in base alla sua vicinanza ad esso. In figura 6.2 si osserva il funzionamento di LIME. In breve, la funzione di decisione del modello black box $f$ (sconosciuta a LIME) è rappresentata dallo sfondo blu/rosa, che non può essere approssimato bene da un modello lineare. La croce rossa in grassetto è l'istanza spiegata. LIME campiona le istanze, ottiene le previsioni usando $f$ , che rappresenta la probabilità (o un indicatore binario) che l'istanza appartiene a una certa classe, e le pesa in base alla vicinanza all'istanza spiegata (rappresentata qui dalla dimensione). La linea tratteggiata è la spiegazione appresa che è fedele a livello locale (ma non a livello globale) [41][43].

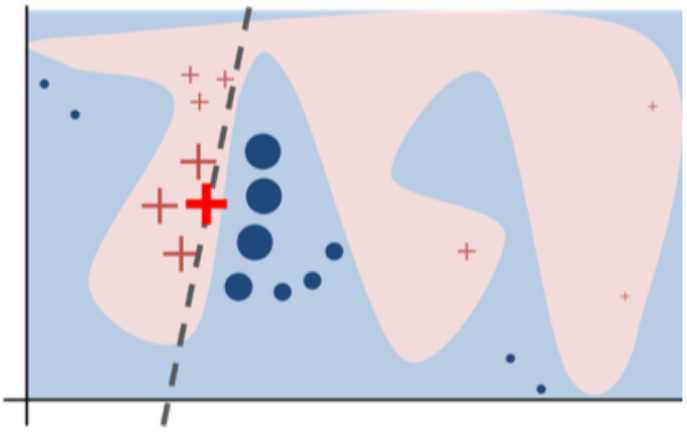

Figura 6.2: Rappresentazione del funzionamento di LIME

.

Nel nostro caso, è stato utilizzato il **sub-modular pick** di LIME. Infatti LIME mira ad attribuire la previsione di un modello a caratteristiche comprensibili all'uomo. A tal fine, è necessario eseguire il modello di spiegazione su un insieme diversificato ma rappresentativo set di istanze per restituire un set di spiegazioni non ridondante che è una rappresentazione globale del modello [43]. Pertanto, LIME seleziona una serie di istanze non ridondanti derivate dalle istanze campionate con un algoritmo chiamato "***sub-modular pick***" o **SP-LIME**. Una volta ottenuta la matrice di spiegazione dal passaggio precedente, dobbiamo derivare il vettore di importanza della funzione. I componenti di questo vettore danno l'importanza globale di ciascuna delle caratteristiche dalla matrice di spiegazione. La funzione precisa che associa la matrice di spiegazione al vettore di importanza dipende dal modello in esame, ma in tutti i casi dovrebbe restituire valori più alti (che significa maggiore importanza) per le funzionalità che spiegano più istanze, cioè le funzionalità trovate in più istanze a livello globale. Infine, SP-LIME trova solo il set minimo di istanze, in modo tale che non vi sia ridondanza nel set di istanze restituito finale. Perciò l'insieme di spiegazioni locali trovato dovrebbe coprire il numero massimo di funzioni del modello con poca o nessuna sovrapposizione tra le caratteristiche a cui si riferisce ogni singola spiegazione

locale. L'insieme minimo è scelto da un approccio avido che deve soddisfare un vincolo relativo al numero di casi che un soggetto umano è disposto a rivedere [44].

## 6.2 Motivazioni

Attraverso l'explainability ed in particolare il model inspection e l'outcome explanation, si possono analizzare in modo approfondito il comportamento dei nostri modelli. La motivazione principale è dovuta ai non ottimi risultati ottenuti sulla trasferibilità del modello. In particolare, vogliamo sia spiegare alcuni risultati ottenuti sia osservare se esistono di pattern di attivazioni all'interno della rete neurale artificiale che ci permettono di capire se è possibile un altro approccio ma soprattutto spiegare il comportamento interno della rete con un punto anomalo e un punto normale.
Un ulteriore analisi eseguita è sul tipo di anomalie che affliggono un nodo. Ciò verrà effettuato grazie a LIME ed in particolare al *sub-modular pick* che ci fornisce, come spiegato nella sezione precedente, le **features rilevanti di un risultato predetto di una determinata instanza**.

## 6.3 Analisi delle Attivazioni

Il modello che andremo ad esplorare, è il Modello Due mostrato nel capitolo 6. Questo modello è costruito, come abbiamo descritto nel capitolo precedente, da una parte dell'autoencoder, cioè l'encoder, per ridurre la dimensionalità dell'input ed estrarre le features rilevanti e da un classificatore. Le figure che verranno mostrate sono delle **colormap** dove sull'asse delle ascisse vi è il numero rappresentante un neurone artificiale, sull'asse delle ordinate abbiamo l'indice dei dati; il gradiente del colore rappresenta l'**intensità delle attivazioni** (si parte da un blu fino ad arrivare ad un rosso) e l'associazione colore-valore intensità è mostrato nella legenda a lato. Ogni figura è uno stack di colormap ed ogni stack rappresenta un layer.

### Nodo r183c15s03

Iniziamo con l'analisi del nodo r183c15s03, in particolare dell'encoder, facendo un confronto tra le attivazioni con i punti normali e con i punti anomali (consideriamo i veri positivi)(figure 6.3 e 6.4):

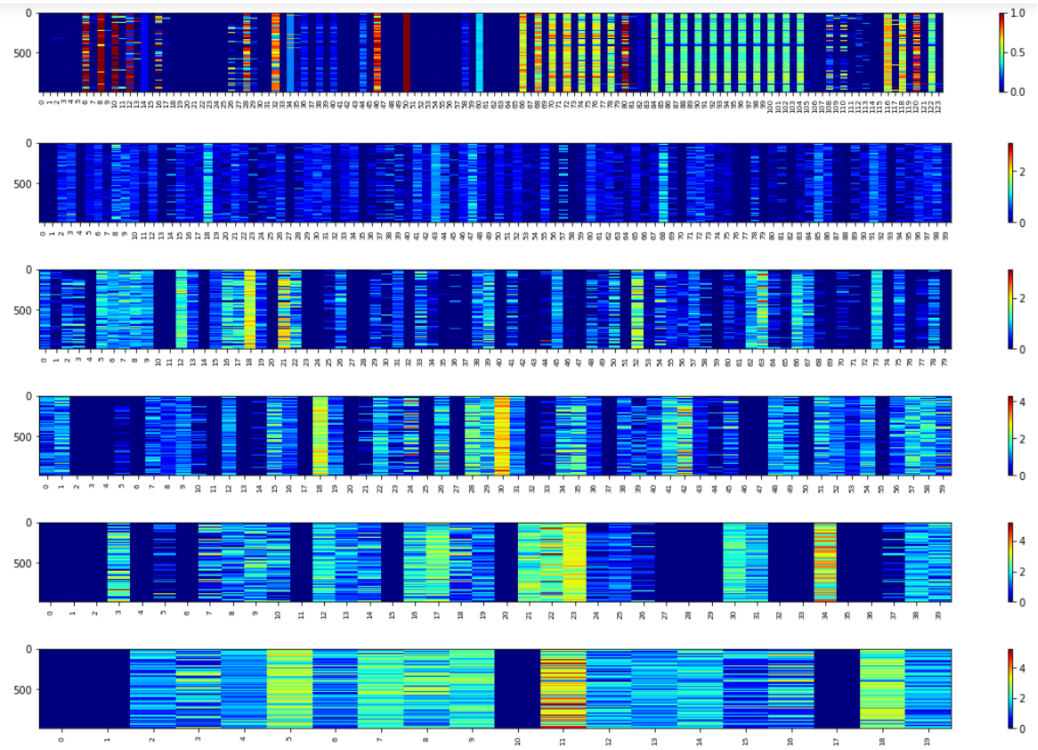

Figura 6.3: Attivazioni nodo r183c15s03 punti normali.

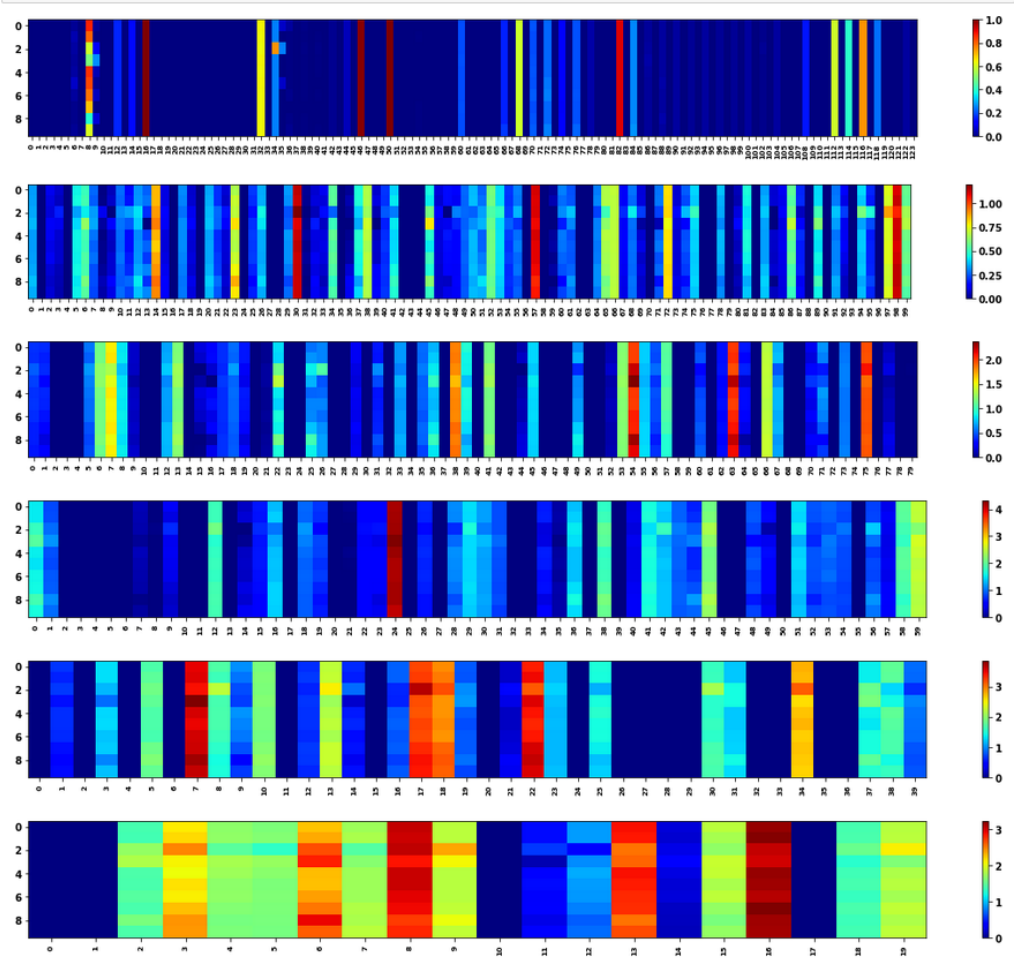

Figura 6.4: Attivazioni nodo r183c15s03 punti anomali.

La prima similarità che si osserva è la ripetizione di neuroni con un attivazione vicina allo 0: in particolare si osserva che nel layer finale (code layer) i neuroni 0,1,10 e 17 hanno un attivazione 0 sia con i punti normali, sia con i punti anomali. La stessa similarità si nota in altri layers. Osserviamo ora il comportamento con i falsi negativi (figura 6.5):

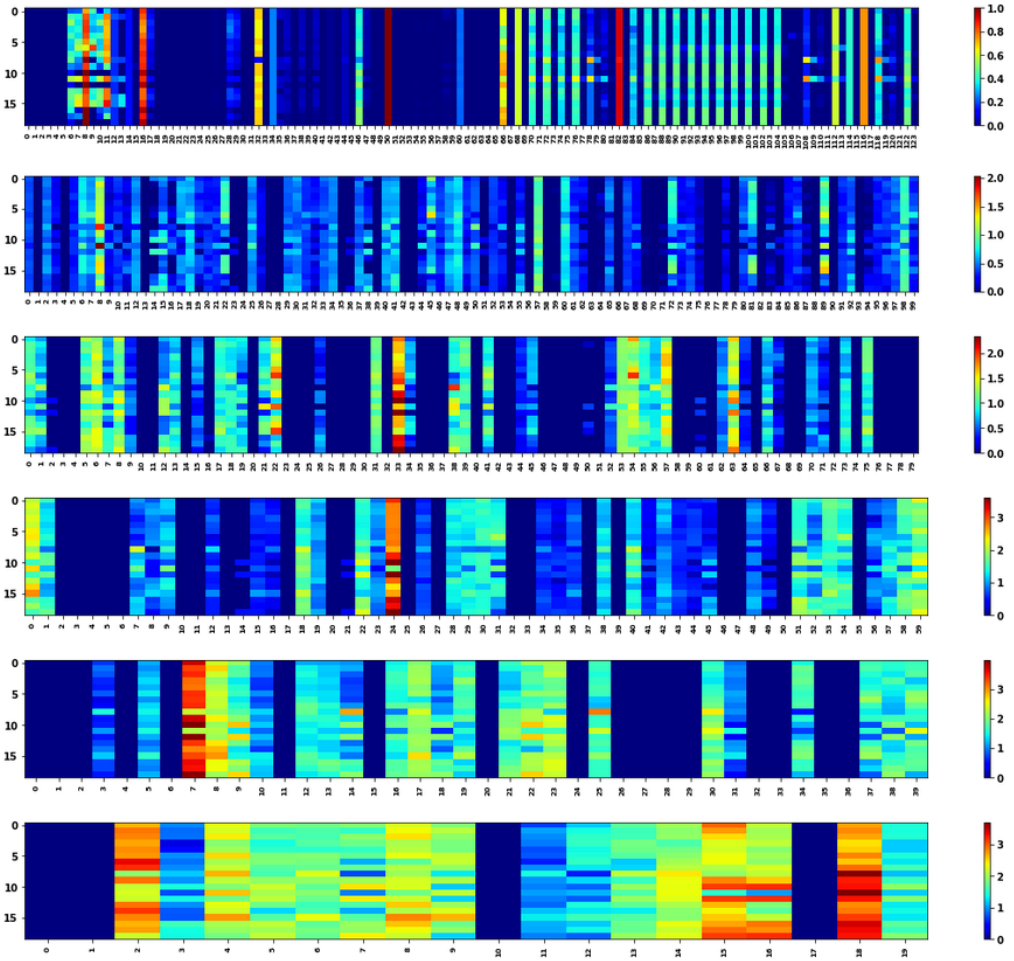

Figura 6.5: Attivazioni nodo r183c15s03 punti falsi negativi.

I falsi negativi presentano alcune variazioni in termini di valore delle attivazioni. Comunque il pattern si ripete: i neuroni attivi sono gli stessi dei punti normali e punti anomali. Osserviamo il comportamento del classificatore con i punti anomali, falsi negativi e normali.

