POLITECNICO DI TORINO

Corso di Laurea
in Ingegneria Matematica

Tesi di Laurea

Anomaly Detection Applicata all'Industria di Processo

Relatori
prof. Alessandro Fiori
prof. Andrea Avignone
firma dei relatori

.....
.....

Candidato

Giovanni Monco
firma del candidato

.....

Anno Accademico 2024-2025

Sommario

L'individuazione di anomalie nelle serie temporali multivariate è un compito critico in vari settori, tra cui l'industria di processo. Questo studio propone una pipeline completa e ottimizzata per il rilevamento delle anomalie, concentrandosi su metodi basati sul deep learning, in particolare gli autoencoder e le LSTM.

La pipeline comprende un'analisi dettagliata delle fasi specifiche di preprocessing per le serie temporali, ottimizzate per garantire robustezza e precisione al modello. La metodologia è stata convalidata su quattro dataset relativi all'industria di processo, utilizzando l'F1 score come metrica principale per la valutazione delle prestazioni.

L'ottimizzazione delle fasi di preprocessing, così come dei parametri dei modelli di anomaly detection, è stata condotta tramite ricerche a griglia complete che, esplorando sistematicamente tutte le combinazioni di parametri, garantiscono configurazioni ottimali per ogni dataset e permettono di individuare possibili configurazioni comuni.

Questo lavoro fornisce un contributo alla letteratura già esistente offrendo linee guida pratiche e innovative per l'implementazione di pipeline di anomaly detection validate su dati reali e applicabili a diversi scenari industriali.

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione

Nell'analisi dei dati, il rilevamento delle anomalie, o **anomaly detection**, consiste nell'individuare osservazioni, elementi o eventi che si discostano in modo significativo dal comportamento della maggior parte dei dati. La rilevazione delle anomalie è complessa perché le anomalie sono rare, poco definite e dipendono fortemente dal contesto. Nel caso delle serie temporali, l'anomaly detection si focalizza sull'identificazione di eventi che si manifestano nel tempo, come cambiamenti improvvisi nei trend dei dati e picchi anomali nei valori.

Il rilevamento di sequenze anomale nei dati temporali è fondamentale per migliorare l'efficienza e la sicurezza in vari ambiti applicativi, come dimostrato nel contesto di questa ricerca. Ad esempio, nei processi industriali, Castro et al. [1], Pota et al. [2] e Tian et al. [3] hanno evidenziato come l'individuazione tempestiva di anomalie possa prevenire guasti di produzione. Nel settore finanziario Crépey et al. [4] hanno studiato l'importanza del rilevamento di anomalie per identificare irregolarità nei mercati e nei modelli di rischio. Nel monitoraggio sanitario Yang et al. [5] hanno sottolineato come anomalie nei dati fisiologici, come il ritmo cardiaco, possano indicare condizioni critiche di salute. L'identificazione tempestiva di queste anomalie è quindi di fondamentale importanza per prevenire eventi dannosi.

Nelle applicazioni industriali, le serie temporali presentano dati di grandi dimensioni e strutture complesse, spesso di difficile interpretazione. Questa complessità deriva dalla natura tipicamente **non stazionaria e multidimensionale** di tali serie (maggiori dettagli nel Capitolo 2). L'analisi delle serie temporali multivariate richiede non solo di esaminare l'evoluzione di ciascuna variabile nel tempo, ma anche di comprendere le relazioni dinamiche tra variabili diverse a ogni istante, aumentando significativamente la difficoltà del problema.

Le serie temporali industriali presentano spesso diverse criticità che possono nascondere comportamenti rilevanti, tra cui rumore, variazioni di scala tra le variabili, elevata dimensionalità e non stazionarietà. Il **preprocessing** dei dati consiste in un insieme di operazioni di pulizia e trasformazione finalizzate a rendere questi dati grezzi adatti all'analisi. Questo passaggio è fondamentale per migliorare la qualità dei dati, garantire

coerenza e ottimizzare le prestazioni dei modelli di anomaly detection, come confermano i risultati di questa ricerca nel Capitolo 6. Il Capitolo 3 approfondisce nel dettaglio le tecniche di preprocessing adottate.

L'obiettivo di questo studio è contribuire alla ricerca nel campo della rilevazione delle anomalie in ambito industriale, attraverso un'analisi approfondita dei metodi impiegati per l'identificazione delle anomalie nelle serie temporali multivariate (maggiori dettagli nel Capitolo 4) e delle tecniche utilizzate per l'elaborazione dei dati. Il lavoro si distingue per l'attenzione posta sulla selezione delle fasi di preprocessing e per lo scenario di applicazione industriale considerato. In particolare, l'intento è offrire una panoramica dei possibili approcci in scenari in cui si possono identificare solamente periodi distinti di misurazione. Questi periodi sono classificati genericamente come contenenti anomalie o assenza di anomalie, ma senza informazioni precise sulle anomalie. Questa situazione è nota come **broad classification**, ed è una situazione comune a molte aziende nell'industria di processo. In altre parole, ci sono dati etichettati come zero, indicanti l'assenza di anomalie, e dati non etichettati, per i quali non è disponibile alcuna informazione preliminare sulla presenza di anomalie. Il criterio utilizzato per suddividere i dati è illustrato in Figura 1.1.

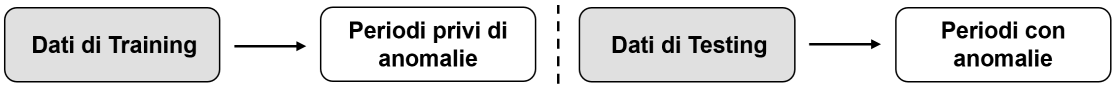

Figura 1.1: Criterio di suddivisione dei dati.

La ricerca utilizza **quattro dataset pubblici** relativi all'industria di processo, descritti nella Sezione 5.1, che includono sia periodi con che periodi senza anomalie, ma con le anomalie sempre etichettate. Questa configurazione permette di valutare e confrontare in modo oggettivo le prestazioni degli algoritmi, cosa che non sarebbe possibile in assenza di etichette. L'obiettivo è duplice: da un lato, individuare le configurazioni più efficaci per il rilevamento delle anomalie; dall'altro, esaminare l'influenza delle diverse fasi di preprocessing sui risultati. In questa maniera si forniscono indicazioni pratiche per affrontare problemi analoghi in contesti industriali reali.

La pipeline generale adottata in questo lavoro è schematizzata nella Figura 1.2. Durante la fase di addestramento (training), vengono utilizzati in input dati privi di anomalie, che vengono sottoposti a un processo di pre-elaborazione. Successivamente, su questi dati viene allenato un modello di ricostruzione o previsione, come descritto più dettagliatamente nella Sezione 2.1 e nel Capitolo 4. L'output della fase di addestramento include sia i dati ricostruiti o predetti, sia una **soglia di errore**. Nella fase di verifica (testing), questa soglia viene utilizzata per classificare le osservazioni come anomale, qualora l'errore superi il valore prestabilito. La scelta della soglia è cruciale. Strategie comuni, come quelle descritte in Pota et al. [2], ottimizzano una metrica di prestazione per determinarla. In questo lavoro, invece, la selezione della soglia è stata condotta in modo indipendente dalla fase di test, secondo un approccio empirico illustrato nel dettaglio nella Sezione 5.3.

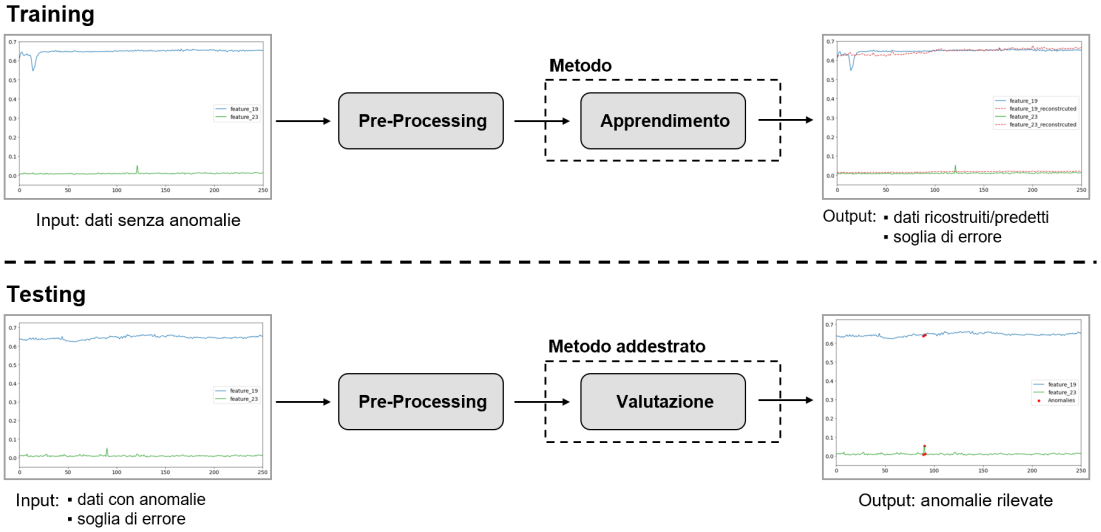

Figura 1.2: Pipeline adottata per l'anomaly detection in serie temporali multivariate (i dati sono tratti dal dataset PSM).

1.1 Contesto e studi precedenti

La complessità dei dati delle serie temporali nelle applicazioni reali ha spinto la comunità scientifica a sviluppare algoritmi specializzati per il rilevamento automatico di modelli anomali. Nel corso degli anni, il numero e la diversità delle tecniche di rilevamento delle anomalie sono aumentati considerevolmente. Diversi studi hanno analizzato e messo a confronto i vari metodi di rilevamento delle anomalie in vari ambiti. Questi lavori includono: Schmidl et al. [6], Belay et al. [7], Darban et al. [8], Choi et al. [9], Castro et al. [1], Ruff et al. [10]. Tali studi si focalizzano non solo sul semplificare la selezione dell'algoritmo più adatto, ma anche a suggerire nuove direzioni di ricerca nel campo del rilevamento delle anomalie.