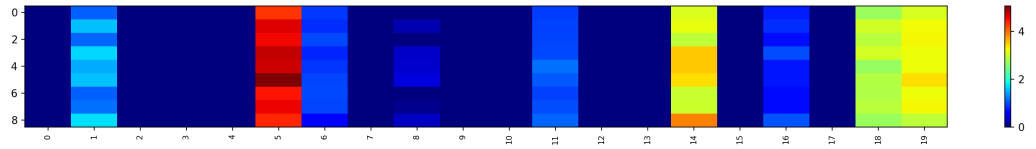

Figura 6.6: Attivazioni classificatore nodo r183c15s03 punti anomali.

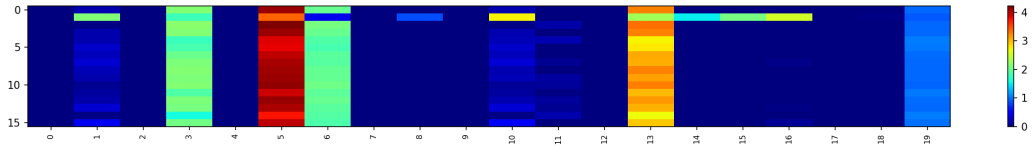

Figura 6.7: Attivazioni classificatore nodo r183c15s03 punti falsi negativi.

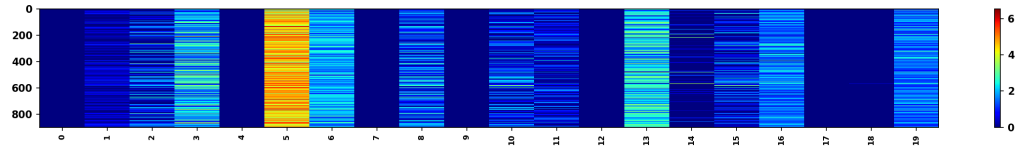

Figura 6.8: Attivazioni classificatore nodo r183c15s03 punti normali.

L'ultimo layer (l'output del classificatore) non è mostrato per ovvi motivi. Si nota come i falsi negativi seguono le attivazioni dei punti normali ad eccezione del neurone 5, il quale mostra una magnitudine maggiore sia nei punti anomali che nei punti falsi negativi.

### Nodo r183c12s04

Osserviamo ora le attivazioni del nodo r183c12s04, con cui abbiamo ottenuto ottimi risultati:

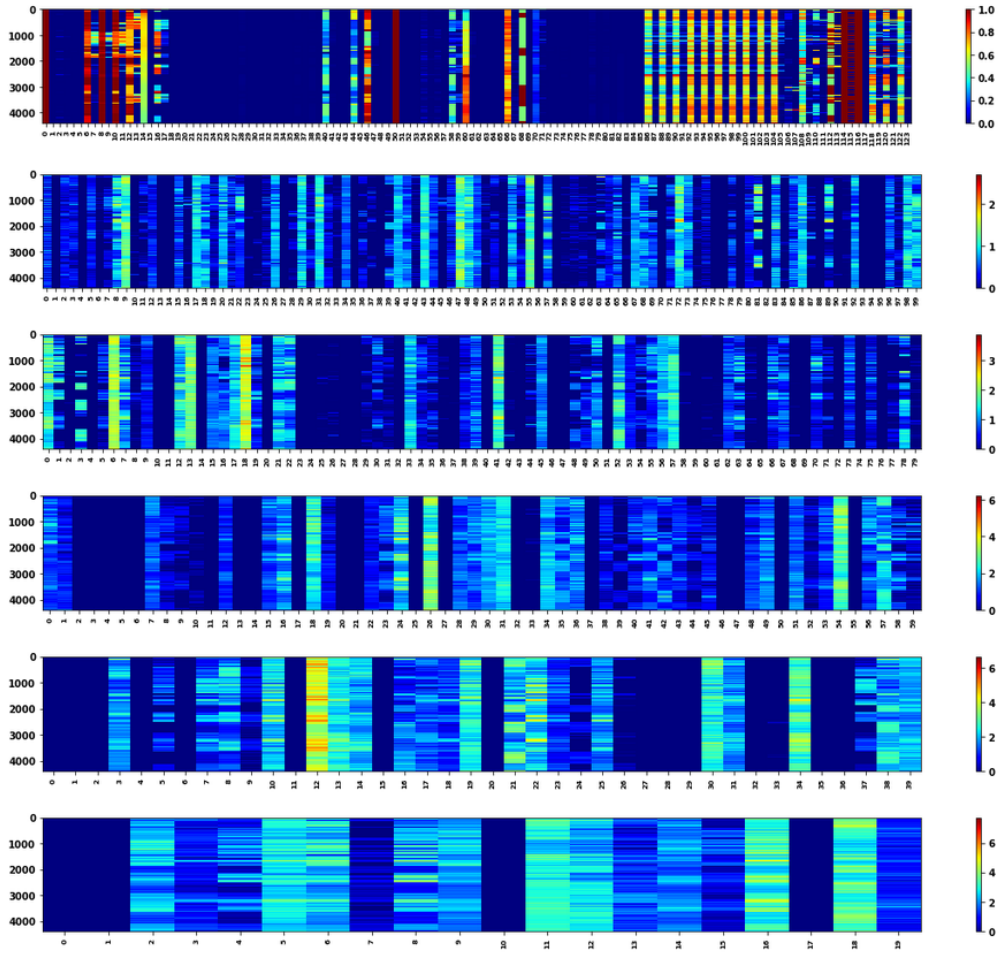

Figura 6.9: Attivazioni nodo r183c12s04 punti normali.

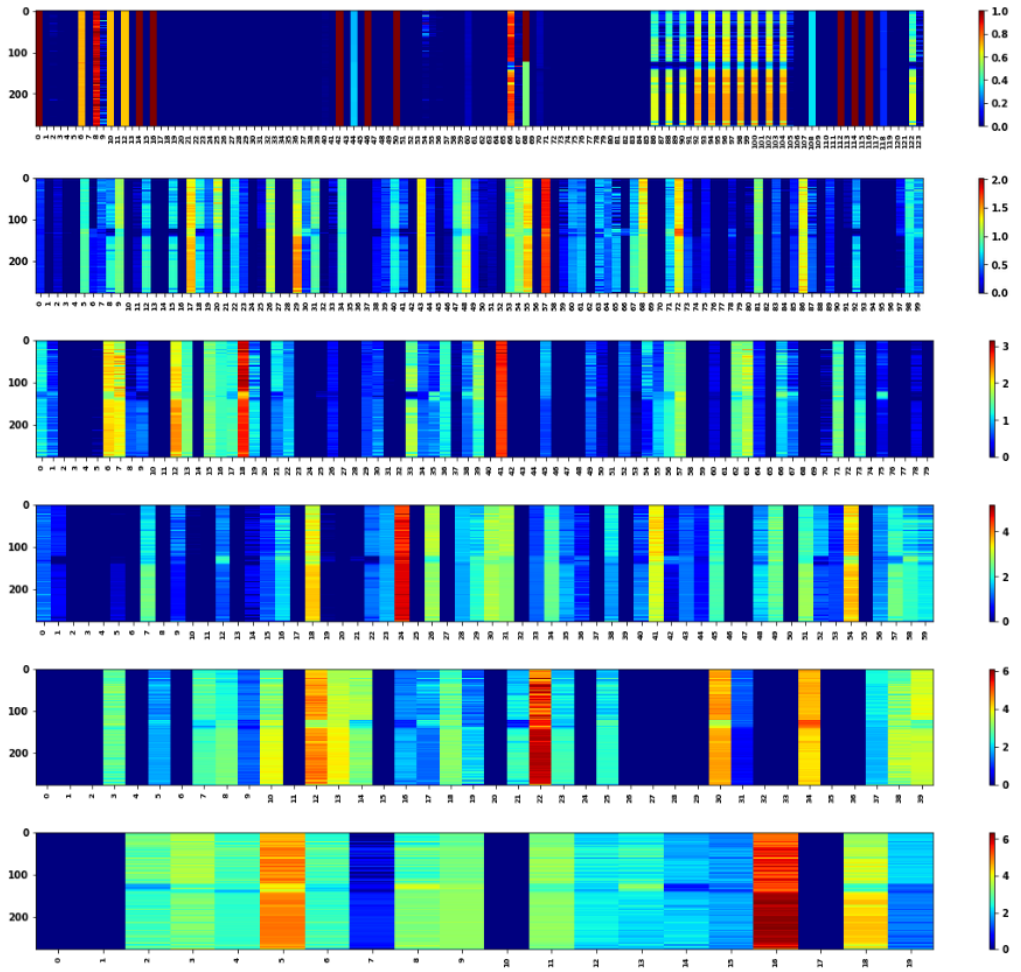

Figura 6.10: Attivazioni nodo r183c12s04 punti anomali.

Nelle attivazioni dei punti anomali, spiccano maggiormente le attivazioni dei neuroni 5 e 16. L'unica ripetizione che abbiamo è, quindi, del neurone 16 rispetto al nodo precedente. L'unico pattern ripetuto è quello dei neuroni con attivazione 0. Osserviamo ora il classificatore:

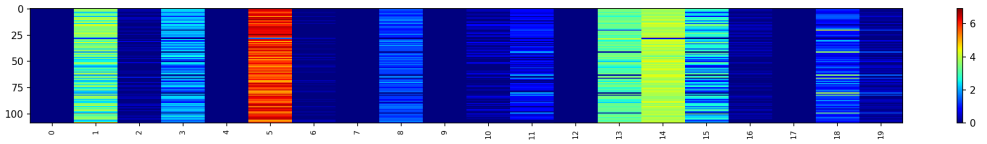


Figura 6.11: Attivazioni classificatore nodo r183c12s04 punti anomali.

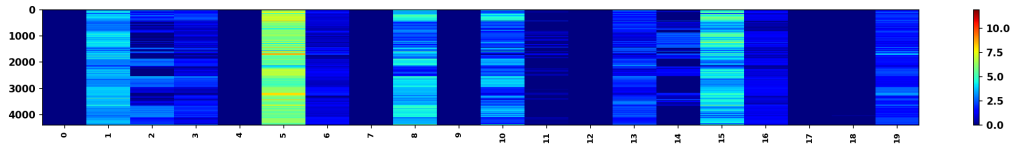


Figura 6.12: Attivazioni classificatore nodo r183c12s04 punti normali.

Anche in questo caso, come nel nodo precedente, i neuroni 5 e 14 sono fortemente attivi e determinanti per la classificazione poichè differiscono dalle attivazioni dei punti normali.

### Nodo r183c09s02

Osservando le attivazioni dell'encoder del nodo r183c09s02 (figure 6.13, 6.14, 6.15), si nota come il pattern dei neuroni con attivazione 0 è ripetuto in tutti i layer come nei nodi precedenti. Tuttavia, nell'ultimo layer notiamo la ripetizione, tra i neuroni attivi, solo del 16°. Nel penultimo layer si osserva che, invece, il nodo 33 è attivo come negli altri nodi. Nel classificatore, invece, nelle attivazioni dei punti anomali (figura 6.17), i neuroni fondamentali sono il 5, 14, 18 e 19. Le attivazioni di questi neuroni sono simile al nodo r183c15s03 ed in particolare, le attivazioni del quinto neurone è visibile anche nel nodo r183c12s04. Infine, le attivazioni dei falsi negativi nel classificatore (figura 6.18) risultano essere una via di mezzo tra quelle dei punti anomali e quelle dei punti normali. I falsi negativi, comunque, non presentano pattern evidenti.

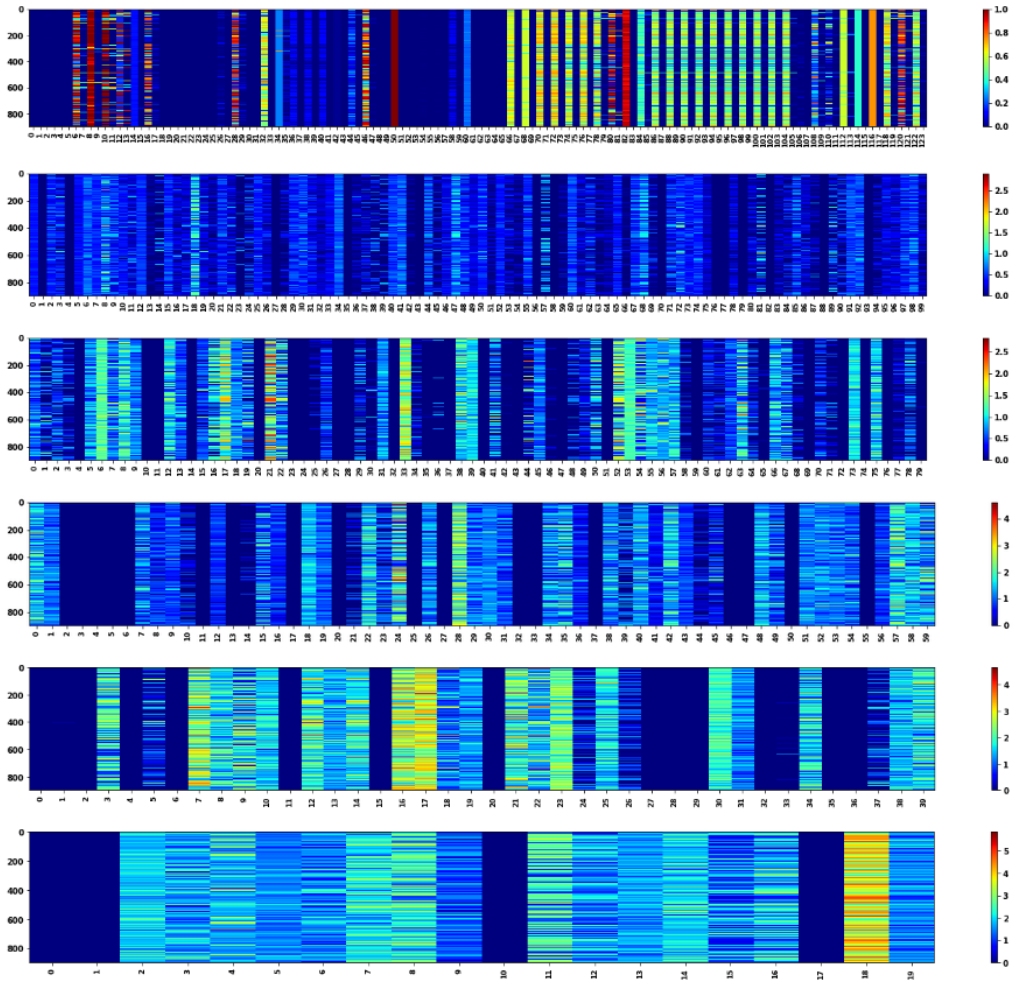

Figura 6.13: Attivazioni Encoder nodo r183c09s02 punti normali.

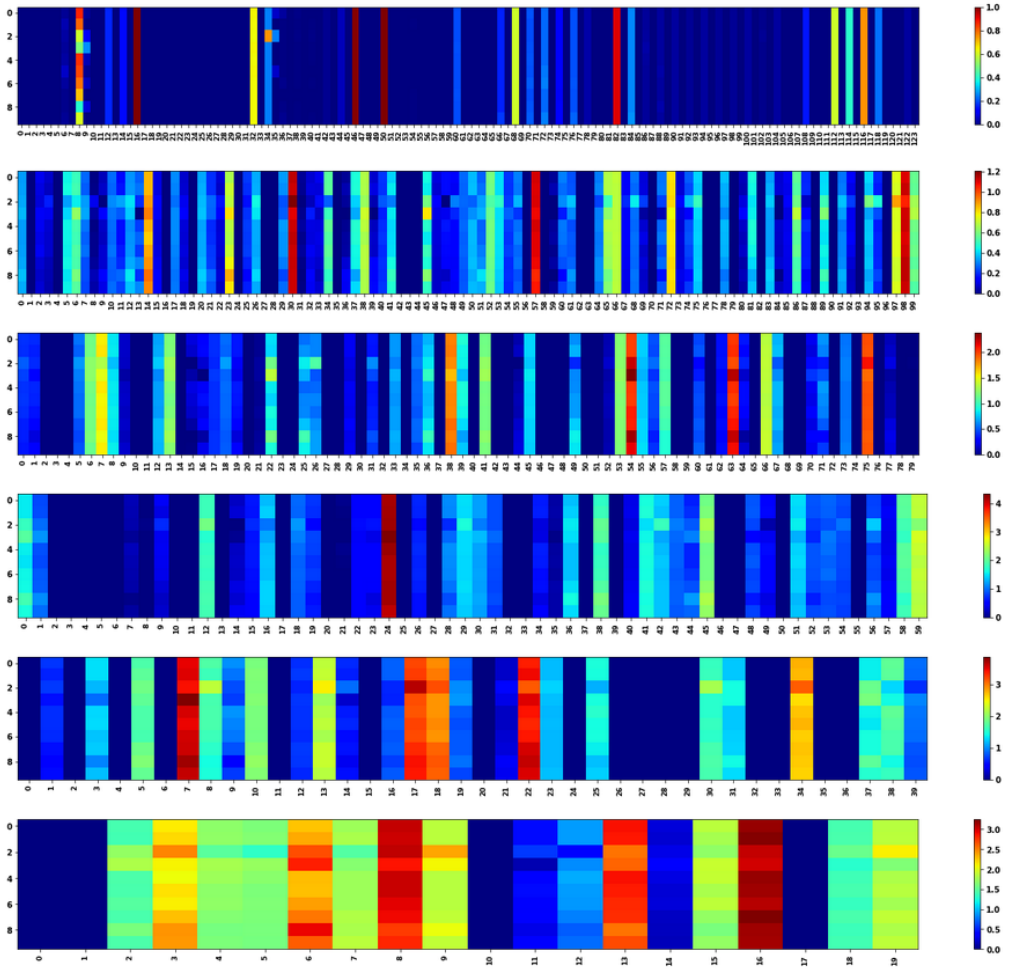

Figura 6.14: Attivazioni Encoder nodo r183c09s02 punti anomali.

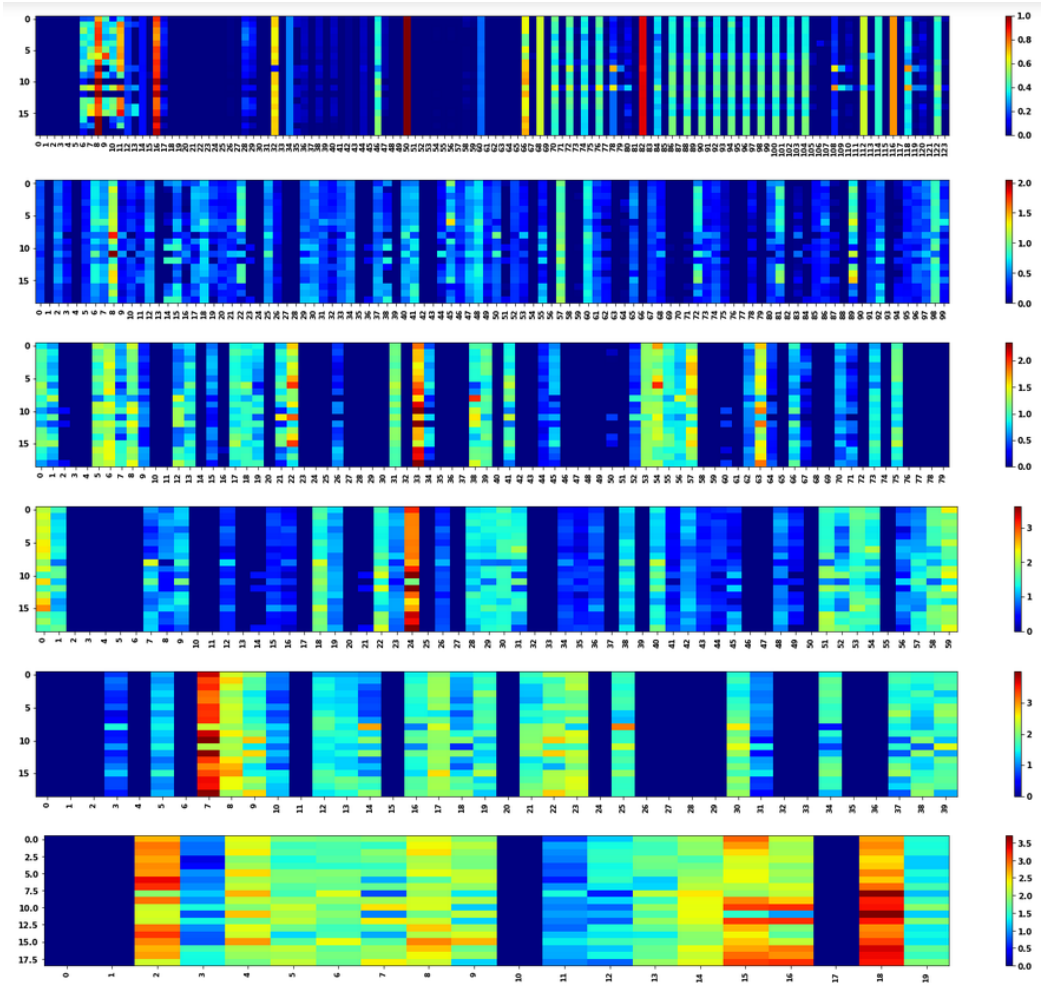

Figura 6.15: Attivazioni Encoder nodo r183c09s02 punti falsi negativi.

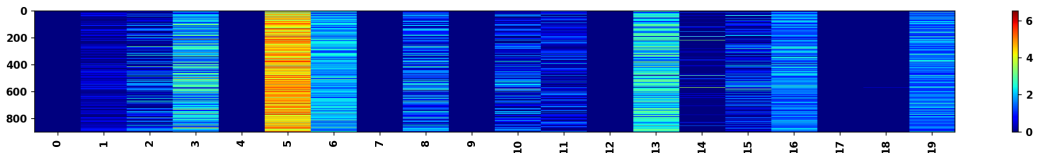

Figura 6.16: Attivazioni Classificatore nodo r183c09s02 punti normali.

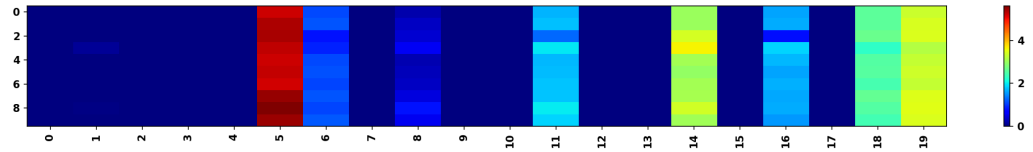


Figura 6.17: Attivazioni Classificatore nodo r183c09s02 punti anomali.

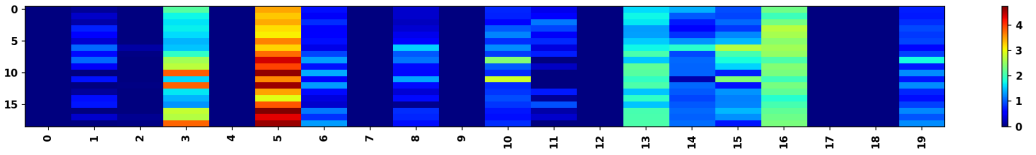


Figura 6.18: Attivazioni Classificatore nodo r183c09s02 punti falsi negativi.

### Nodi r183c13s01 e r183c16s03

I pattern di attivazione, osservati nei nodi descritti nelle sezioni precedenti, sono visibili anche nei nodi r183c13s01 e r183c16s03 come mostrato nelle figure sottostanti.

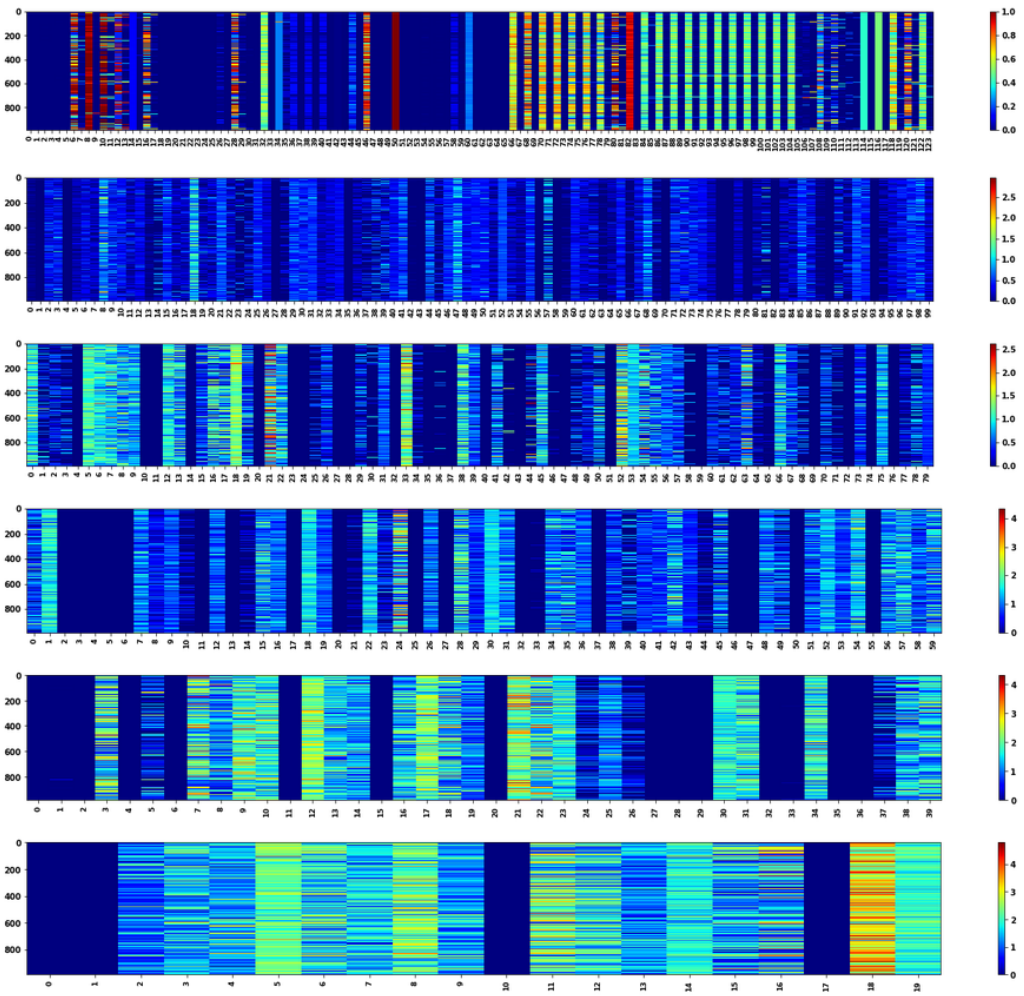

Figura 6.19: Attivazioni Encoder nodo r183c13s01 punti normali.

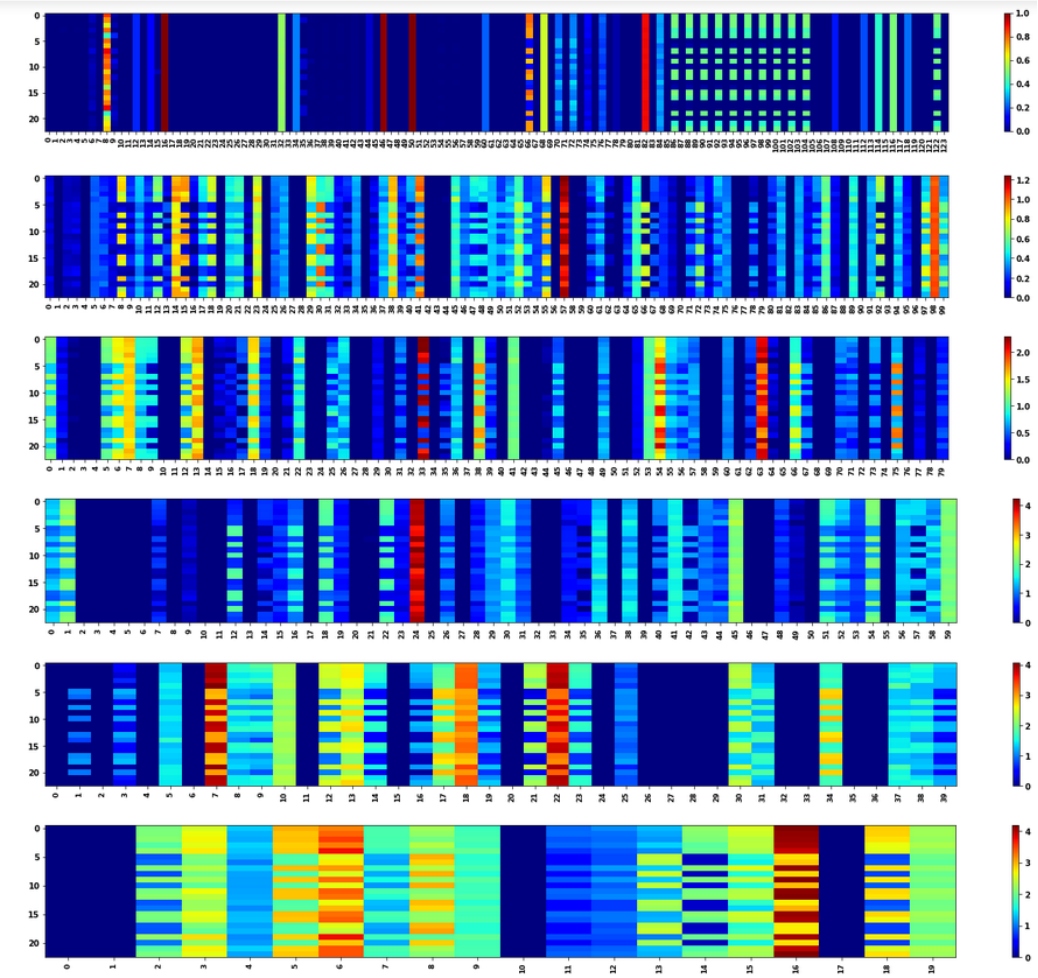

Figura 6.20: Attivazioni Encoder nodo r183c13s01 punti anomali.