Gli studi più specifici presi come punto di riferimento, in quanto applicano gli algoritmi di rilevamento delle anomalie in contesti industriali, sono i lavori di Pota et al. [2], Panza et al. [11], Zope et al. [12] e West et al. [13]. Inoltre, lo studio recente di Tawakuli et al. [14] è stato preso come punto di partenza per sviluppare la parte riguardante il preprocessing (Capitolo 3). Esso fornisce una rassegna generale e completa delle tecniche di preprocessing delle serie temporali e ne evidenzia l'importanza nel migliorare l'addestramento dei modelli basati sulle reti neurali.

A partire dagli anni 2020 non esiste uno studio esaustivo che valuti e confronti in modo sistematico i diversi approcci all'anomaly detection. Di conseguenza la scelta della tecnica più efficace per un determinato compito di anomaly detection rappresenta una sfida complessa. Un contributo significativo in questo ambito è lo studio di Schmidl et al. [6], uno dei primi lavori scientifici a fornire una valutazione sistematica dei principali algoritmi di rilevamento delle anomalie. Gli autori hanno raccolto e reimplementato numerosi

algoritmi sviluppati in diversi domini, confrontandone le prestazioni su serie temporali. Il lavoro presenta un'impostazione ampia e di natura generale, e ha gettato le basi per studi successivi più specifici, focalizzati solo su particolari ambiti applicativi o su determinate categorie di tecniche di anomaly detection. Tutti i materiali utilizzati negli esperimenti sono disponibili al seguente `link`.

Nelle applicazioni le aziende si trovano spesso a dover affrontare la rilevazione di anomalie in dataset con informazioni limitate, in cui generalmente sono disponibili solo etichette a livello di broad classification, come detto in precedenza. In questi scenari si ricorre all'utilizzo di approcci semi-supervisionati o non supervisionati, in quanto in grado di operare senza la necessità di etichette dettagliate. L'articolo di Belay et al. [7] offre una rassegna completa degli algoritmi più avanzati per il rilevamento delle anomalie non supervisionate, presentando una valutazione numerica dettagliata di 13 algoritmi testati su due dataset multivariati pubblici. Oltre all'analisi sperimentale, lo studio fornisce un ampio background teorico, come la suddivisione dei metodi non supervisionati che verrà approfondita nel Capitolo 2. Tuttavia, la sezione relativa ai risultati non è perfettamente riproducibile dalle informazioni fornite, ad esempio mancano informazioni precise sull'implementazione e l'addestramento delle reti neurali utilizzate.

Tra le diverse categorie di metodi non supervisionati del Capitolo 2, quelli basati sulla ricostruzione sono particolarmente diffusi per la loro efficacia, applicabilità e flessibilità. Questi metodi riducono la dimensionalità della serie temporale multivariata, comprimendola in uno spazio più piccolo e ricostruendola nelle sue dimensioni originali. L'idea alla base di questi modelli è che le anomalie siano difficili da ricostruire con precisione; pertanto, l'errore di ricostruzione, ossia la differenza tra i dati ricostruiti e quelli originali, può essere utilizzato come indicatore di anomalie. Un errore di ricostruzione basso suggerisce un comportamento normale, mentre un errore elevato indica la presenza di dati anomali. L'articolo di Pota et al. [2] fornisce un'analisi comparativa focalizzata esclusivamente su questo tipo di architettura. L'analisi è molto dettagliata e precisa, offrendo tutte le informazioni necessarie per replicare gli esperimenti, comprese le specifiche dei modelli applicati. Inoltre, coerentemente con l'approccio adottato in questo lavoro, i dati utilizzati per l'addestramento sono esclusivamente privi di anomalie.

La maggior parte dei lavori sopra citati si basa prevalentemente su metodi di deep learning. L'aumento della complessità e della dimensione dei dati ha progressivamente motivato lo sviluppo di modelli specializzati di deep learning nel corso degli anni, progettati per identificare comportamenti anomali. Questi modelli sfruttano reti neurali complesse per catturare dipendenze temporali intricate e relazioni non lineari all'interno dei dati, risultando particolarmente efficaci nel contesto delle serie temporali multivariate, dove la struttura dei dati è altamente complessa. Un contributo significativo in questo campo è fornito dai recenti studi di Darban et al. [8] e Choi et al. [9], che si concentrano esclusivamente sulle tecniche di deep learning. Entrambi forniscono una panoramica completa dei principali algoritmi di anomaly detection basati su reti neurali, accompagnata da un'adeguata cornice teorica. Tuttavia, solitamente in questi testi che esaminano un gran numero di algoritmi, le informazioni sull'implementazione sono limitate, salvo la disponibilità di un archivio pubblico. È questo il caso del lavoro di Darban et al. [8], che fornisce un archivio contenente codice e implementazioni al seguente `link`.

In letteratura esistono studi comparativi sugli algoritmi di anomaly detection applicati a specifici scenari industriali. Tra questi, il lavoro di Pota et al. [2] esamina un forno a vuoto usato nell'industria aerospaziale per il trattamento termico sotto vuoto, con l'obiettivo di individuare anomalie in tempo reale e prevenire guasti o rallentamenti nel processo produttivo. Un altro contributo rilevante è offerto da Castro et al. [1], che esamina lo sviluppo delle tecniche di monitoraggio delle condizioni, diagnostica e prognosi applicate alle turbine a gas nell'ultimo decennio. Lo studio pertanto non si limita solo al rilevamento delle anomalie. Inoltre questo lavoro non fornisce, a differenza di altre analisi, risultati dettagliati in tabelle con misure di performance per ogni dataset, poiché si concentra su un confronto generale e conciso degli algoritmi basato sulle loro caratteristiche.

La presente tesi propone un'analisi critica e comparativa degli algoritmi di deep learning per l'identificazione di anomalie in serie temporali multivariate di natura industriale, concentrandosi su scenari in cui il modello viene addestrato esclusivamente su dati privi di anomalie e successivamente valutato su dati contenenti anomalie. Questa ricerca si concentra su due aspetti chiave spesso trascurati: la fase di preprocessing dei dati, che è fondamentale per garantire un input ottimale ai modelli, e la selezione della soglia di errore di ricostruzione (o previsione) durante l'addestramento, un passaggio critico per migliorare l'affidabilità nella rilevazione delle anomalie. Questi aspetti sono spesso trascurati a causa del tempo richiesto per ottimizzarli e della maggiore enfasi posta sui modelli, ma sono cruciali per il successo del modello, come dimostrano i risultati ottenuti.

Capitolo 2

Principi Anomaly Detection e Serie Temporali

Per offrire una visione più chiara dell'anomaly detection nelle serie temporali multivariate, questo capitolo introduce la **notazione** e le **basi teoriche** necessarie, strutturandosi in tre sezioni. La prima Sezione 2.1 definisce il problema del rilevamento delle anomalie e classifica i metodi di soluzione; la seconda Sezione 2.2 analizza le tipologie di anomalie nei dati temporali; la terza Sezione 2.3 esplora le proprietà delle serie temporali multivariate, evidenziando le sfide nel rilevamento delle anomalie.

2.1 Anomaly Detection: introduzione al problema e ai metodi di soluzione

L'**anomaly detection (AD)** [15] consiste nell'identificare osservazioni, pattern o eventi che si discostano in modo significativo dal comportamento atteso nei dati. In altre parole, il suo obiettivo è riconoscere quelle osservazioni che non seguono la distribuzione predominante del dataset. Tali osservazioni sono pertanto dette **anomalie**.

In base alla disponibilità di etichette nei dati, i metodi di rilevamento delle anomalie sono suddivisi da Chandola et al. [16] in tre categorie principali: supervisionati, semi-supervisionati e non supervisionati.

- **Metodi supervisionati**:

  Questi metodi richiedono un dataset di addestramento con dati etichettati sia per le istanze normali che per quelle anomale. Un problema significativo in questo approccio è ottenere dati etichettati rappresentativi delle anomalie, poiché queste sono eventi rari. Questa limitazione rende l'anomaly detection supervisionata meno applicabile in molti scenari reali.

- **Metodi semi-supervisionati**:

Questa classe di metodi presuppone la disponibilità di etichette solo per le istanze normali, senza la necessità di conoscere le anomalie a priori. Questo metodo è maggiormente praticabile perché non richiede dati di addestramento etichettati.

- **Metodi non supervisionati**:

  Quest'ultima categoria di metodi non richiede dati di addestramento etichettati e fa l'assunzione implicita che le istanze normali siano molto più frequenti rispetto alle anomalie nel dataset. Questo approccio è il più diffuso perché si applica a molti problemi.

Come discusso nella Sezione 1.1, nella tesi l'anomaly detection è trattata come un problema non supervisionato, in quanto non si dispone di etichette che definiscano a priori cosa costituisce un'anomalia. In questo approccio il modello è sviluppato per riconoscere i pattern tipici/normali nei dati e segnalare come anomali quei campioni che deviano significativamente da tali pattern [16]. Inoltre, l'approccio non supervisionato risulta particolarmente adatto nei contesti dinamici, in cui nuove anomalie possono emergere senza che siano stati forniti esempi etichettati di tali comportamenti anomali.

A loro volta, secondo Belay et al. [7], i metodi di anomaly detection non supervisionati si suddividono in **tre principali categorie di approcci**: di ricostruzione, predittivi e di compressione (vedi la seguente Fig. 2.1).

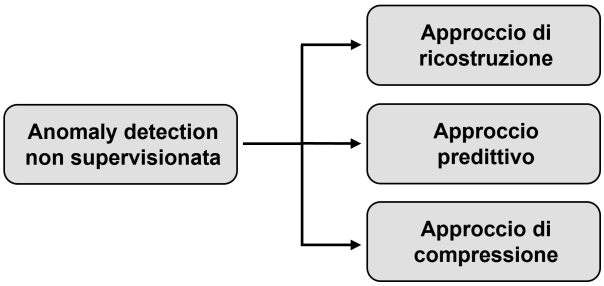

Figura 2.1: Approcci per l'anomaly detection non supervisionata.

- **Approccio basato sulla ricostruzione**:

  Questo metodo consiste nel ridurre la dimensionalità della serie temporale multivariata, comprimendola in uno spazio di dimensioni inferiori, per poi ricostruire i dati originali. Si presuppone che sia complesso ricostruire accuratamente i dati anomali. Pertanto, l'errore tra i dati ricostruiti e quelli originali, chiamato errore di ricostruzione, può essere utilizzato per rilevare le anomalie. Tra le tecniche più comunemente adottate in questo approccio troviamo l'analisi delle componenti principali (Sezione 4.1.2) e gli autoencoder (Sezione 4.2).

- **Approccio basato sulla previsione**:

  In questo approccio i valori passati e correnti della serie temporale vengono utilizzati per effettuare previsioni. I valori previsti vengono quindi confrontati con quelli

effettivamente osservati, e l'anomalia viene identificata in base alla differenza tra questi due valori. Tra le tecniche più comuni in questa categoria si trovano i modelli autoregressivi come ARIMA (Sezione 4.1.1) e le reti neurali ricorrenti (Sezione 4.4).

- **Approccio basato sulla compressione**:

  Come per l'approccio di ricostruzione, questo metodo riduce le dimensioni dei segmenti della serie temporale, rappresentandoli in uno spazio latente a bassa dimensione. Tuttavia, anziché cercare di ricostruire le sequenze originali, l'anomalia viene identificata direttamente nello spazio latente. Queste tecniche di riduzione della dimensionalità sono solitamente utilizzate per velocizzare i calcoli e ridurre la complessità del modello.

Un'ulteriore classificazione di Belay et al. [7] dei metodi di anomaly detection non supervisionati nelle serie temporali multivariate suddivide le tecniche in tre categorie principali, che si differenziano dalle categorie di approcci precedentemente menzionate.

1. **Metodi convenzionali/statistici**: includono modelli autoregressivi e modelli di clustering.
2. **Metodi basati sulle reti neurali**: tra questi rientrano autoencoder, reti neurali ricorrenti come la LSTM (Long Short-Term Memory) e modelli generativi com le GAN (Generative Adversarial Networks).
3. **Metodi composti**: comprendono modelli in cui si combinano due o più approcci delle categorie precedenti.

In Figura 2.2 queste tre categorie di metodi sono elencate insieme alle relative tecniche; le tecniche trattate nella tesi sono riportate in grassetto.

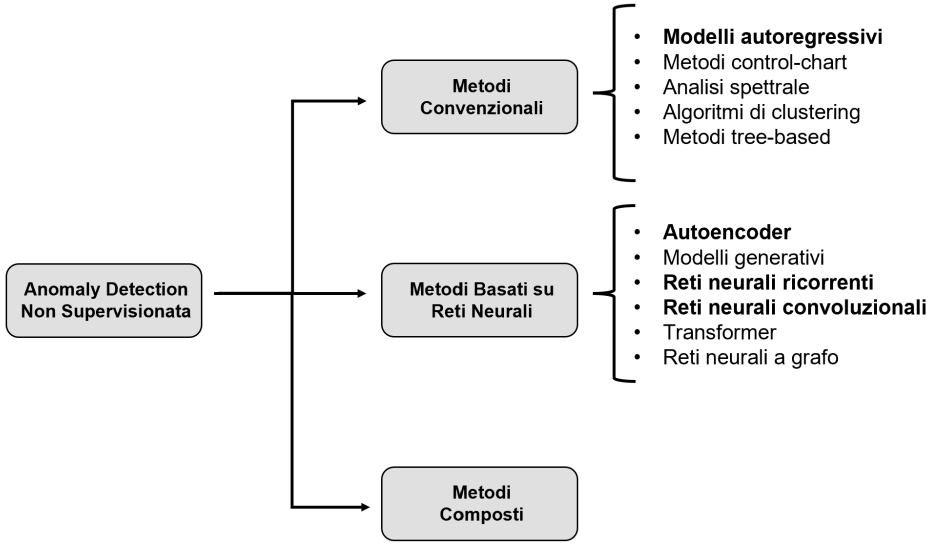

Figura 2.2: Metodi di anomaly detection non supervisionati.

Dopo aver illustrato i diversi approcci all'anomaly detection, ci si concentra ora esclusivamente su quelli analizzati in questa tesi, ovvero gli **approcci di ricostruzione e previsione**. Questi metodi sono stati scelti perché particolarmente adatti al contesto applicativo considerato (maggiori dettagli nel Capitolo 1), caratterizzato da broad classification, in cui le anomalie vengono etichettate solo a livello di interi periodi e non come singoli eventi. In questo contesto l'errore di ricostruzione (o previsione) è utilizzato come criterio per identificare anomalie, assumendo che i dati anomali non possano essere ricostruiti accuratamente. In generale, l'errore di ricostruzione (o previsione) per un'osservazione al tempo t di una serie storica multivariata $x \in \mathbb{R}^{T \times N}$ è dato dalla differenza tra il valore originale x_t e il valore ricostruito (o previsto) $\hat{x}_t$:

$$\|x_t - \hat{x}_t\|\,, \tag{2.1}$$

dove $\|\cdot\|$ rappresenta una norma (generalmente la norma euclidea $\|x_t\|_2 = \sqrt{x_{t,1}^2 + \cdots + x_{t,N}^2}$).
In particolare, è stato adottato il **mean squared error (MSE)** (errore quadratico medio) come misura dell'errore per ciascuna osservazione nella serie temporale. L'MSE è definito come:

$$MSE = \frac{1}{T}\sum_{t=1}^{T} \|x_t - \hat{x}_t\|^2\,, \tag{2.2}$$

dove T rappresenta il numero di osservazioni.
Questo errore misura quanto un'osservazione si discosta dal modello, fungendo da indicatore di anomalie. Un errore elevato suggerisce che il modello non riesce a riprodurre correttamente il comportamento osservato nel dataset di addestramento [17]. Inoltre, per determinare quando un punto deve essere considerato anomalo, è necessario stabilire una soglia di errore. Una soglia troppo bassa potrebbe portare a falsi positivi, mentre una soglia troppo alta potrebbe non rilevare tutte le anomalie (trattazione completa nella Sezione 5.3).

2.2 Tipi di Anomalie

Le anomalie nelle serie temporali possono assumere forme diverse, e la loro classificazione è essenziale per poter selezionare gli algoritmi di rilevamento più appropriati. In letteratura sono state proposte diverse classificazioni, tra cui quelle di Choi et al. [9] e Chandola et al. [16], che identificano tre tipologie principali di anomalie: puntuali, collettive e contestuali. Questa classificazione, illustrata nella Figura 2.3 con esempi tratti dal dataset PSM [18], viene approfondita di seguito.

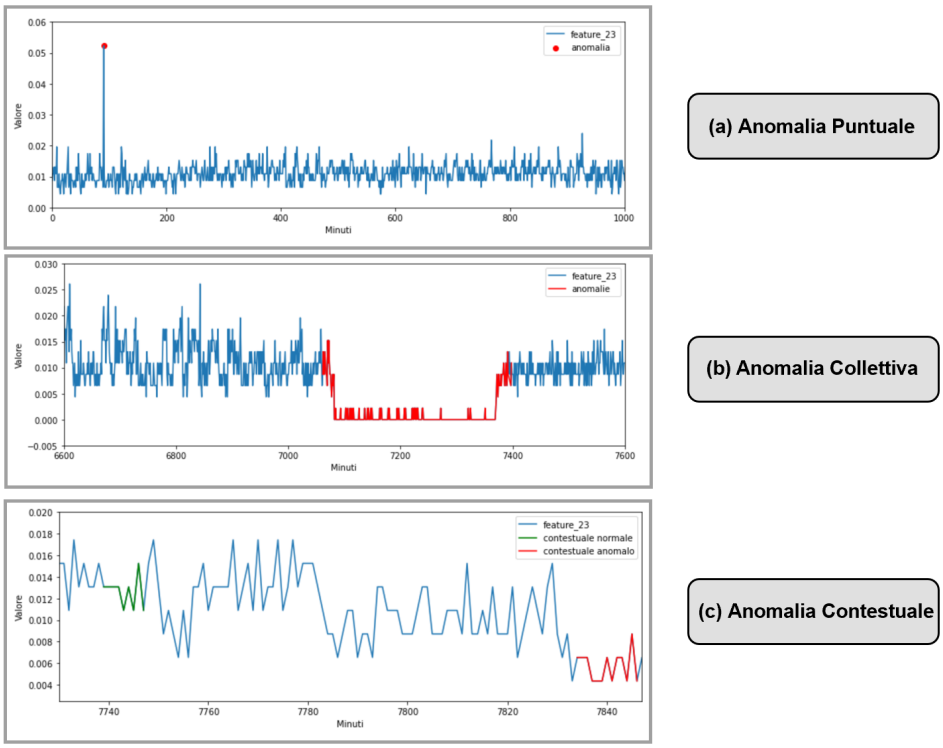

Figura 2.3: Tipologie di anomalie nelle serie temporali. Dati presi dal dataset PSM.

a) **Anomalia puntuale**:

Si tratta della forma più semplice di anomalia e si verifica quando un singolo punto (anche detto outlier) o una breve sequenza di dati si discosta significativamente dal normale andamento. Queste anomalie spesso sono identificabili come rumore temporale e sono generalmente dovute a errori di sensori o a improvvisi cambiamenti nelle condizioni operative del sistema.

Gli algoritmi per trattare questo tipo di anomalia possono essere anche semplici metodi statistici come lo Z-Score [19], [16], la Median Absolute Deviation (MAD) [19], [16] e l'Isolation Forest (IF) [20]. Le anomalie puntuali sono presenti in tutti i quattro dataset analizzati (Sezione 5.1).

b) **Anomalia collettiva**:

Le anomalie collettive si manifestano quando un insieme di punti dati forma un pattern inusuale, anche se i singoli valori all'interno del gruppo possono apparire normali. Un esempio è un output elettrocardiografico in cui un periodo di bassa attività cardiaca persiste anormalmente a lungo. Per identificare queste anomalie è necessario valutare i pattern a lungo termine e le interazioni tra i diversi punti dati.