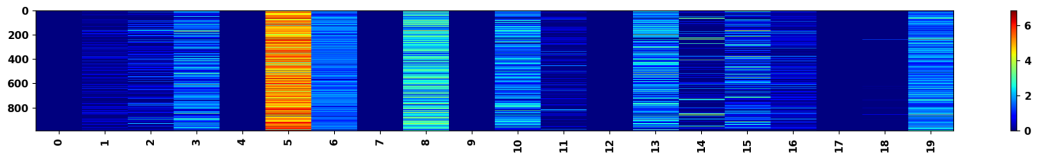

Figura 6.21: Attivazioni Classificatore nodo r183c13s01 punti normali.

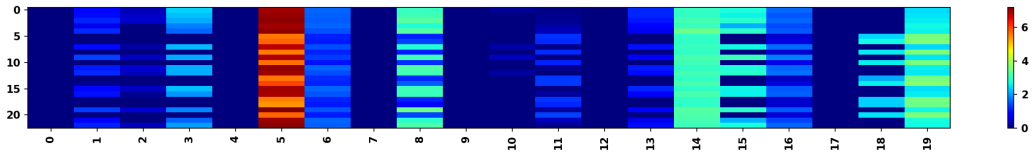

Figura 6.22: Attivazioni Classificatore nodo r183c13s01 punti anomali.

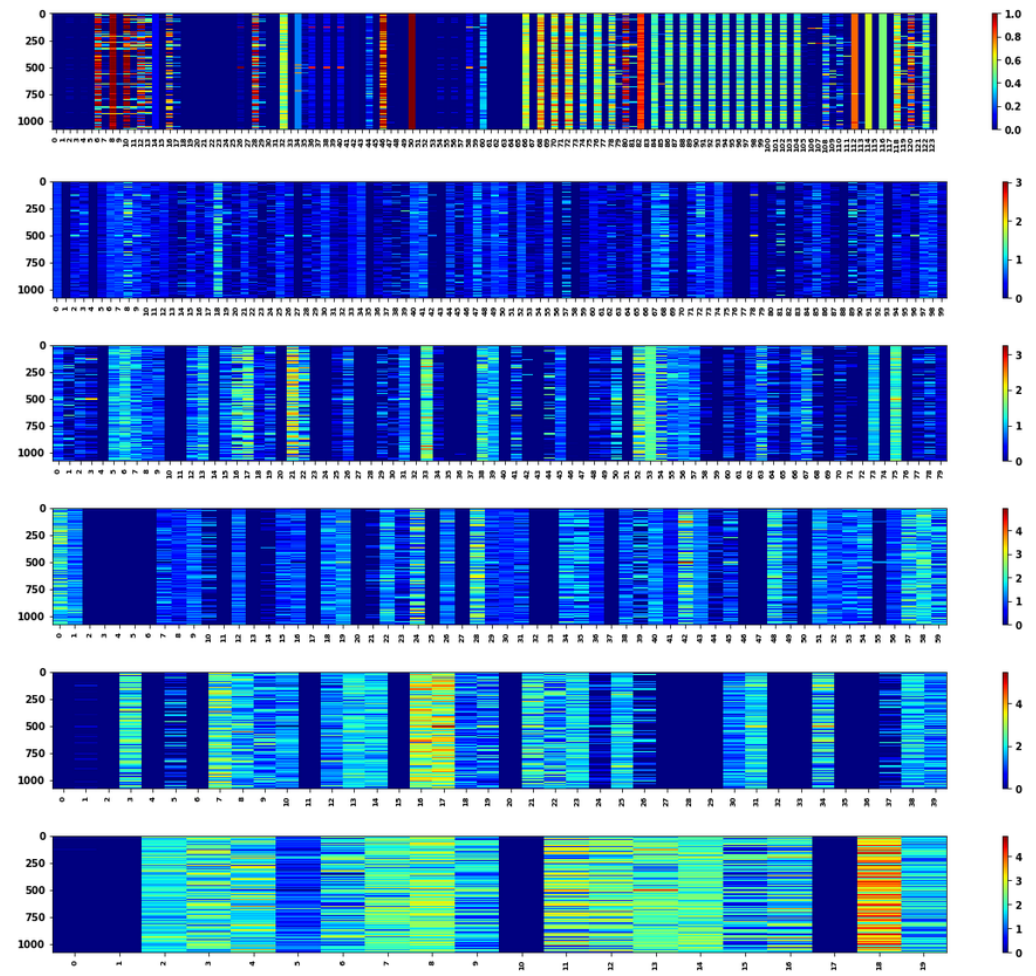

Figura 6.23: Attivazioni Encoder nodo r183c16s03 punti normali.

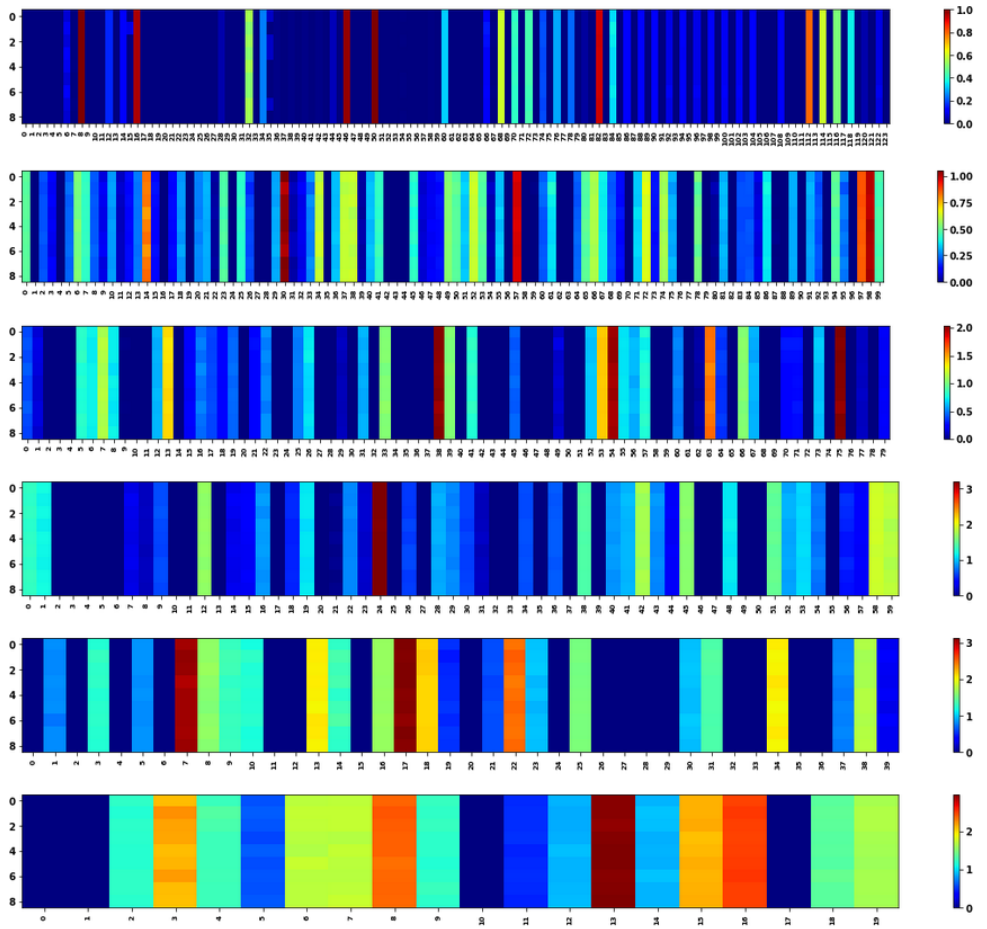

Figura 6.24: Attivazioni Encoder nodo r183c16s03 punti anomali.

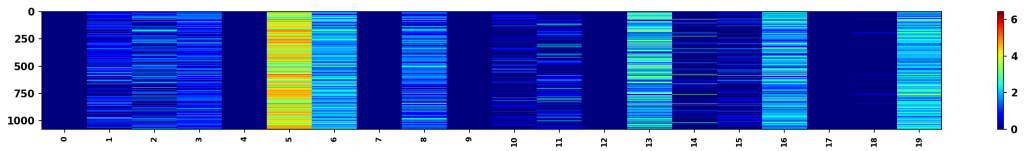

Figura 6.25: Attivazioni Classificatore nodo r183c16s03 punti normali.

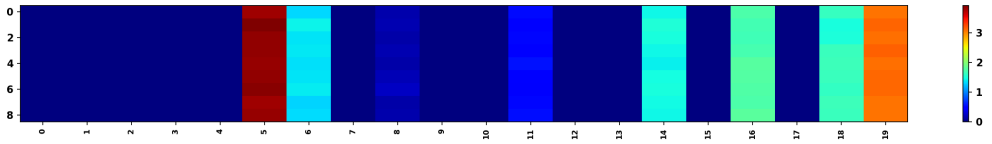

Figura 6.26: Attivazioni Classificatore nodo r183c16s03 punti anomali.

### Nodi r183c10s04 e r183c16s04

I nodo r183c10s04 e r183c16s04 presentano un periodo molto anomalo. Nelle prossime figure sono rappresentati le attivazioni dell'encoder e del classificatore: non si notano pattern di attivazione come nei precedenti nodi. Si osservano solo alcune eccezioni nel code layer degli encoder e nel classificatore. Infatti il neurone 19 è leggermente attivo in presenza di anomalie nell'encoder di entrambi i nodi. Per quanto riguarda il classificatore, si nota l'attivazione del neurone 5, presente anche negli altri nodi. I falsi negativi, invece, presentano le stesse attivazioni dei punti anomali, soprattutto nel nodo r183c16s04.

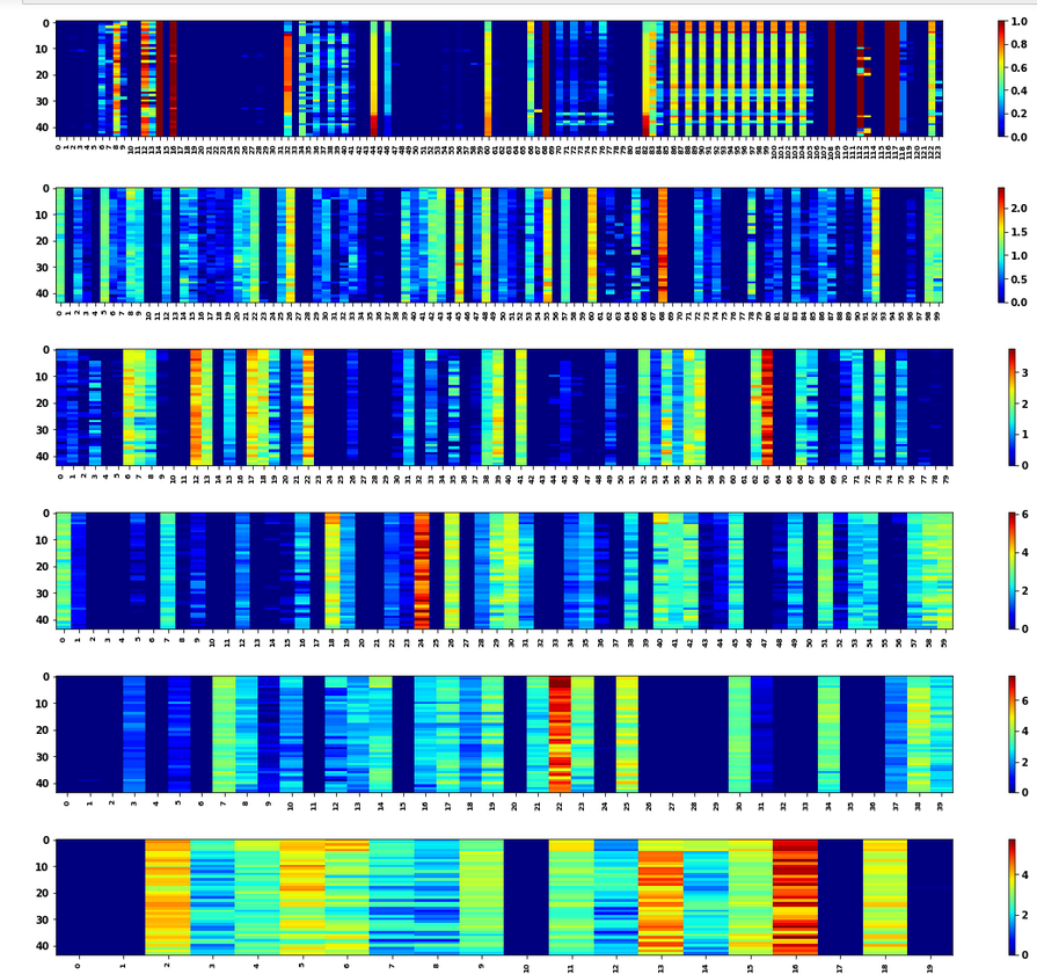

Figura 6.27: Attivazioni Encoder nodo r183c10s04 punti anomali.

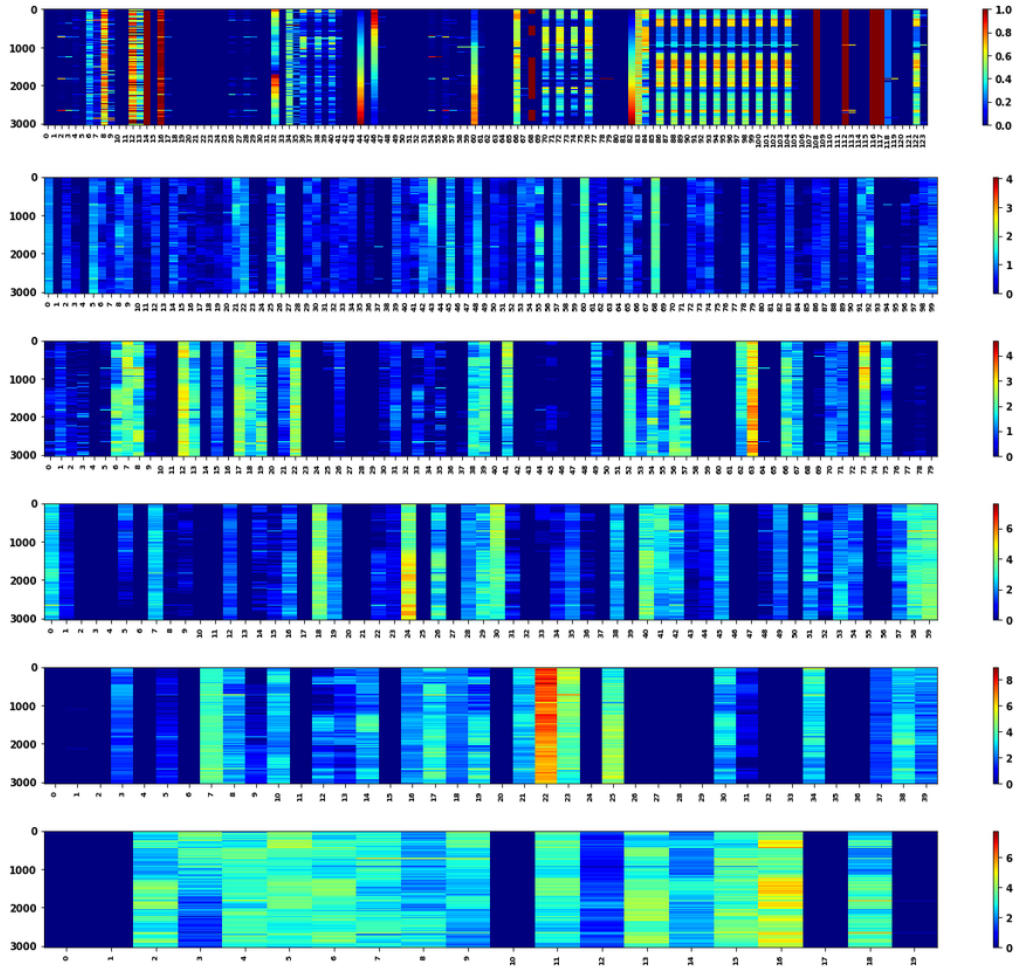

Figura 6.28: Attivazioni Encoder nodo r183c10s04 punti falsi negativi.