Gli algoritmi per trattare questo tipo di anomalia possono includere i semplici metodi statistici citati sopra, tuttavia, i metodi più complessi descritti nel Capitolo 4 mostrano significativi miglioramenti nelle prestazioni nei dataset analizzati in tesi.

Le anomalie collettive sono presenti in tutti i quattro dataset analizzati (Sezione 5.1).

c) **Anomalia contestuale**:

Diversamente dalle anomalie puntuali, le anomalie contestuali dipendono dal contesto in cui si verificano, pertanto un dato può risultare normale in una situazione e anomalo in un'altra. Ad esempio, registrare una temperatura di 30° può essere del tutto plausibile durante l'estate, ma insolita in inverno. La difficoltà nel rilevarle deriva dal fatto che richiedono un'analisi del contesto, che può variare nel tempo e in base alle diverse condizioni operative.

Gli algoritmi per trattare questo tipo di anomalia possono essere anche le baseline descritte nella Sezione 4.1, tuttavia, i dataset analizzati in tesi mostrano significativi miglioramenti nelle prestazioni con i metodi più complessi descritti nelle sezioni successive del Capitolo 4. Le anomalie contestuali sono presenti in modo più evidente nel dataset PSM [18].

È importante notare che un'anomalia puntuale o collettiva potrebbe rientrare anche nella categoria delle anomalie contestuali. Ad esempio, in Figura 2.3, l'anomalia rappresentata in (c) è anche un'anomalia collettiva.

Oltre a questi tre tipi principali, le anomalie nelle serie temporali possono essere ulteriormente classificate, in base a specifiche definizioni di normalità, maggiori dettagli in Choi et al. [9].

2.3 Proprietà Serie Temporali

In questa sezione si analizzano più nello specifico le caratteristiche dei dati considerati, ovvero serie temporali [21], [22], [9]. Una serie temporale è una sequenza di osservazioni ordinate nel tempo, in cui ogni valore è associato a un istante o intervallo temporale specifico. Questo tipo di dato è fondamentale in numerose applicazioni. Infatti, il tempo rappresenta l'elemento caratterizzante dei fenomeni dinamici. Si introducono di seguito le proprietà e assunzioni più importanti riguardanti le serie temporali.

2.3.1 Dipendenza Temporale

Una serie temporale è una successione di osservazioni registrate in ordine cronologico, nelle quali si presume che ciascun valore sia influenzato, almeno in parte, dai valori precedenti. Si assume inoltre che le misurazioni vengano effettuate a intervalli regolari, come ogni minuto, caratteristica definita frequenza di campionamento.

Per quantificare la dipendenza temporale all'interno di una serie temporale, si utilizza la **funzione di autocorrelazione** (ACF) [21], [22], che misura il grado di correlazione lineare tra i valori della serie a diverse distanze temporali (lag). Matematicamente l'autocorrelazione al lag h è definita come:

$$\rho(h) = \frac{\mathbb{E}[\,(x_t - \mu)(x_{t-h} - \mu)\,]}{\sigma^2}\,, \tag{2.3}$$

dove:

- $\rho(h)$ rappresenta il coefficiente di autocorrelazione al lag h;
- x_t è il valore della serie temporale all'istante t;
- μ è la media della serie temporale, ovvero $\mathbb{E}[x_t]$;
- σ^2 è la varianza della serie temporale, definita come $\mathbb{E}[(x_t - \mu)^2]$;
- h è il ritardo temporale (lag), che indica la distanza temporale tra le osservazioni considerate.

La funzione di autocorrelazione fornisce informazioni sulla struttura di dipendenza della serie nel tempo, permettendo di individuare la presenza di trend e componenti stagionali (maggiori dettagli nella Sezione 2.3.5). In particolare, il grafico dell'autocorrelazione (correlogramma) rappresenta i valori di $\rho(h)$ in funzione del lag h, offrendo una lettura immediata della dinamica della serie. Un'autocorrelazione che decresce lentamente può indicare una componente di trend, mentre valori elevati di autocorrelazione anche per lag alti suggeriscono una radice unitaria (maggiori dettagli nella Sezione 4.1.1), ossia una dipendenza particolarmente persistente nel tempo. Al contrario, un coefficiente di autocorrelazione che diminuisce rapidamente a zero indica che i valori passati influenzano poco quelli futuri, suggerendo un'assenza di dipendenza temporale. Infine, la presenza di picchi regolari a determinati lag può segnalare una componente stagionale.

In altre parole, se la funzione di autocorrelazione (ACF) decresce rapidamente, tipicamente entro pochi lag, la serie può essere considerata stazionaria. Avere valori alti di $\rho(h)$ per piccoli h non implica automaticamente una non stazionarietà, ma se l'autocorrelazione decresce lentamente, allora la serie potrebbe essere non stazionaria e richiedere trasformazioni come la differenziazione per stabilizzarla (maggiori dettagli nella Sezione 3.4).

Un esempio di grafico dell'autocorrelazione (correlogramma) è mostrato in Figura 2.4, che rappresenta i primi 50 lag della serie temporale illustrata in Figura 2.5. Questo esempio include bande di confidenza al 95%, oltre le quali i valori di autocorrelazione sono considerati statisticamente significativi; per ulteriori dettagli si rimanda a Hyndman et al. [22]).

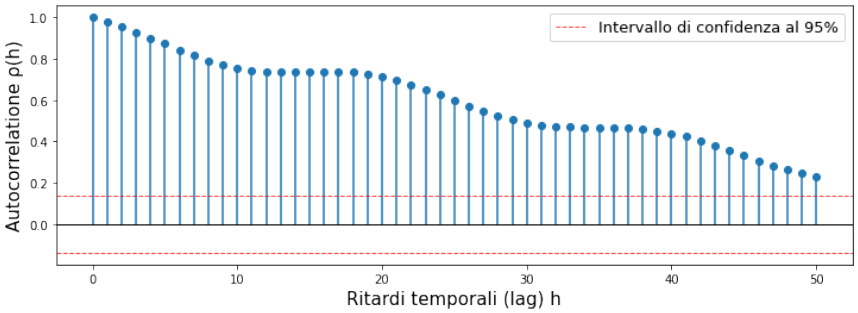

Figura 2.4: Esempio di correlogramma.

2.3.2 Dimensionalità: Serie Univariate e Multivariate

Il numero di variabili registrate per ciascuna osservazione nel tempo è detto dimensionalità. Questa caratteristica suddivide le serie temporali in serie univariate e multivariate.

- **Serie temporali univariate**:

  Una serie temporale univariata $\mathbf{x}$ è una sequenza di misure raccolte nel tempo, relative a una singola variabile vettoriale, in formula:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_t \\ \vdots \\ x_T \end{bmatrix} .$$

  L'unica dipendenza che si può studiare in questo caso è quella temporale, cioè la dipendenza di x_t da $x_{t-1}, \ldots, x_1$. Essa si misura con la funzione di autocorrelazione, definita nella Sottosezione precedente 2.3.1. Un esempio comune di serie storica univariata è il valore di un titolo azionario misurato nel tempo. In Figura 2.5 si può vedere un esempio generato artificialmente.

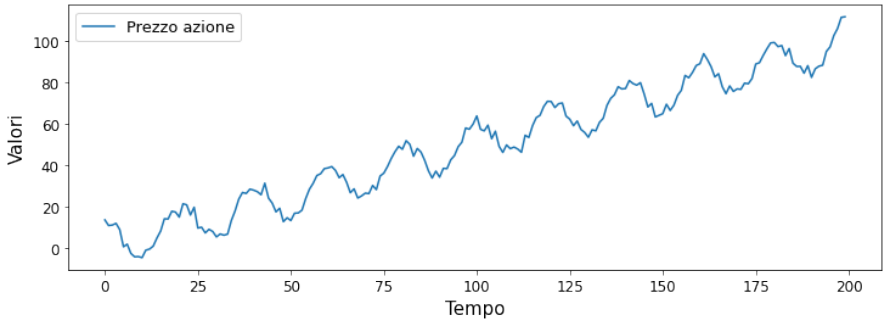

Figura 2.5: Prezzo di un'azione che varia nel tempo generata artificialmente.

- **Serie temporali multivariate**:

  Una serie temporale N-variata $\mathbf{X}$ è una raccolta sequenziale di osservazioni relative a N variabili in T istanti di tempo. Ogni sequenza di osservazioni è quindi un vettore multidimensionale (o matrice):

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{T,1} & x_{T,2} & \cdots & x_{T,N} \end{bmatrix},$$

  con T righe e N colonne. L'analisi di serie multivariate, pertanto, deve considerare non solo le relazioni temporali delle singole colonne con i $t-1$ istanti precedenti, ma anche le correlazioni tra le N diverse colonne misurate in t. Per esempio, nel monitoraggio industriale, si possono misurare simultaneamente pressione, temperatura e voltaggio, come mostrato in Figura 2.6.

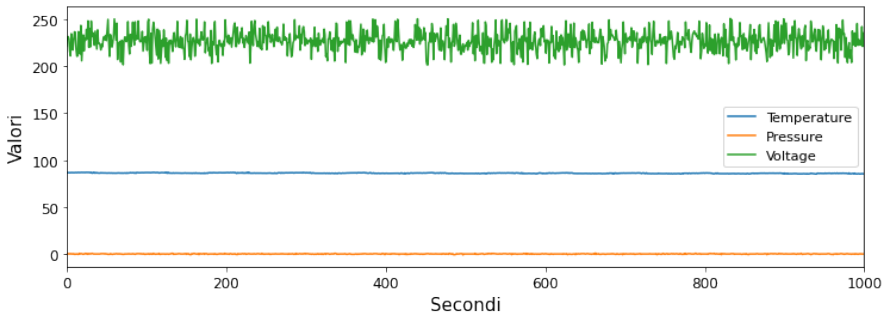

Figura 2.6: Pressione, temperatura e voltaggio esempio presi dal dataset SKAB.

2.3.3 Stazionarietà

La stazionarietà [21], [22] di una serie temporale è una proprietà che implica che determinate caratteristiche della serie (elencate successivamente) rimangano costanti nel tempo. Per l'analisi delle serie temporali rappresenta una condizione fondamentale, poiché molti modelli statistici la richiedono per garantire che le previsioni siano affidabili e non influenzate da variazioni nel tempo. In letteratura [22], [23] si distinguono due principali tipologie di stazionarietà, la stazionarietà debole e la stazionarietà forte, definite di seguito.