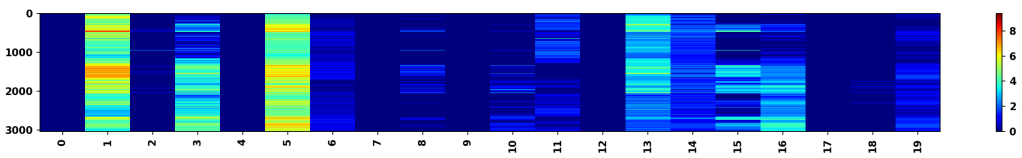

Figura 6.29: Attivazioni Classificatore nodo r183c10s04 punti falsi negativi.

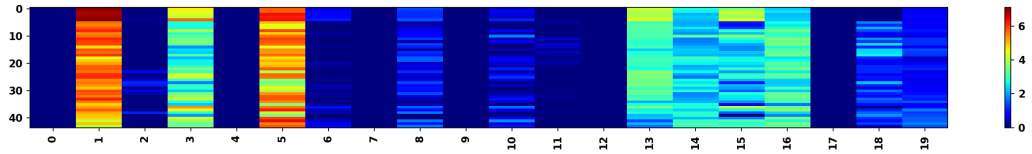

Figura 6.30: Attivazioni Classificatore nodo r183c10s04 punti anomali.

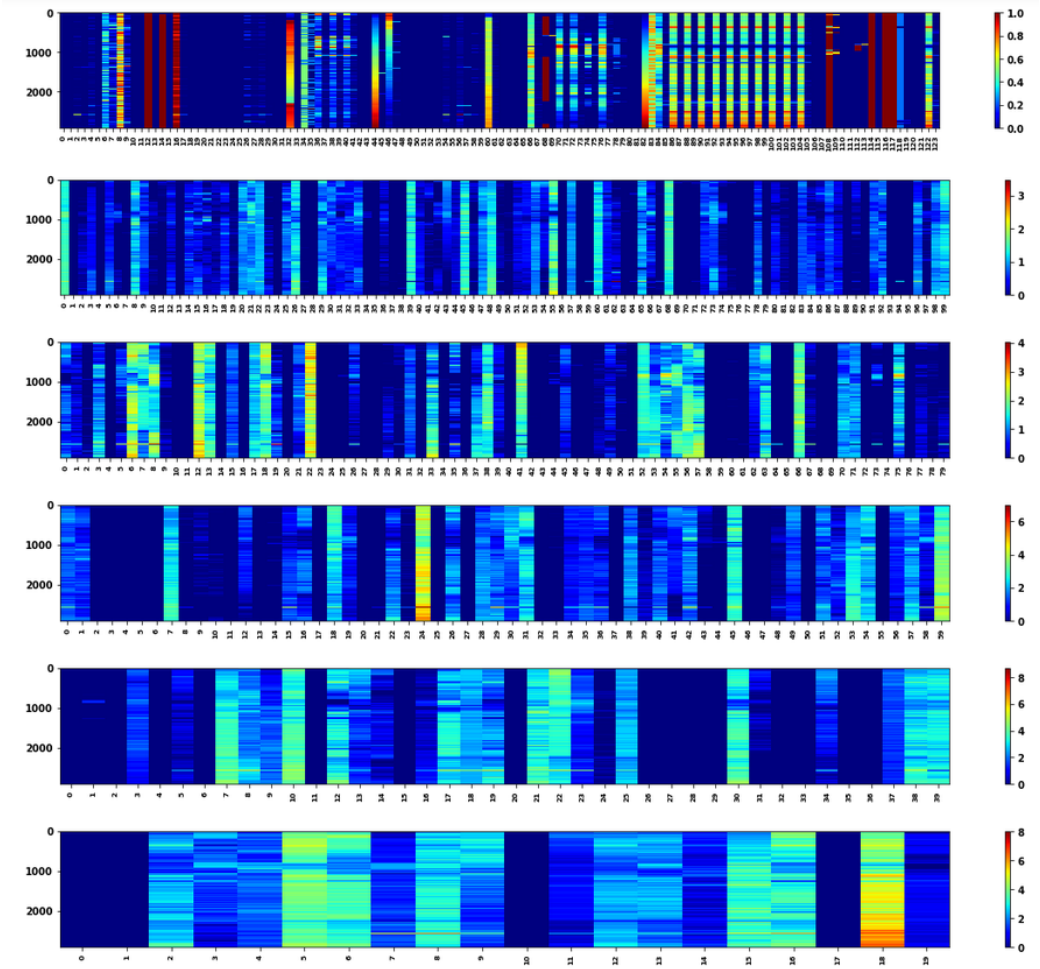

Figura 6.31: Attivazioni Encoder nodo r183c16s04 punti falsi negativi.

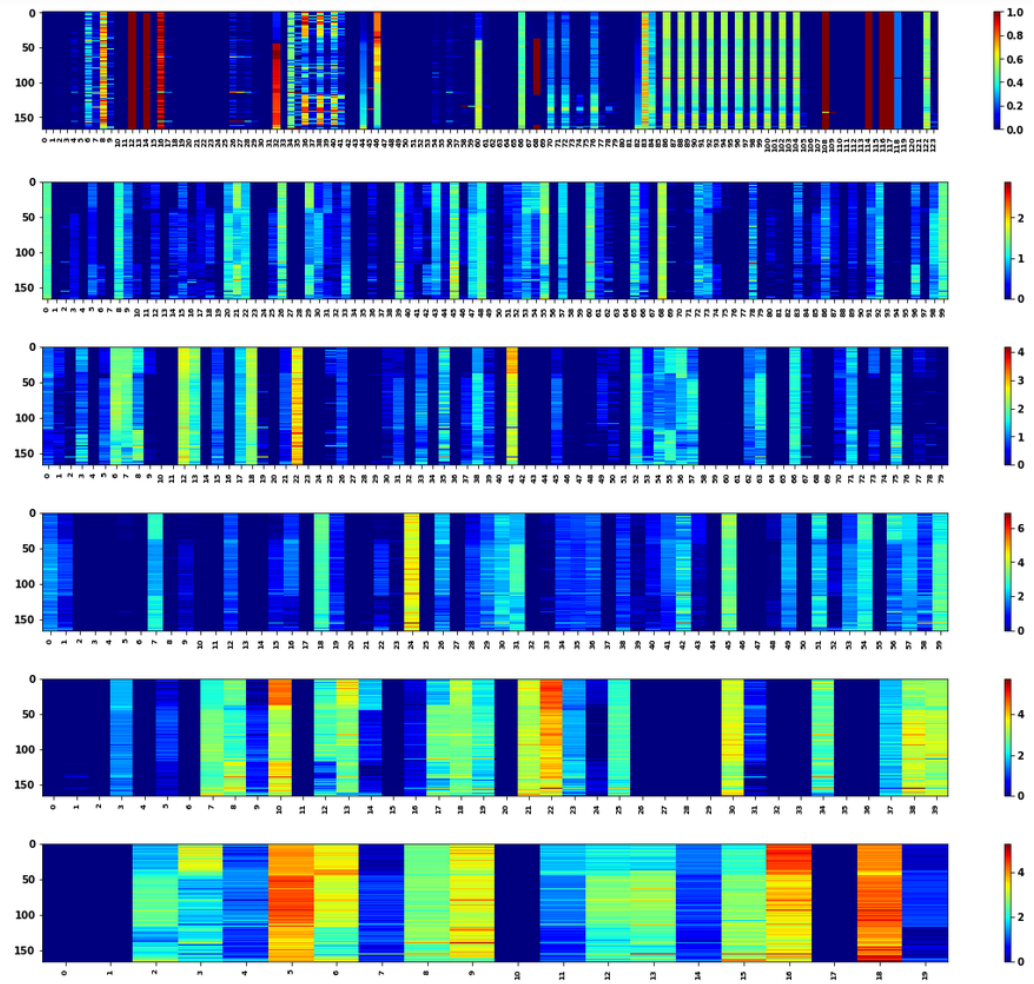

Figura 6.32: Attivazioni Encoder nodo r183c16s04 punti anomali.

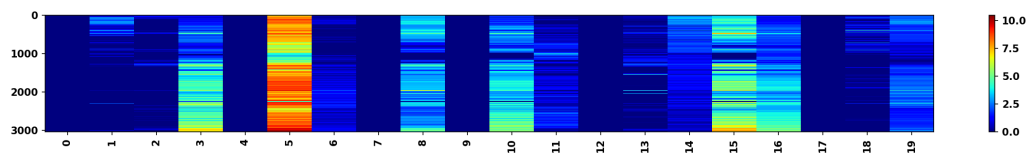

Figura 6.33: Attivazioni Classificatore nodo r183c16s04 punti falsi negativi.

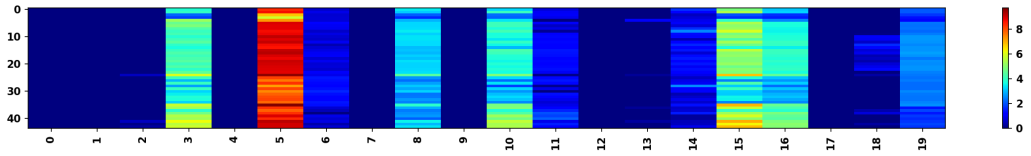


Figura 6.34: Attivazioni Classificatore nodo r183c16s04 punti anomali.

### 6.3.1 Considerazioni

Attraverso il metodo realizzato in questo lavoro, sono state esplorate le attivazioni dei modelli utilizzati per il Transfer Learning. Il risultato complessivo è che, all'interno dei modelli**, esistono alcuni pattern delle attivazioni**. Infatti, osservando la figura 6.35 che mostra una panoramica delle attivazioni degli encoder dei nodi analizzati, notiamo che nella predizione delle anomalie, il contributo dei neuroni, evidenziati da un cerchio rosso nella figura, si ripete in ogni nodo. Di conseguenza, le anomalie sembrano essere tutte della stessa origine quando vengono individuate poichè attivano lo stesso numero di neuroni artificiali (ciò verrà confermato nella prossima sezione, utilizzando LIME). Per quanto riguarda le anomalie dei nodi r183c10s04 e r183c16s04, nelle figure 6.27 e 6.32 si nota come questo pattern di attivazione non si ripete in modo evidente. Pertanto, l'autoencoder non riesce a cattuare le features rilevanti di questi due nodi inducendo ad un errore di classificazione in fase di predizione. Inoltre, il pattern di attivazioni suggerisce che la strada dell'apprendimento semi-supervisionato attraverso gli autoencoder (sezione 5.6.2) o attraverso un apprendimento non supervisionato potrebbe essere un approccio alternativo valido e funzionale. In un approccio non supervisionato o semi-supervisionato potrebbe essere possibile, osservando i pattern di attivazione dei neuroni di una rete neurale artificiale, individuare non solo l'esistenza di una sola ma anche di molteplici tipi di anomalie.

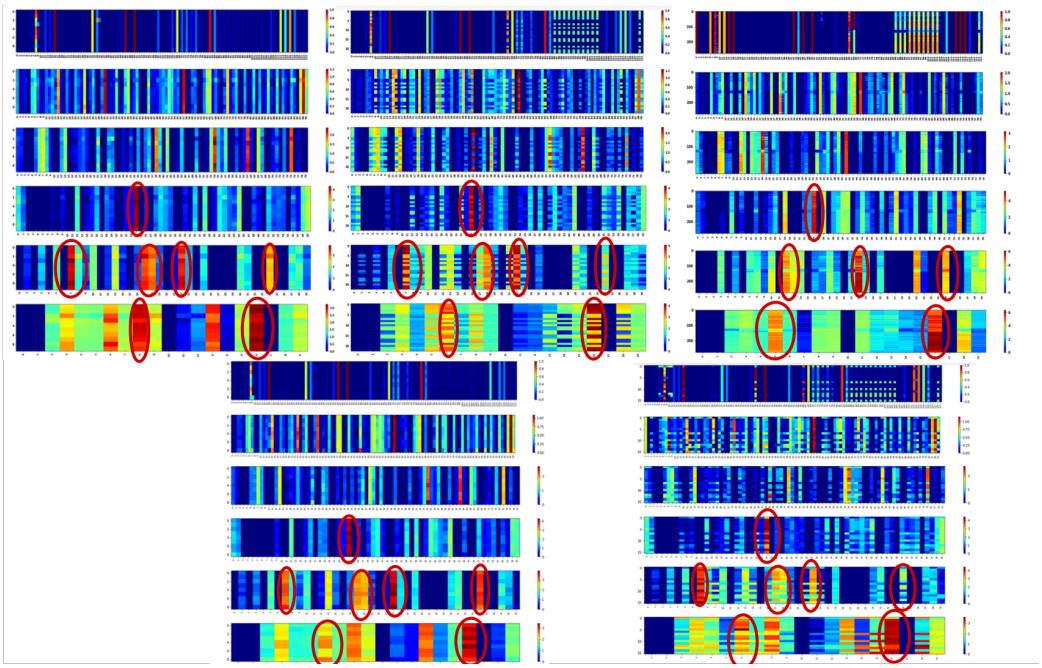

Figura 6.35: Panoramica delle attivazioni dell'encoder dei nodi analizzati.

## 6.4 Analisi dei risultati mediante LIME

L'algoritmo sub-modular pick, come descritto nell'introduzione di questo capitolo, ci fornisce le features rilevanti che supportano un risultato. Pertanto, verranno mostrate le features che sono state determinanti per le predizione dei punti punti anomali di alcuni nodi. I grafici che verranno mostrati sono degli istogrammi in cui ogni barra verticale è il risultato del sub-modular pick (ossia il peso della feature per quella predizione) mediato su tutte le instanze sotto analisi, per l'occorrenza nelle instanze (ad esempio se una feature ha un peso medio di 0.4 in 100 predizioni, il risultato è 0.4∗100) in modo da eliminare le features meno rilevanti. I nodi analizzati sono: r183c09s01, r183c09s02, r183c09s03, r183c09s04, r183c10s01, r183c10s02, r183c10s03, r183c11s01, r183c11s03,r183c11s04, r183c12s02 e r183c13s04.

### 6.4.1 Punti Anomali

Osservando le figure, si nota la netta prevalenza della feature ***"avg:cpu_idle"*** in quasi tutti i nodi. La feature si riferisce allo stato di idle del nodo. Infatti, i modelli sono stati addestrati su set di dati il cui stato anomalo è un etichetta data dal plugin Nagios, in particolare allo stato DOWN+DRAIN. Questa condizione fornisce uno stato di anomalia del nodo ma anche che è stato messo offline, quindi irrangiungibile agli utenti. Il modello è stato, quindi, addestrato su questo particolare stato anomalo che viene rilevato per la maggior parte dei nodi. Non mancano, tuttavia, altri tipi di anomalie rilevate. In particolare, si nota come la maggior parte delle feature che supportano una predizione di anomalia si riferiscono al funzionamento del sistema di raffraddemaneto (rappresentato dalle features *"avg:Fan1A_Tech", "avg:Fan2A_Tech", "avg:Fan2B_Tech","avg:System_Air_Flow"* e molte altre), al funzionamento dell cpu (*"avg:freq_core_max","avg:CPU_POWER"* e molte altre ), alla memoria e al disco (*"var:mem_free"* e *"var:disk_free"*).

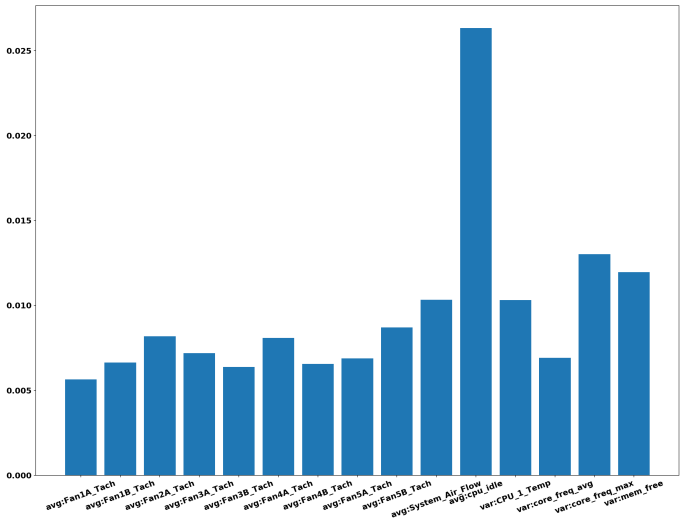


Figura 6.36: Features rilevanti nei punti anomali per il nodo r193c09s01

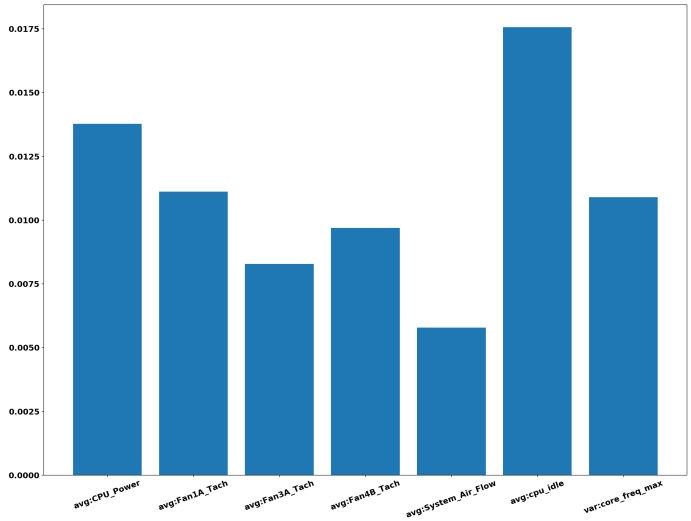


Figura 6.37: Features rilevanti nei punti anomali per il nodo r193c09s02

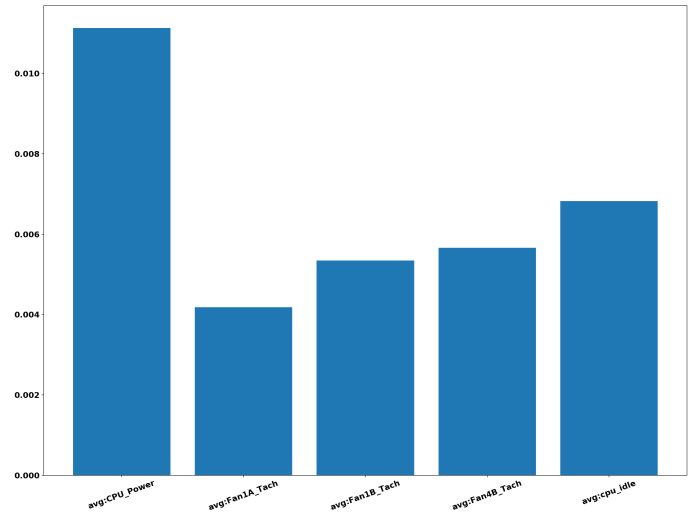


Figura 6.38: Features rilevanti nei punti anomali per il nodo r193c09s03

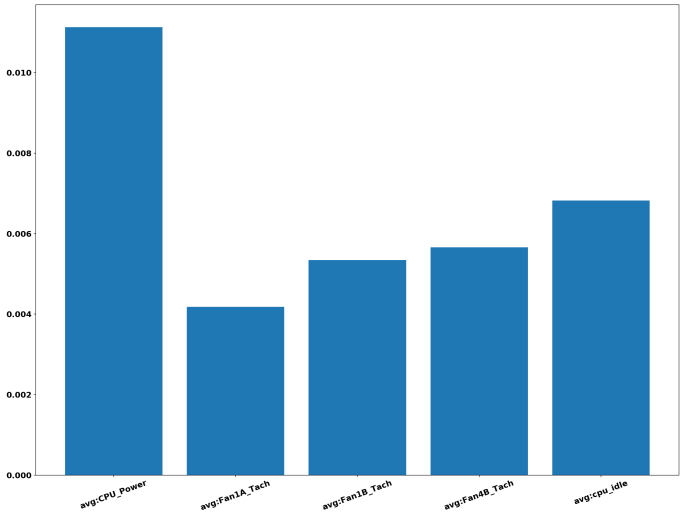


Figura 6.39: Features rilevanti nei punti anomali per il nodo r193c09s03

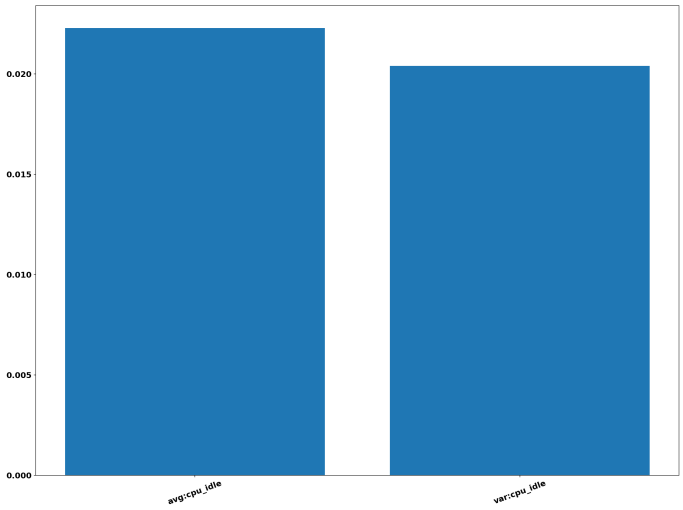


Figura 6.40: Features rilevanti nei punti anomali per il nodo r193c09s04

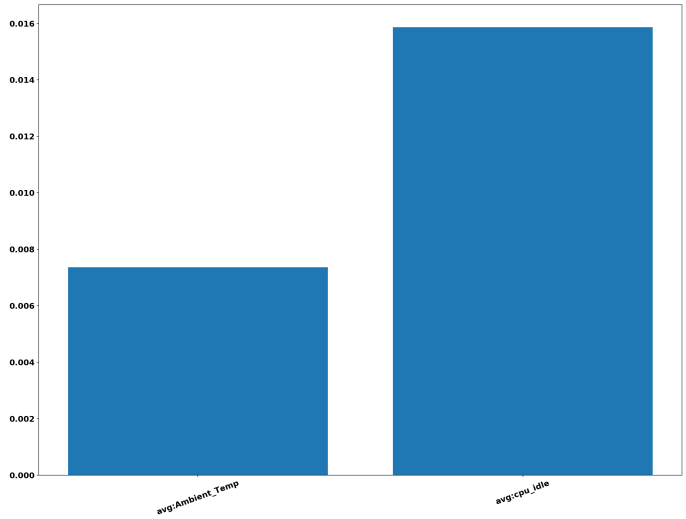


Figura 6.41: Features rilevanti nei punti anomali per il nodo r193c10s01

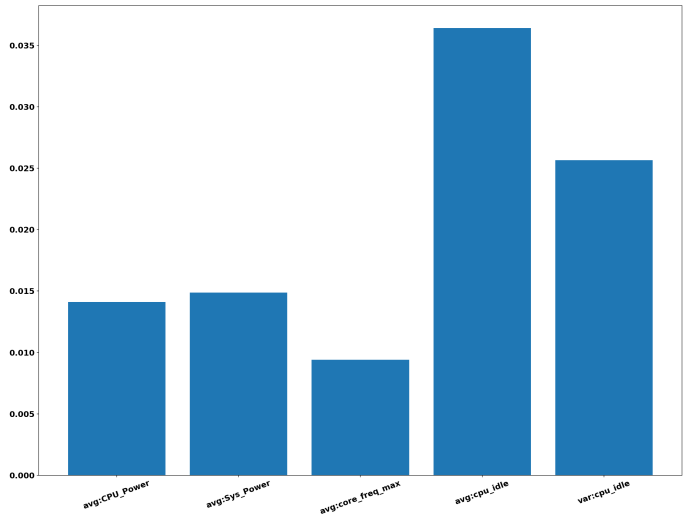


Figura 6.42: Features rilevanti nei punti anomali per il nodo r193c10s02

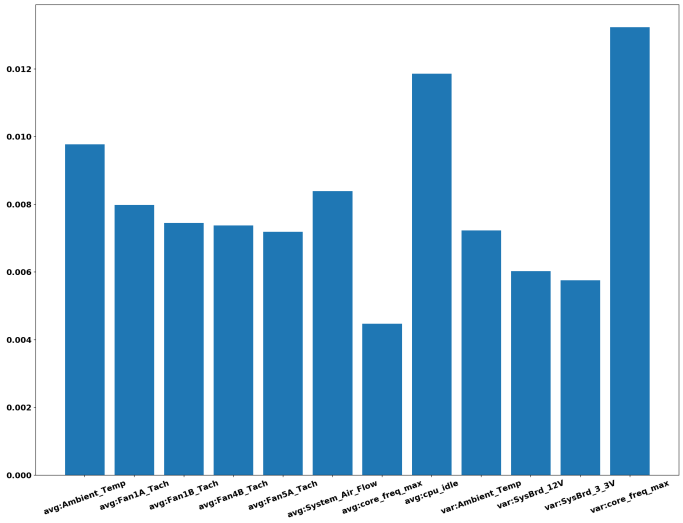


Figura 6.43: Features rilevanti nei punti anomali per il nodo r193c10s03

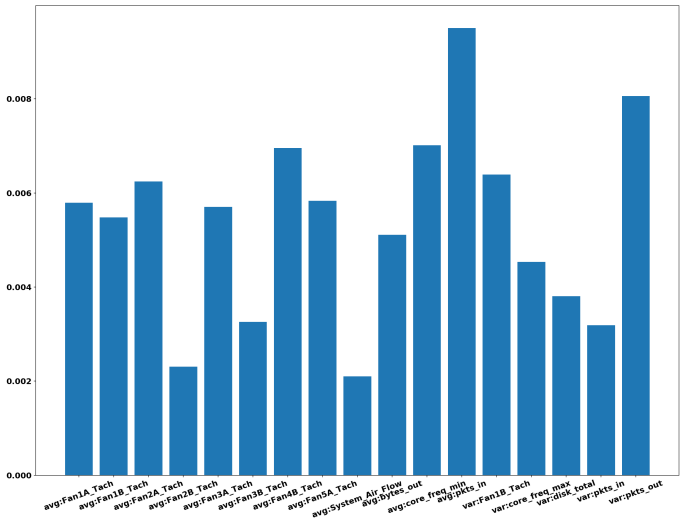


Figura 6.44: Features rilevanti nei punti anomali per il nodo r193c11s01

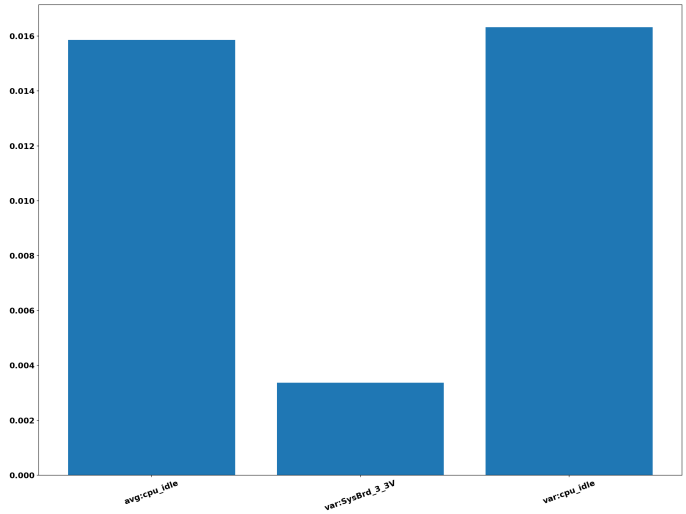


Figura 6.45: Features rilevanti nei punti anomali per il nodo r193c11s03

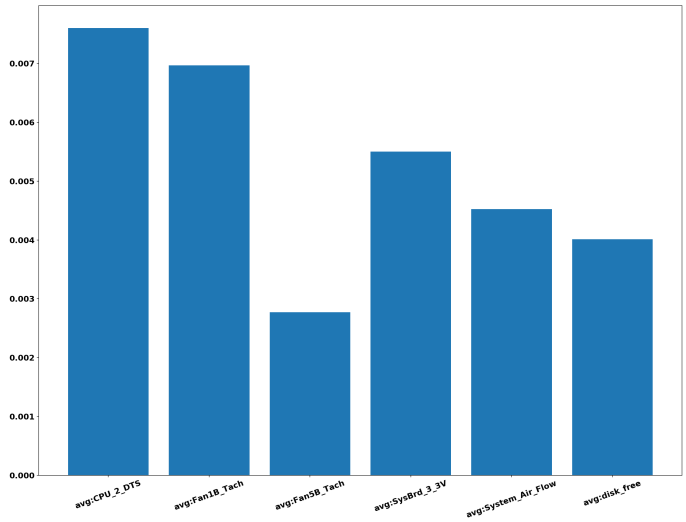


Figura 6.46: Features rilevanti nei punti anomali per il nodo r193c11s04

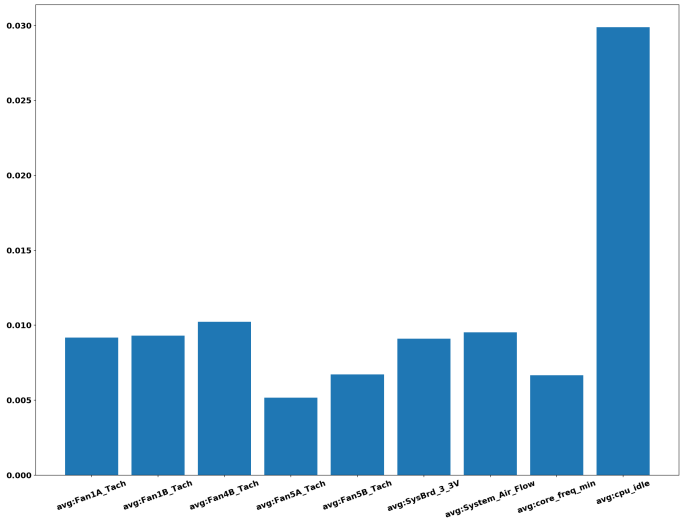


Figura 6.47: Features rilevanti nei punti anomali per il nodo r193c12s02

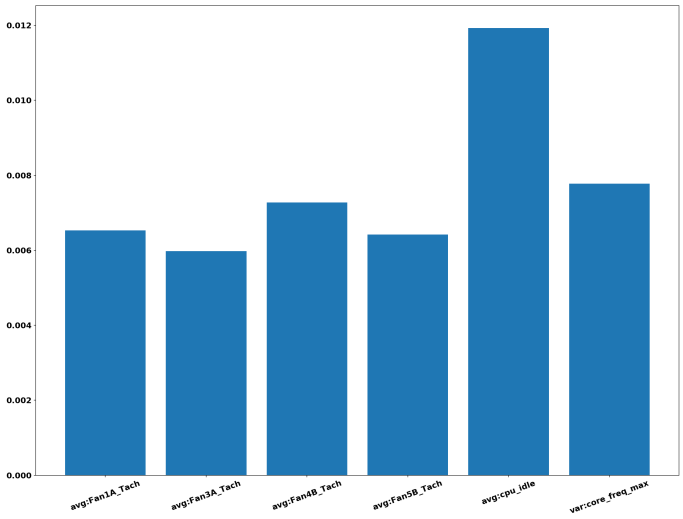


Figura 6.48: Features rilevanti nei punti anomali per il nodo r193c13s04

### Nodi r183c10s04 e r183c16s04

Analizzando con LIME i nostri due casi particolare, i nodi r183c10s04 e r183c16s04, nei punti anomali (figure 6.49 e 6.50) si nota come le features più determinanti sono "avg:CMOS_battery" e "avg:mem_cached" che non sono le più rilevanti negli altri nodi. I due nodi sono accomunati, in modo molto rilevante, da "avg:CMOS_battery" che potrebbe essere l'anomalia dei due nodi nel periodo considerato. Tuttavia, si ripetono anche quelle features che sono state determinanti negli altri nodi. L'errore di predizione può essere causato, quindi, da un'anomalia diversa rispetto agli altri nodi. Ciò è suggerito dall'assenza della feature "avg:CPU_idle".