Definizione - Stazionarietà debole (o di covarianza)

Una serie temporale univariata $\mathbf{x}_t$ è stazionaria in senso debole se:

- La sua media è costante nel tempo: $E(\mathbf{x}_t) = \mu$.
- La sua varianza è costante nel tempo e finita: $\mathrm{Var}(\mathbf{x}_t) = \sigma^2$.

- La sua covarianza tra $\mathbf{x}_t$ e $\mathbf{x}_{t+h}$ dipende solo da h, la differenza temporale tra le osservazioni, e non da t, ovvero: $\text{Cov}(\mathbf{x}_t, \mathbf{x}_{t+h}) = f(h)$, dove f indica una funzione generica.

Questa forma di stazionarietà è quella più frequentemente richiesta nei modelli statistici, come l'ARIMA (Sezione 4.1.1), in cui si ipotizza che i processi siano stazionari per garantire la validità delle previsioni.

Definizione - Stazionarietà forte

Una serie temporale è stazionaria in senso forte se la distribuzione congiunta di qualsiasi sottoinsieme di variabili osservate nella serie non cambia nel tempo. Formalmente, una serie temporale univariata $\mathbf{x}_t$ è stazionaria in senso forte se per qualsiasi $t_1, t_2, ..., t_T$, la distribuzione di probabilità congiunta di $(x_{t_1}, x_{t_2}, ..., x_{t_T})$ è la stessa di $(x_{t_1+h}, x_{t_2+h}, ..., x_{t_T+h})$, per ogni h. In altre parole, le proprietà stocastiche della serie sono indipendenti dal tempo.

Nella tesi si considerano principalmente serie multivariate, perciò è necessario estendere il concetto di stazionarietà **in più dimensioni**. Una serie temporale multivariata è considerata stazionaria in senso debole o forte se tutte le sue N colonne sono stazionarie in senso debole o forte, rispettivamente [24].

Successivamente, nella Sezione 4.1.1, si definiscono i test statistici per verificare la stazionarietà di una serie temporale, e i metodi possibili per rendere stazionaria una serie temporale che inizialmente non è stazionaria.

In molte applicazioni le serie temporali risultano non stazionarie, principalmente a causa della presenza di:

- **Stagionalità**: andamenti periodici che si ripetono regolarmente, ad esempio le vendite di gelati d'estate e d'inverno.
- **Trend**: variazione a lungo termine dei valori medi delle osservazioni nel tempo. Indica l'andamento generale dei dati, che può essere in aumento, in diminuzione o stabile. Un esempio di trend positivo è l'aumento dei consumi di energia quando il numero di abitanti di un comune è in crescita.
- **Changepoint**: variazione improvvisa nelle condizioni operative di un sistema, come l'apertura o chiusura di una valvola da cui dipendono diverse variabili.

2.3.4 Rumore

Le serie temporali, soprattutto nei sistemi industriali, spesso presentano variazioni indesiderate dovute a interferenze o errori nei sistemi di misurazione. Nel campo dell'elaborazione dei segnali, il rumore [9] rappresenta proprio queste fluttuazioni casuali che si manifestano durante le fasi di acquisizione, memorizzazione, trasmissione o elaborazione dei dati. Sebbene il rumore derivi spesso da lievi variazioni nella sensibilità dei sensori

e non alteri la struttura dei dati, può diventare problematico quando la distinzione tra rumore e anomalia è complessa. È quindi importante applicare tecniche adeguate per ridurre il rumore durante la fase di preprocessing dei dati. Queste tecniche aiutano a isolare i segnali rilevanti, migliorando la qualità della serie, e sono trattate nella Sezione 3.3. Segue in Figura 2.7 un esempio di serie temporale del dataset SKAB [25] che presenta del rumore, e che verrà riproposto nella Sezione 3.3 con un filtro apposito per rimuoverlo.

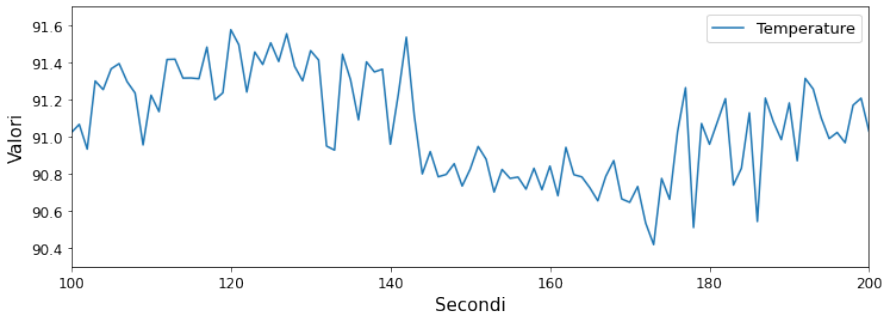

Figura 2.7: Esempio di serie temporale che presenta rumore. Variabile *Temperature* del dataset SKAB

2.3.5 Decomposizione STL

Dopo aver discusso delle principali cause di non stazionarietà nelle serie temporali e aver definito il rumore, è utile introdurre le **tecniche di decomposizione** [22]. Esse permettono di scomporre una serie nelle sue principali componenti:

- **Componente di trend** $T(t)$;
- **Componente stagionale** $S(t)$;
- **Componente residua** $R(t)$: rappresenta il rumore, che non è spiegato dalle altre componenti.

Matematicamente, una serie temporale $\mathbf{x}_t$ può essere espressa in forma additiva:

$$\mathbf{x} = T + S + R\,, \tag{2.4}$$

o moltiplicativa, se le componenti interagiscono in maniera proporzionale:

$$\mathbf{x} = T \cdot S \cdot R\,. \tag{2.5}$$

Tra i diversi metodi di decomposizione delle serie temporali, la **decomposizione STL** (Seasonal and Trend decomposition using Loess) [26], [22] si distingue per la sua flessibilità ed è ampiamente utilizzata. Questo metodo non parametrico, basato su regressioni

locali (Loess), è in grado di separare le componenti stagionali e di trend con diversi vantaggi. STL si distingue per la sua flessibilità nella gestione delle stagionalità variabili e per la possibilità di personalizzare il livello di smoothing, adattandosi a serie temporali complesse; è anche robusto al rumore e agli outlier.

La decomposizione STL presenta anche alcune limitazioni, come la difficoltà di gestire automaticamente gli effetti stagionali come le variazioni mensili e la compatibilità esclusiva con decomposizioni additive. Tuttavia, applicando il logaritmo, la relazione moltiplicativa si trasforma in una relazione additiva:

$$\log(\mathbf{x}) = \log(T \cdot S \cdot R) = \log(T) + \log(S) + \log(R)\,. \tag{2.6}$$

In questo modo la decomposizione STL può essere applicata per stimare le componenti additive $\log(T_t)$, $\log(S_t)$ e $\log(R_t)$, che, una volta calcolate, possono essere riconvertite nella forma originale utilizzando la funzione esponenziale.

Il processo di decomposizione avviene attraverso un ciclo iterativo che alterna il calcolo della componente stagionale, del trend e dei residui. Utilizza come input la serie temporale, la lunghezza del periodo stagionale s (ad esempio, 12 per dati mensili con stagionalità annuale) e i parametri di smoothing. Con questi input, la decomposizione segue i passaggi:

1. **Calcolo della stagionalità** S: si utilizzano medie locali per identificare e isolare i pattern periodici a intervalli regolari, corrispondenti alla lunghezza del periodo s.

2. **Rimozione della stagionalità**: la serie originale viene *deseasonalizzata* sottraendo la componente stagionale calcolata al passo precedente, ottenendo $\mathbf{x} - S$, in cui rimangono il trend e il rumore.

3. **Calcolo della componente di trend** T: si applica un filtro *Loess* (smoothing locale) alla serie deseasonalizzata, per stimare la componente di trend.

4. **Calcolo del residuo** R: sottraendo sia la componente stagionale S sia il trend T dalla serie originale $\mathbf{x}$, si ottiene la componente residua, che rappresenta il rumore:

$$R = \mathbf{x} - S - T\,.$$

5. **Aggiornamento della stagionalità**: sui residui, si ricalcola la stagionalità raffinata attraverso un altro ciclo di *smoothing Loess*.

Il processo viene iterato più volte per affinare il calcolo di S, T e R.

In Figura 2.8 si può vedere un esempio di applicazione della decomposizione STL sugli stessi dati di Figura 2.5.

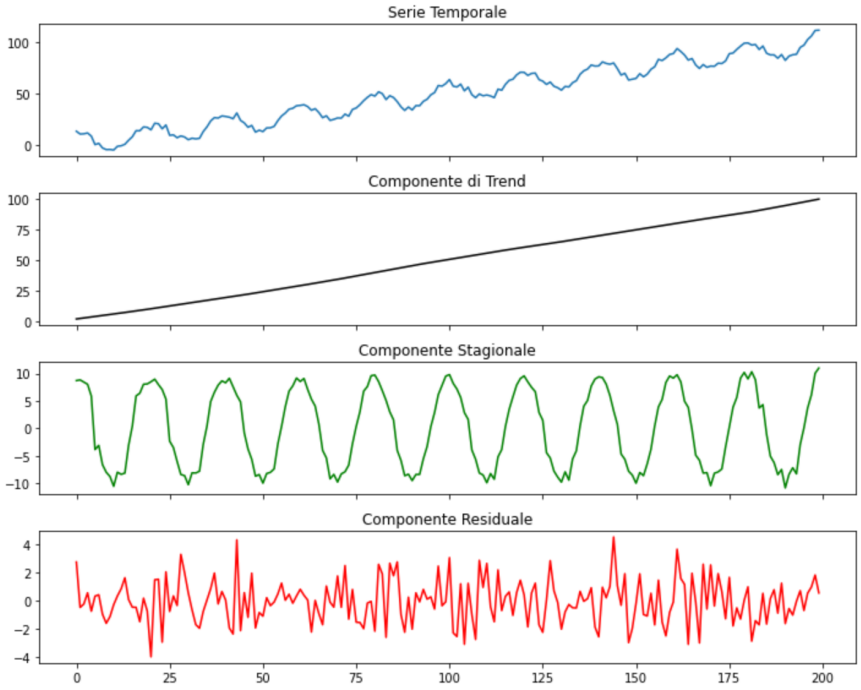

Figura 2.8: Esempio di decomposizione STL.