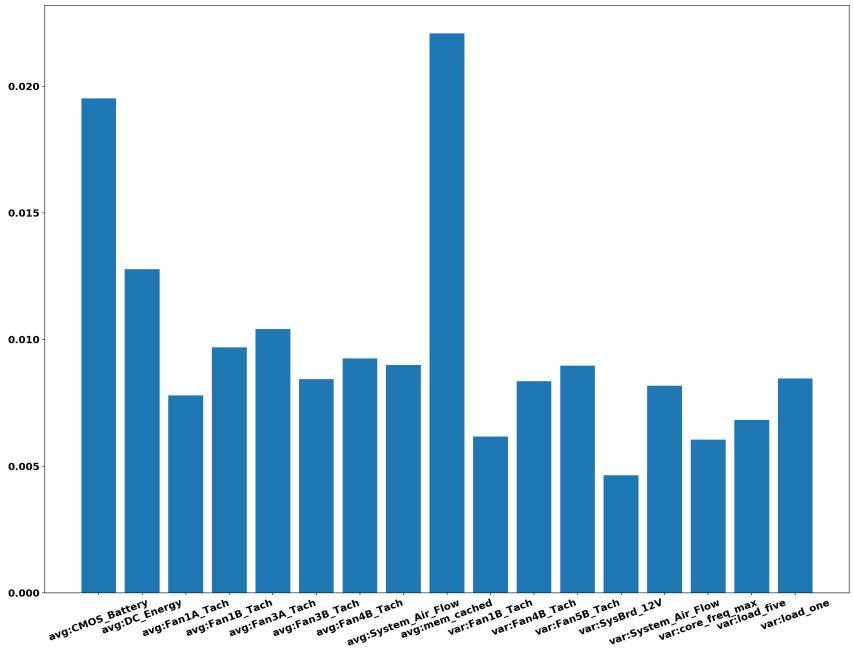


Figura 6.49: Features rilevanti nei punti anomali per il nodo r193c10s04

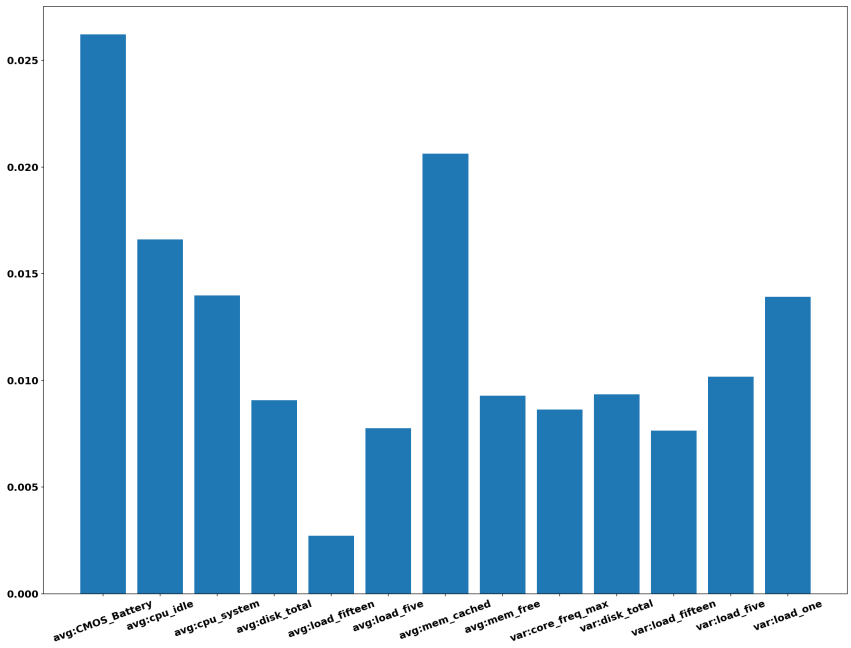


Figura 6.50: Features rilevanti nei punti anomali per il nodo r193c16s04

### 6.4.2 Considerazioni

In conclusione, osservando una panoramica che comprende non solo i nodi analizzati nella sezione precedente ma tutti quelli su cui è stato fatto l'addestramento del modello, si notano che le features di maggiore rilevanza a supportare una predizione di anomalia del modello, sono quelle riferite al sistema di raffreddamento, funzionamento della CPU e del disco. In particolare, la feature con la maggiore occorrenza si riferisce allo stato di idle del sistema.

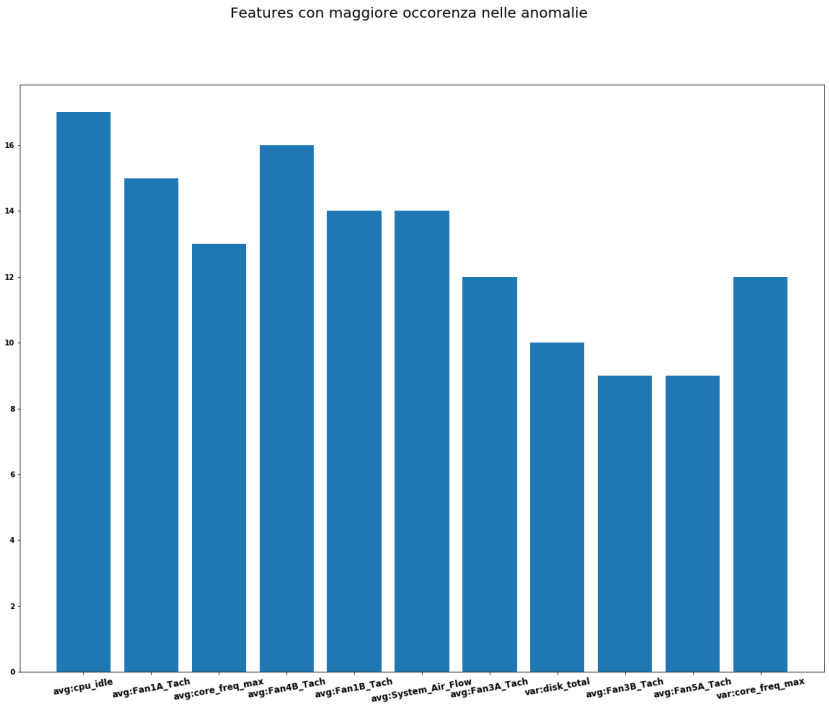


Figura 6.51: Occorrenze delle features più rilevanti secondo LIME

Capitolo 7

# Conclusione e Sviluppi Futuri

Attraverso la visualizzazione delle attivazioni e l'analisi dei risultati mediante LIME, si è scoperto che l'origine delle anomalie derivano dallo stato di idle del nodo. Infatti, lo stato descritto dal plugin Nagios indica uno stato di idle, in cui il nodo è entrato a causa dell'anomalia, e solo dopo il ripristino da parte dell'amministratore di sistema, il nodo esce dalla condizione di idle in cui un utente è impossibilitato ad usare le risorse. Osservando in figura 7.1, un esempio di periodo di funzionamento analizzato in un nodo, si può pensare di analizzare il periodo pre-anomalia (periodi 1 e 2 indicati in figura) per verificare se il modello realizzato riesce a prevedere un'anomalia prima dell'intervento dell'amministratore di sistema.

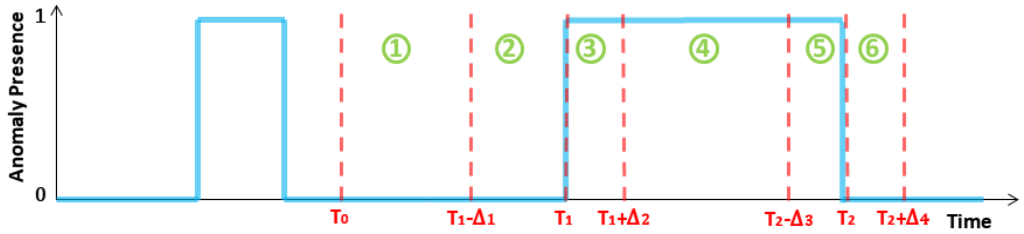


Figura 7.1: Esempio di periodo di funzionamento di un nodo; sull'asse x abbiamo il tempo, sull'asse y la presenza di un anomalia indicato dallo stato 1.

Analizzando le predizioni in un periodo, come descritto in figura 7.1,

dei nodi r183c09s01 (f1-score 0.61), r183c09s02(f1-score 0.50), r183c10s01(f1-score 0.50), r183c10s02(f1-score 0.80), r183c10s03(f1-score 0.67), r183c13s01(f1-score 0.94) e r183c14s01(f1-score 0.44) si ottengono i risultati raffigurati nelle figure 7.2, 7.3, 7.4, 7.5, 7.6 e 7.7. Le figure mostrano il valore di probabilità di anomalia sull'asse delle ordinate e gli indici dei dati sull'asse delle ascisse che rappresentano in successione un periodo del nodo. Si nota come il modello nei 60-75 minuti prima dell'avvento del periodo etichettato da nagios come anomalo, inizia a predire con una probabilità maggiore di 0 le anomalie (si ricorda che nel nostro lavoro abbiamo considerato anomalie predette correttamente, quelle con probabilità maggiore della soglia ossia 0.5). Pertanto, il modello sembra rilevare le anomalie prima che il nodo entra in idle e vengono rilevate, nel nostro caso come falsi negativi o punti normali poichè sotto soglia.

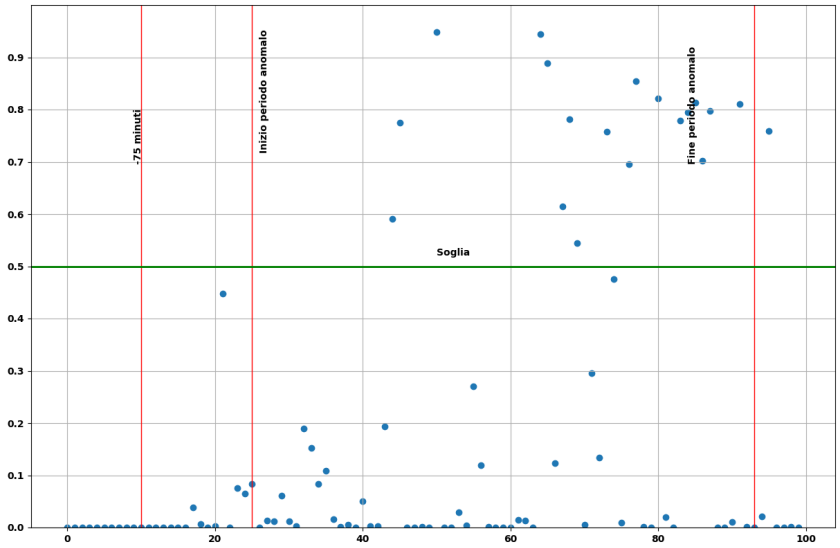


Figura 7.2: Predizioni del modello nel periodo pre-anomalo, anomalo e post-anomalo del nodo r183c09s01

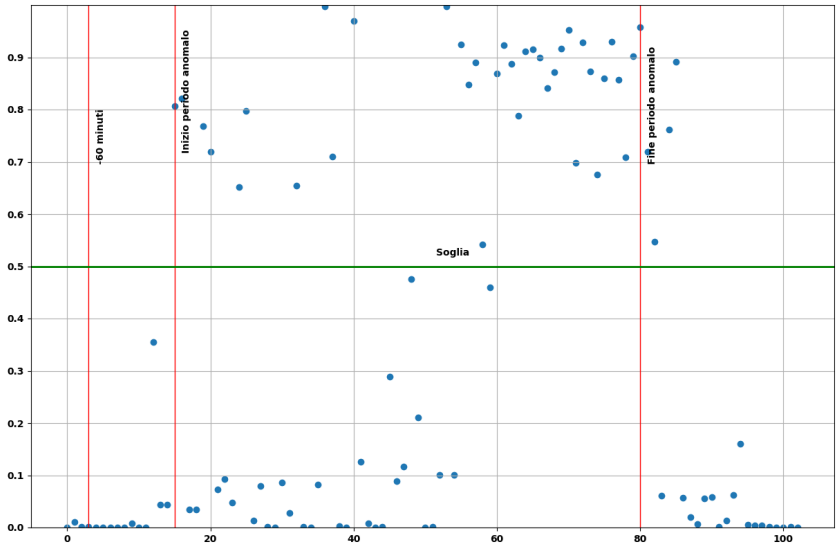


Figura 7.3: Predizioni del modello nel periodo pre-anomalo, anomalo e post-anomalo del nodo r183c09s02

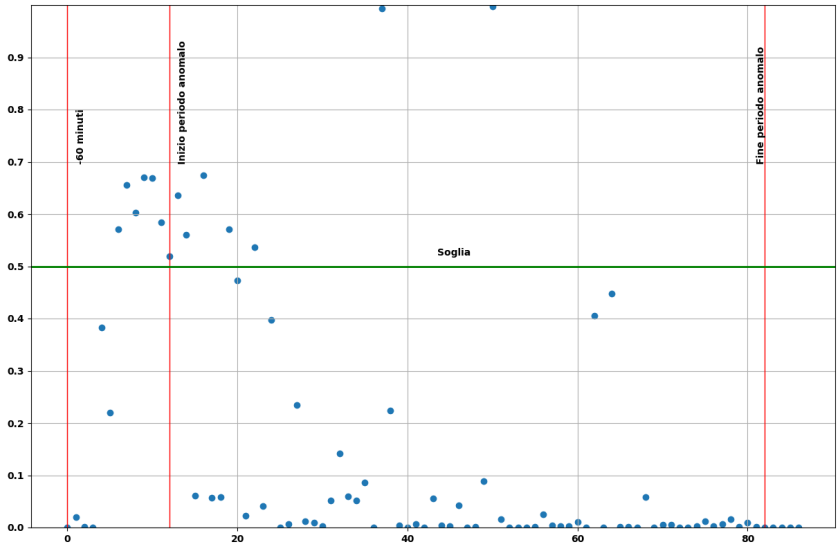


Figura 7.4: Predizioni del modello nel periodo pre-anomalo, anomalo e post-anomalo del nodo r183c10s02

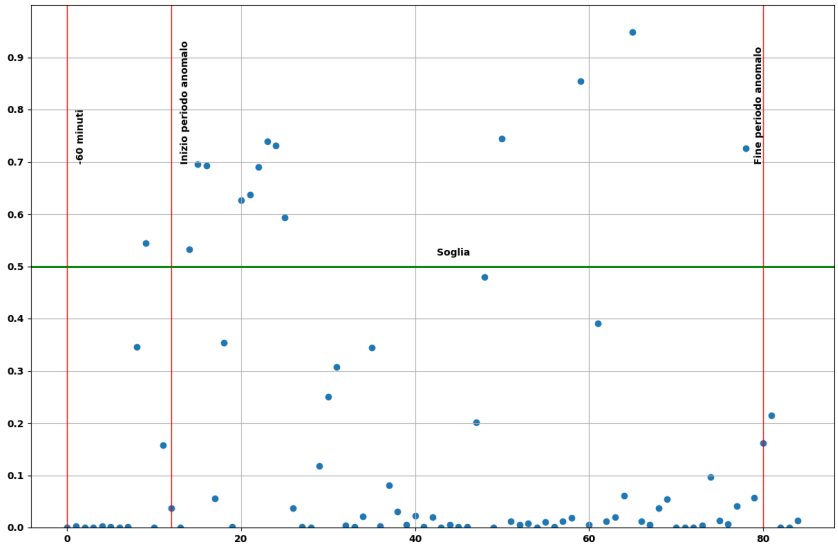


Figura 7.5: Predizioni del modello nel periodo pre-anomalo, anomalo e post-anomalo del nodo r183c10s01

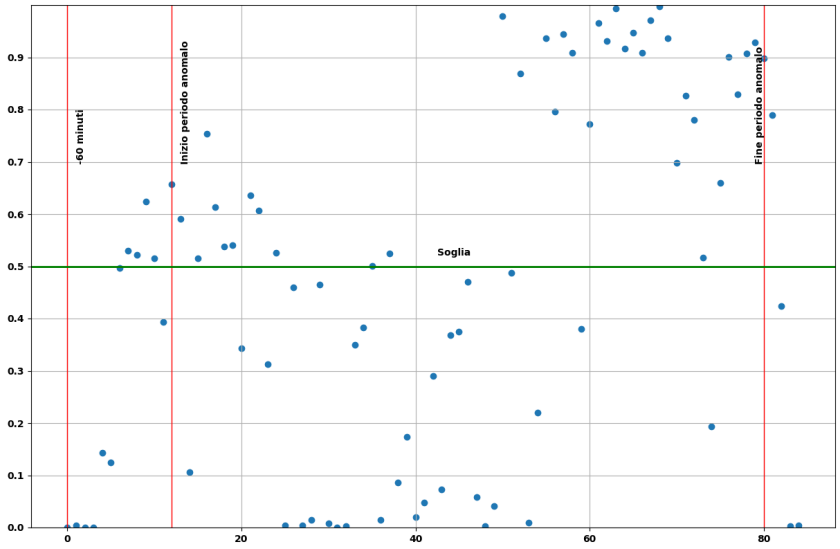


Figura 7.6: Predizioni del modello nel periodo pre-anomalo, anomalo e post-anomalo del nodo r183c10s03

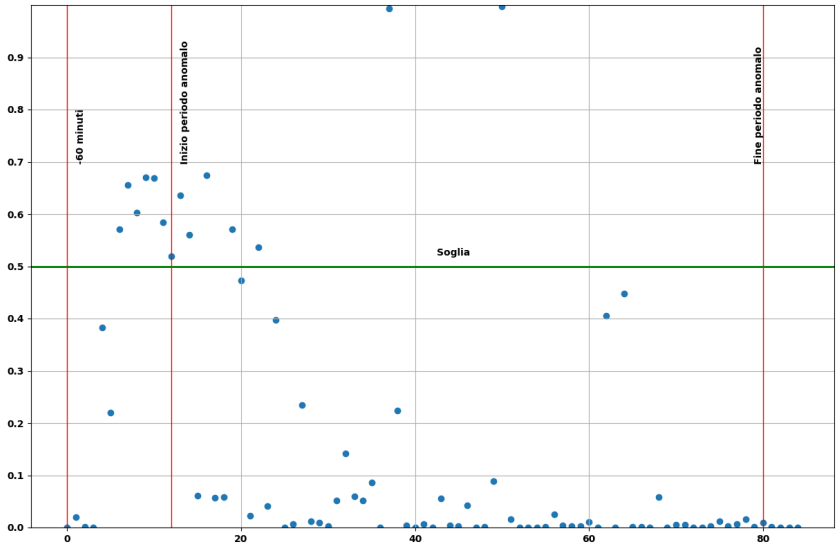


Figura 7.7: Predizioni del modello nel periodo pre-anomalo, anomalo e post-anomalo del nodo r183c13s01

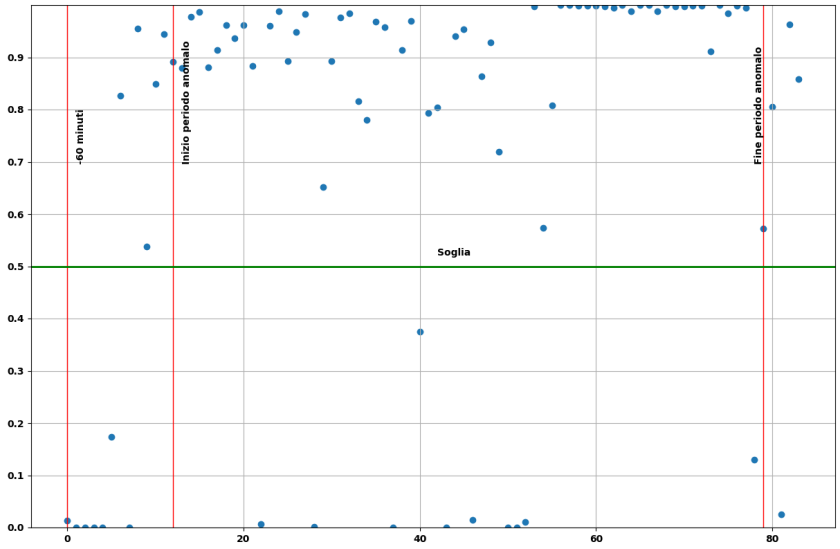


Figura 7.8: Predizioni del modello nel periodo pre-anomalo, anomalo e post-anomalo del nodo r183c14s01

In conclusione, uno degli approcci possibili oltre ai già citati, Zero-Shot Learning e apprendimento semi-supervisionato in sezione 5.6, è separare le anomalie dai periodi di idle, etichettati da nagios, e riaddestrare i modelli, per poi analizzare se con un sotto-insieme di nodi si riesce a trasferire l'apprendimento su tutti i nodi dell'HPC.

# Elenco delle figure

# Elenco delle tabelle

# Bibliografia


[1] C. W. Italia, "Hpc: conosciamo meglio l'high performance computing," 2019, available on line. [Online]. Available: https://www.cwi.it/data-center/high-performance-computing-hpc/hpc-conosciamo-meglio-lhigh-performance-computing-122419#

[2] S. C. Foundation, "What is an hpc system?" 2019, available on line. [Online]. Available: https://epcced.github.io/hpc-intro/010-hpc-concepts/

[3] Top500List, "Top500 list - november 2019," 2019, available on line. [Online]. Available: https://www.top500.org/list/2019/11/?page=1

[4] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[5] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[6] C. C. Aggarwal, "An introduction to outlier analysis," in *Outlier analysis*. Springer, 2017, pp. 1–34.

[7] M. B. Goldstein, *Anomaly detection in large datasets*. Verlag Dr. Hut, 2014.

[8] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini, "Anomaly detection using autoencoders in high performance compu-

ting systems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 9428–9433.

[9] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[10] M. Mohammed, M. Khan, and E. Bashier, *Machine Learning: Algorithms and Applications*, 07 2016.

[11] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, July 1959.

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

[13] S. Harnad, "The annotation game: On turing (1950) on computing, machinery, and intelligence," in *The Turing test sourcebook: philosophical and methodological issues in the quest for the thinking computer*. Kluwer, 2006.

[14] Wikipedia, "Machine learning," 2019, [Online; in data 02-febbraio-2020]. [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning

[15] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[16] M. Troìa, "Machine learning: la regressione lineare," 2018, [Online; in data 06-febbraio-2020]. [Online]. Available: https://medium.com/matteotroia/machine-learning-la-regressione-lineare-7b2f096adc26

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[18] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[19] MindOrks, "understanding the recurrent neural network," 2018, available on line. [Online]. Available: https://medium.com/mindorks/understanding-the-recurrent-neural-network-44d593f112a2

[20] Wikipedia, "Rete neurale ricorrente — wikipedia, l'enciclopedia libera," 2019, [Online; in data 25-gennaio-2020]. [Online]. Available: http://it.wikipedia.org/w/index.php?title=Rete_neurale_ricorrente&oldid=105165899

[21] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini, "A semisupervised autoencoder-based approach for anomaly detection in high performance computing systems," *Engineering Applications of Artificial Intelligence*, vol. 85, pp. 634–644, 2019.

[22] K. Leto, "Anomaly detection in hpc systems," Master's thesis, Università di Bologna, 2019.

[23] P. Dayan, M. Sahani, and G. Deback, "Unsupervised learning," *The MIT encyclopedia of the cognitive sciences*, pp. 857–859, 1999.

[24] D. Greene, P. Cunningham, and R. Mayer, "Unsupervised learning and clustering," in *Machine learning techniques for multimedia*. Springer, 2008, pp. 51–90.

[25] N. Suthar, P. Indr, and P. Vinit, "A technical survey on dbscan clustering algorithm," *Int. J. Sci. Eng. Res*, vol. 4, pp. 1775–1781, 2013.

[26] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to data mining*. Pearson Education India, 2016.

[27] B. R. Joseph Rocca, "Understanding variational autoencoders (vaes)," 2019, [Online; in data 01-febbraio-2020]. [Online]. Available: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

[28] toward data science, "Applied deep learning - part 3: Autoencoders," 2017, [Online; in data 02-febbraio-

2020]. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

[29] V. J. Prakash and D. L. Nithya, "A survey on semi-supervised learning techniques," *International Journal of Computer Trends and Technology*, vol. 8, no. 1, p. 25–29, Feb 2014. [Online]. Available: http://dx.doi.org/10.14445/22312803/IJCTT-V8P105

[30] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, 2015.

[31] "Autoencoders and anomaly detection with machine learning in fraud analytics," 2017, [Online; in data 02-febbraio-2020]. [Online]. Available: https://shiring.github.io/machine_learning/2017/05/01/fraud

[32] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[33] "5 things you need to know about reinforcement learning," 2018, [Online; in data 2-febbraio-2020]. [Online]. Available: https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html

[34] Wikipedia, "Q-learning — wikipedia, l'enciclopedia libera," 2019, [Online; in data 2-febbraio-2020]. [Online]. Available: it.wikipedia.org/w/index.php?title=Q-learning&oldid=109233200

[35] K. Beneventi, "Keras documentation," 2020, [Online; in data 2-febbraio-2020]. [Online]. Available: https://keras.io/

[36] Tensorflow, "Tensorflow documentation," 2020, [Online; in data 2-febbraio-2020]. [Online]. Available: https://www.tensorflow.org/

[37] F. Beneventi, "Examon-client," 2019, [Online; in data 2-febbraio-2020]. [Online]. Available: https://git.eees.dei.unibo.it/francesco.beneventi/examon-client

[38] Pandas, “Pandas documentation,” 2020, [Online; in data 2-febbraio-2020]. [Online]. Available: https://pandas.pydata.org/

[39] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[40] A. Gonfalonieri, “Applications of zero-shot learning,” 2019, [Online; in data 05-febbraio-2020]. [Online]. Available: https://towardsdatascience.com/applications-of-zero-shot-learning-f65bb232963f

[41] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, “A survey of methods for explaining black box models,” *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

[42] M. T. Ribeiro, S. Singh, and C. Guestrin, “" why should i trust you?" explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[43] A. Sharma, “Decrypting your machine learning model using lime,” 2019, [Online; in data 06-febbraio-2020]. [Online]. Available: https://towardsdatascience.com/decrypting-your-machine-learning-model-using-lime-5adc035109b5

[44] T. Martin, “Shining a light on black box models with lime,” 2019, [Online; in data 06-febbraio-2020]. [Online]. Available: http://tpgmartin.com/shining-a-light-on-black-box-models-with-lime