Capitolo 3

Preprocessing

Il preprocessing dei dati [14] rappresenta una fase cruciale in ogni pipeline di machine learning, in quanto ha l'obiettivo di garantire che i dati siano adeguatamente elaborati per l'analisi successiva. Questa fase non solo migliora la qualità complessiva dei dati, ma consente anche di eliminare informazioni superflue o rumorose che potrebbero compromettere le prestazioni dei modelli. Attraverso un insieme di tecniche specifiche, il preprocessing affronta le specifiche caratteristiche e le sfide presenti nei dati, tra cui i valori mancanti, gli outlier e le scale non uniformi, garantendo una maggiore robustezza dei risultati, come evidenziato nei risultati presentati nel Capitolo 6.

In questa tesi sono state impiegate diverse tecniche di preprocessing. Più precisamente, il procedimento inizia con il caricamento dei dati, il controllo dei valori mancanti e la rimozione delle colonne costanti. Successivamente, è stata effettuata la riduzione del rumore, che ha permesso di pulire i dati. In seguito, è stata applicata la normalizzazione per uniformare le scale delle variabili, seguita dalla feature selection per rimuovere le variabili ridondanti e ridurre le dimensioni del dataset. La Figura 3.1 illustra le fasi di preprocessing sopracitate, che sono state testate per ottimizzare le prestazioni e che sono descritte in dettaglio di seguito.

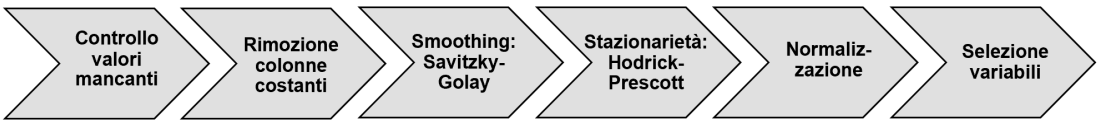

Figura 3.1: Pipeline di preprocessing.

3.1 Caricamento Dati e Controllo Valori Mancanti

Per preparare il dataset all'analisi, si esegue il caricamento dei dati e un controllo preliminare per identificare la presenza di valori mancanti. I valori mancanti sono un problema comune quando si lavora con le serie temporali e possono verificarsi, ad esempio, a causa

di malfunzionamenti del sistema. Le tecniche di data imputation possono essere utilizzate per colmare le lacune nelle serie temporali in cui si verificano i valori mancanti. In questo studio è stata utilizzata un'interpolazione spline di ordine 2 o 3 per le sequenze mancanti fino a 5 punti consecutivi. Per le sequenze più lunghe di 5 punti, il segmento con i valori mancanti è stato rimosso, poiché si rischiava di introdurre valori errati. Di seguito è riportato un esempio grafico di come sono stati rimossi i valori mancanti (dataset di riferimento: PSM [18]). La prima sequenza di NaN è stata eliminata poiché comprende più di 5 valori consecutivi. La seconda sequenza, composta da 4 valori consecutivi, è stata invece interpolata con una spline di ordine 3.

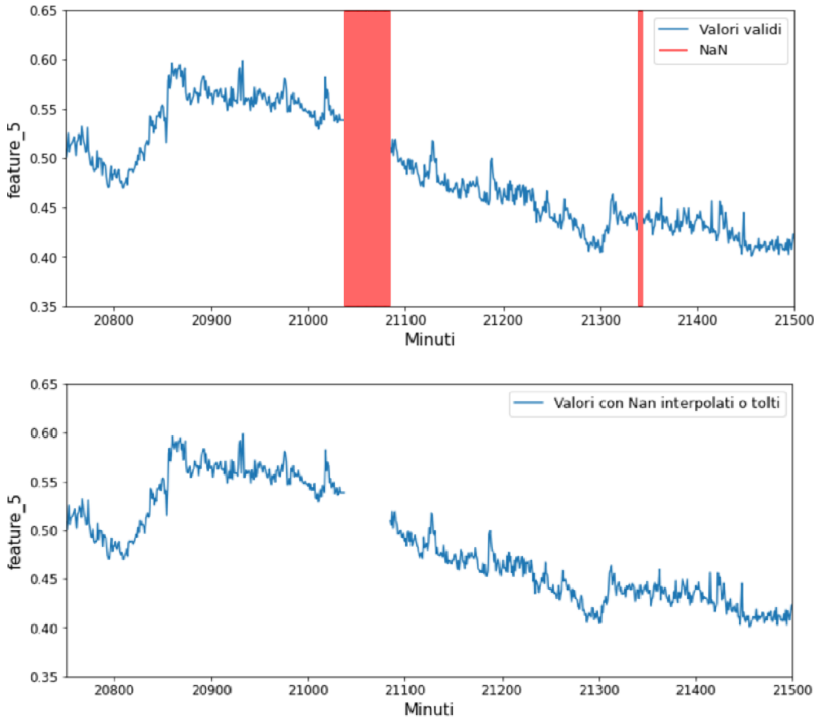

Figura 3.2: Esempio di data imputation.

3.2 Rimozione colonne costanti

Le colonne costanti, ossia quelle che mantengono costante un valore in tutte le osservazioni, non contribuiscono all'obiettivo di determinare la variabile target. Un algoritmo di apprendimento automatico non può imparare a distinguere tra diverse classi o prevedere valori sulla base di una feature che rimane sempre identica. Tale approccio è stato convalidato negli esperimenti condotti in questo lavoro e in quello di Kim et al. [27], dove la rimozione delle colonne costanti ha permesso di concentrarsi sulle colonne essenziali per un efficace rilevamento delle anomalie. Il mantenimento di tali colonne, infatti, non solo

aumenta inutilmente la dimensionalità del dataset, ma può anche portare a processi di selezione o modellazione delle caratteristiche fuorvianti.

3.3 Riduzione del Rumore

Le serie temporali dei processi industriali sono spesso soggette a rumore [9] causato da fluttuazioni casuali e da errori di misurazione, come descritto più nel dettaglio nella Sottosezione 2.3.4. Tali elementi possono compromettere l'accuratezza dei modelli analitici. Per questo motivo è importante applicare tecniche di riduzione del rumore per isolare i segnali rilevanti, con l'obiettivo di migliorare la qualità dei dati e facilitare l'analisi successiva. Nel presente studio sono state esaminate tre tecniche comuni di smoothing per la riduzione del rumore: il filtro mediano, il filtro passa-basso e il filtro Savitzky-Golay.

3.3.1 Filtro Mediano

Il filtro mediano [28] è una tecnica di smoothing particolarmente efficace nel ridurre il rumore impulsivo, caratterizzato da picchi improvvisi, nelle serie temporali. La procedura consiste nel sostituire ogni valore con la mediana dei valori contenuti all'interno di una finestra mobile centrata su di esso. Matematicamente, dato un segnale discreto x_t e una finestra di lunghezza dispari $2k+1$, il filtro mediano si esprime come:

$$x'_t = \text{mediana}(x_{t-k},\, x_{t-(k-1)},\, \ldots,\, x_t,\, \ldots,\, x_{t+(k-1)},\, x_{t+k})\,, \tag{3.1}$$

dove x'_t rappresenta il segnale filtrato e mediana$(\cdot)$ indica l'operazione di mediana. Tuttavia, è importante notare che il filtro mediano presenta alcune limitazioni. Se il rumore è distribuito in modo continuo, ovvero se non è costituito da picchi isolati ma è diffuso lungo il segnale, il filtro può risultare inefficace. Inoltre, in presenza di variazioni molto brusche nel segnale, può introdurre distorsioni, smussando eccessivamente i dettagli rilevanti.

3.3.2 Filtro Passa-Basso

Il filtro passa-basso [28] agisce come un ammortizzatore per le fluttuazioni rapide, consentendo solo le variazioni a bassa frequenza di attraversare il filtro, riducendo così il rumore. L'intensità del filtro è determinata dalla frequenza di taglio f_c, che stabilisce quali frequenze vengono mantenute e quali attenuate. Un classico filtro passa-basso può essere implementato tramite un filtro di tipo media mobile esponenziale. Una rappresentazione tipica è il filtro passa-basso ideale, definito in termini di risposta in frequenza:

$$H(f) = \begin{cases} 1, & |f| \leq f_c \\ 0, & |f| > f_c \end{cases}\,, \tag{3.2}$$

dove f_c è la frequenza di taglio e $H(f)$ è la funzione di trasferimento del filtro. Nel dominio del tempo, una comune approssimazione discreta è il filtro passa-basso di primo ordine con media mobile esponenziale:

$$x'_t = \alpha x_t + (1 - \alpha)x'_{t-1} \ , \tag{3.3}$$

dove x'_t è il segnale filtrato, x_t è il segnale in ingresso e α $(0 < \alpha < 1)$ è un parametro che controlla la risposta del filtro, determinando la frequenza di taglio. Questo tipo di filtro è definito di primo ordine in quanto il valore corrente x'_t dipende unicamente dal valore immediatamente precedente x'_{t-1}, creando una relazione ricorsiva basata su una sola differenza temporale. Il termine media mobile esponenziale deriva dal fatto che i pesi assegnati ai valori passati decadono esponenzialmente nel tempo, come si può osservare espandendo iterativamente la formula ricorsiva del filtro.

Il parametro α determina il comportamento del filtro $H(f)$ e della frequenza di taglio effettiva f_c. La funzione di trasferimento del filtro passa-basso con media mobile esponenziale è espressa dalla seguente formula:

$$H(f) = \frac{\alpha}{1 - (1 - \alpha)e^{-i2\pi fT}} \ , \tag{3.4}$$

dove il termine $e^{-i2\pi fT}$ rappresenta un'onda sinusoidale complessa e i è l'unità immaginaria. L'espressione mostra come la selezione di α determini la risposta in frequenza del filtro, influenzando il grado di riduzione delle componenti ad alta frequenza. Se si sceglie α troppo piccolo, il filtro sarà molto smorzante, taglierà molte frequenze alte e potrebbe rimuovere dettagli importanti del segnale. Se si sceglie α troppo grande, il filtro risulterà meno efficace nel ridurre il rumore e lascerà passare troppe alte frequenze indesiderate.

3.3.3 Filtro di Savitzky-Golay

Il filtro di Savitzky-Golay [29] è una tecnica di smoothing che applica un'approssimazione polinomiale locale ai dati all'interno di una finestra mobile. Questo significa che, per ogni sottoinsieme di punti dati compreso nella finestra, viene eseguito un fitting di un polinomio di ordine specificato. Il valore centrale della finestra viene poi sostituito con il valore generato dal polinomio approssimato, smussando le fluttuazioni presenti nei dati, ma preservando le caratteristiche principali del segnale. Questo tipo di filtro è particolarmente utile per ridurre il rumore casuale e ad alta frequenza.

Matematicamente il valore filtrato x'_i di un dato punto x_i si ottiene tramite la convoluzione con un insieme di coefficienti c_j ottenuti dal fitting polinomiale:

$$x'_i = \sum_{j=-m}^{m} c_j x_{i+j} \ , \tag{3.5}$$

dove:

- x_{i+j} sono i punti dati nella finestra centrata in x_i;
- c_j sono i coefficienti di convoluzione calcolati minimizzando l'errore quadratico tra i valori reali e il polinomio di approssimazione;

- m indica la metà della finestra di smoothing, dove la lunghezza totale della finestra è $l = 2m + 1$.

Più precisamente, i coefficienti c_j vengono calcolati scegliendo il polinomio approssimante di grado k:

$$P(x) = \sum_{n=0}^{k} a_n x^n \,, \tag{3.6}$$

dove i coefficienti a_n sono stimati minimizzando l'errore quadratico tra i dati osservati e il polinomio.
L'elemento centrale del polinomio approssimato, $P(0)$, determina i coefficienti di convoluzione c_j secondo la seguente formula:

$$P(0) = \sum_{j=-m}^{m} c_j x_{i+j} \,, \tag{3.7}$$

dove i coefficienti c_j sono calcolati esplicitamente risolvendo il problema di regressione polinomiale ai minimi quadrati.
La **scelta finale** è ricaduta sul **filtro di Savitzky-Golay**, che è stato preferito per diversi motivi. Innanzitutto, offre una maggiore flessibilità grazie alla possibilità di regolare parametri come l'ordine del polinomio k e la dimensione della finestra l, adattandosi meglio alle caratteristiche specifiche della serie. A differenza del filtro mediano, che altera la struttura del segnale, e del filtro passa-basso, che può causare una perdita di dettagli, il filtro di Savitzky-Golay riduce solamente il rumore ad alta frequenza. Inoltre, l'approssimazione polinomiale locale lo rende particolarmente robusto in scenari in cui è necessario preservare le caratteristiche del segnale. Come evidenziato in Figura 3.3, che riproduce un esempio del funzionamento del filtro al dataset SKAB [25], il filtro di Savitzky-Golay si dimostra efficace, affidabile e flessibile.

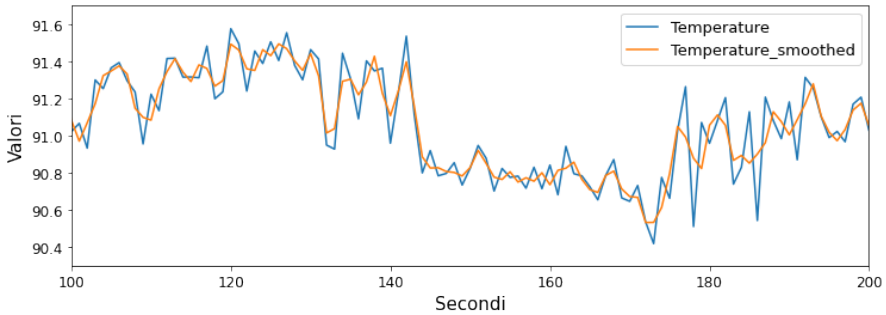

Figura 3.3: Esempio di applicazione del filtro Savitzky-Golay alla variabile *Temperature* del dataset SKAB.

3.4 Verifica e Trattamento della Stazionarietà

Dopo aver introdotto nel Capitolo 2 le nozioni di stazionarietà forte e debole, si presentano ora i metodi per verificare se una serie temporale è stazionaria e, in caso contrario, le tecniche per trasformarla in una serie stazionaria. Per **verificare la stazionarietà**, si possono utilizzare test statistici come il test di Dickey-Fuller e il test KPSS [22].

Test di Dickey-Fuller (ADF - Augmented Dickey-Fuller)

Il test di Dickey-Fuller verifica l'ipotesi nulla che una serie temporale contenga una radice unitaria, indicando così che è non stazionaria. Se l'ipotesi nulla viene respinta, la serie è stazionaria. Il test viene eseguito stimando la seguente equazione:

$$\Delta x_t = \alpha + \beta t + \gamma x_{t-1} + \delta_1 \Delta x_{t-1} + ... + \delta_k \Delta x_{t-k} + \epsilon_t \,, \tag{3.8}$$

dove:

- x_t è la serie temporale osservata;
- $\Delta x_t = x_t - x_{t-1}$ è la differenza di primo ordine della serie;
- α è una costante;
- βt rappresenta una tendenza lineare;
- γ è il coefficiente che determina se c'è una radice unitaria (cioè, se $\gamma = 0$);
- k è il numero di lag inclusi nel modello per controllare l'autocorrelazione;
- δ_i sono i coefficienti che pesano le differenze ritardate Δx_{t-i}, con $i = 1, \ldots, k$;
- ϵ_t è un termine di errore stocastico.

Test KPSS (Kwiatkowski-Phillips-Schmidt-Shin)

Il test KPSS verifica se una serie temporale è stazionaria intorno a una media o a una tendenza deterministica. A differenza del test di Dickey-Fuller, nel test KPSS l'ipotesi nulla assume che la serie sia stazionaria, mentre l'ipotesi alternativa che non sia stazionaria. Questo test si basa sulla regressione:

$$x_t = r_t + z_t \,, \tag{3.9}$$

dove x_t è la serie temporale osservata, r_t è una componente deterministica (ad esempio un trend lineare) e z_t è un errore stocastico che può essere modellato come una random walk.

Quando una serie temporale non è stazionaria è possibile applicare varie **trasformazioni** per renderla stazionaria. Queste tecniche sono progettate per eliminare tendenze, stagionalità o ridurre la varianza all'interno dei dati. Di seguito vengono presentate alcune delle tecniche utilizzate [22]:

3.4.1 Differenziazione

La differenziazione [22] è forse la tecnica più utilizzata per rimuovere i trend in una serie temporale non stazionaria. Si basa sulla sottrazione dei valori successivi, trasformando la serie in una differenza prima. Questo processo riduce le tendenze a lungo termine che potrebbero influire sulla stazionarietà.
La differenziazione è definita come:

$$x'_t = x_t - x_{t-1}\,, \tag{3.10}$$

dove x'_t è la serie differenziata e x_t è il valore della serie originale al tempo t.

3.4.2 Trasformazioni logaritmiche e Box-Cox

Le trasformazioni logaritmiche e Box-Cox [22] sono spesso impiegate per rendere stazionaria una serie temporale, in quanto aiutano a stabilizzare la varianza nel tempo. Tuttavia, quando si vogliono individuare le anomalie, questa operazione può essere dannosa, in quanto variazioni significative della varianza possono essere un segnale di anomalia [30]. Ad esempio, un'improvvisa variazione della temperatura potrebbe segnalare un guasto nel sistema e, applicando queste trasformazioni, si perderebbe un'informazione importante per la manutenzione predittiva.

3.4.3 Filtro di Hodrick-Prescott

Il Filtro di Hodrick-Prescott (HP) [31] è una tecnica di decomposizione delle serie temporali comunemente usata per separare la componente ciclica da quella di trend. Il filtro agisce identificando una rappresentazione del trend più *liscia* rispetto alla serie di partenza, mantenendo separate le componenti cicliche.
Il filtro si basa sull'ottimizzazione della seguente funzione obiettivo:

$$\min_{\tau} \left[\sum_{t=1}^{T} (\mathbf{x}_t - \tau_t)^2 + \lambda \sum_{t=2}^{T-1} \left((\tau_{t+1} - \tau_t) - (\tau_t - \tau_{t-1}) \right)^2 \right], \tag{3.11}$$

dove $\mathbf{x}_t$ è il valore osservato della serie temporale al tempo t, τ_t è il trend al tempo t e λ è un parametro di smoothing che controlla il bilanciamento tra la fedeltà ai dati di partenza e la regolarità di τ_t.
La funzione obiettivo si compone di due parti:

- **Termine di fitting**:
$$\sum_{t=1}^{T} (\mathbf{x}_t - \tau_t)^2.$$
Questo termine misura quanto i valori del trend τ_t si discostano dai valori osservati $\mathbf{x}_t$. Minimizzare questo termine significa cercare di far sì che il trend segua il più possibile i dati.

- **Termine di penalità sulla curvatura**:

$$\lambda \cdot \sum_{t=2}^{T-1} \left((\tau_{t+1} - \tau_t) - (\tau_t - \tau_{t-1})\right)^2$$

  Questo secondo termine penalizza, tramite la penalità λ, le grandi variazioni nel trend τ_t, assicurando che esso sia più regolare. Minimizzare questo termine serve per avere un trend che vari più lentamente, evitando oscillazioni brusche.

Il parametro λ, che è una penalità, controlla il grado di smoothing ed è da scegliere accuratamente tenendo conto che:

- Se λ è piccolo, allora il trend stimato sarà molto vicino alla serie osservata $\mathbf{x}_t$, seguendone anche le piccole oscillazioni.

- Se λ è grande, allora il trend sarà molto liscio e varierà solo lentamente nel tempo, trascurando le oscillazioni a breve termine.

Hodrick e Prescott [31] suggeriscono 1600 come valore di λ per i dati trimestrali. Ravn e Uhlig [32], invece, sostengono che tale parametro debba variare in base alla quarta potenza del rapporto di frequenza di osservazione. In base a tale teoria, il valore di λ dovrebbe corrispondere a 6.25 ($\frac{1600}{4^4}$) per i dati annuali e a 129600 ($1600 * 3^4$) per i dati mensili. Tuttavia, nella pratica, si utilizza comunemente il valore di $\lambda = 100$ per i dati annuali e il valore di $\lambda = 14400$ per i dati mensili. Nella Tabella 6.1, vengono riportati i valori di λ provati negli esperimenti.

Il filtro di Hodrick-Prescott, comunemente impiegato in macroeconomia per separare il ciclo economico dal trend di lungo periodo, si è rivelato efficace anche nell'analisi del dataset considerato, contribuendo alla rimozione dei trend. In questo studio è stato adottato come principale metodo per rendere le variabili stazionarie. Questa scelta è legata alla sua flessibilità, in quanto include un parametro regolabile e ottimizzabile, λ, che controlla il grado di smorzamento del trend, permettendo un miglior controllo sul processo di decomposizione. La Figura 3.4 mostra l'effetto del filtro applicato alla variabile *Temperature* del dataset SKAB [25], confrontando la serie originale con quella trasformata.

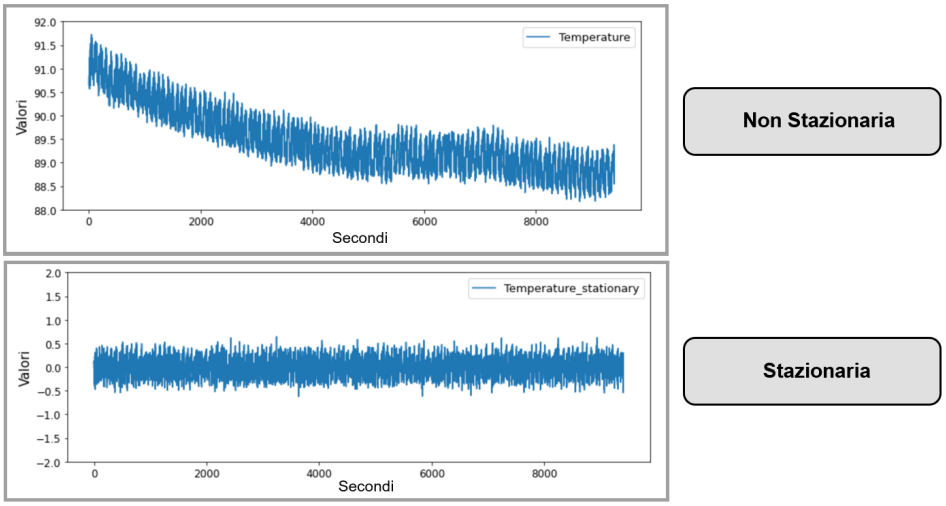

Figura 3.4: Esempio di applicazione del filtro Hodrick-Prescott.

3.5 Normalizzazione

La normalizzazione [14], [33] è la fase di preprocessing che uniforma le scale delle variabili, evitando che differenze di ordine di grandezza influenzino eccessivamente il processo di apprendimento. Nell'analisi delle componenti principali (PCA) (trattata nella sezione 4.1.2), questo passaggio è necessario per garantire che le variabili contribuiscano equamente all'individuazione delle direzioni di massima varianza, evitando distorsioni che potrebbero attribuire maggiore importanza a certe variabili solo a causa della loro scala. Inoltre, la normalizzazione aiuta a migliorare la stabilità numerica e accelera la convergenza degli algoritmi di discesa del gradiente, risultando particolarmente utile nelle reti neurali. Sono state scalate solamente le **colonne non booleane**. Le variabili booleane, infatti, hanno valori limitati tra 0 e 1 e la loro scala è già uniforme e coerente rispetto al range di valori. Normalizzare anche queste colonne potrebbe alterare il loro significato, introducendo distorsioni e senza apportare benefici al modello. I risultati delle prove condotte indicano che la normalizzazione di tali colonne non ottimizza le prestazioni del modello, ma in diverse circostanze le peggiora. Nel corso di questo studio, sono state testate e successivamente confrontate (Capitolo 6) le seguenti tecniche di normalizzazione dei dati.

3.5.1 Min-Max Scaling

Il min-max scaling [33] è una tecnica di normalizzazione che ridimensiona i valori di ciascuna variabile in un intervallo specifico; generalmente si sceglie l'intervallo [0, 1]. La formula per applicare questo tipo di normalizzazione è la seguente:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}, \tag{3.12}$$

dove x' è il valore normalizzato, x quello originale, mentre $x_{\min}$ e $x_{\max}$ rappresentano, rispettivamente, il minimo e il massimo della variabile.

Questa tecnica permette di ridimensionare i valori senza modificarne il rapporto reciproco, preservando così la loro distribuzione relativa. Tuttavia, il min-max scaling è sensibile agli outlier, poiché i valori estremi possono distorcere la scala e alterare la distribuzione dei dati trasformati. Per questo motivo, è più adatto a dataset privi di variazioni estreme e in situazioni in cui i valori minimi e massimi delle variabili sono noti a priori.

3.5.2 Standardizzazione (Z-Score)

La standardizzazione [33], nota anche come *Z-score normalization*, è un metodo di normalizzazione che trasforma i dati in modo che abbiano una distribuzione con media zero e deviazione standard pari a uno. Questo processo consente di ridurre l'influenza di valori estremi e rende i dati confrontabili. La formula per applicare la standardizzazione è la seguente:

$$z = \frac{x - \mu}{\sigma}, \tag{3.13}$$

dove z è il valore standardizzato, x è il valore originale, μ è la media della variabile, e σ è la deviazione standard della variabile.

La standardizzazione è particolarmente indicata per gli algoritmi che presuppongono una distribuzione gaussiana dei dati. A differenza del Min-Max Scaling, è meno influenzata dagli outlier, poiché si basa sulla media e sulla deviazione standard invece che sui valori estremi. Questa tecnica è quindi molto efficace per dataset in cui i valori possono variare ampiamente e non è garantito che siano distribuiti uniformemente.

3.5.3 Robust Scaling

Il robust scaling [14] è una tecnica di normalizzazione progettata per rendere i dati meno sensibili agli outlier. A differenza della standardizzazione tradizionale che utilizza la media e la deviazione standard, il Robust Scaling si basa sulla mediana e sull'intervallo interquartile (IQR) per scalare i dati. La formula per applicare il Robust Scaling è la seguente:

$$x' = \frac{x - Q_2}{\mathrm{IQR}(x)}, \tag{3.14}$$

dove:

- x' è il valore normalizzato;
- x è il valore originale;
- Q_2 è la mediana (50° percentile);

- IQR(x) è il range interquartile dei dati ed è definito come: IQR(x) = $Q_3 - Q_1$, dove Q_1 è il primo quartile (25° percentile) e Q_3 è il terzo quartile (75° percentile).

Il robust scaling si è dimostrato efficace nell'elaborazione di dati soggetti a outlier significativi, in quanto limita l'influenza di questi ultimi sulla distribuzione dei dati, prevenendo distorsioni legate alla media e alla deviazione standard. Inoltre risulta efficace anche per dataset con distribuzioni non gaussiane. Tuttavia, a differenza di altre tecniche di normalizzazione come il min-max scaling, non comprime i valori entro un intervallo fisso, quindi le variabili trasformate possono assumere valori al di fuori degli intervalli [0, 1] o [−1, 1], aspetto che potrebbe non essere ideale per alcuni modelli.

3.6 Feature Selection

La feature selection, ultima fase della pipeline di preprocessing in Figura 3.1, ha lo scopo di individuare le variabili più significative per il modello, migliorandone l'efficienza computazionale e la qualità complessiva. Ridurre il numero di variabili consente di limitare il rumore nei dati e di evitare un'eccessiva complessità del modello, contribuendo così a ottenere risultati più affidabili.

Il primo metodo proposto per selezionare le feature più significative è l'**analisi della correlazione tra le variabili** [34], che consente di misurare il grado di relazione tra coppie di variabili. Per questa analisi sono stati utilizzati tre diversi coefficienti di correlazione, descritti di seguito.

- **Coefficiente di correlazione di Pearson** [35]:

  Il coefficiente di correlazione di Pearson è una misura parametrica della relazione lineare tra due variabili numeriche X, Y ed è indicato con ρ. È definito come il rapporto tra la covarianza tra le due variabili in esame e il prodotto delle loro deviazioni standard individuali. Pearson assume valori tra -1 e 1, con valori estremi che indicano una correlazione perfettamente positiva o negativa e 0 che indica l'assenza di relazione lineare. Il coefficiente di correlazione di Pearson è calcolato matematicamente come segue:

$$\rho_{X,Y} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}, \tag{3.15}$$

  dove X_i e Y_i sono i valori osservati delle due variabili, $\bar{X}$ e $\bar{Y}$ sono le loro medie e n è il numero di osservazioni.

- **Coefficiente di correlazione di Spearman** [35]:

  Quest'altro coefficiente è una misura non parametrica che quantifica la relazione monotona tra due variabili numeriche X e Y. A differenza del coefficiente di Pearson, che opera direttamente sui valori delle variabili, Spearman si basa sui ranghi, ovvero sui valori trasformati in una scala ordinata compresa tra 1 e n. In altre parole, gli

i-esimi valori osservati di X e Y vengono prima convertiti nei rispettivi ranghi r_{X_i} e r_{Y_i}, e il coefficiente viene quindi calcolato sugli indici di posizione anziché sui dati originali. Matematicamente il coefficiente di Spearman ρ_s è definito come:

$$\rho_s(X,Y) = 1 - \frac{6\sum d_i^2}{n(n^2-1)}\,, \tag{3.16}$$

dove $d_i = r_{X_i} - r_{Y_i}$ rappresenta la differenza tra i ranghi delle osservazioni corrispondenti e n è il numero totale di osservazioni.

- **Coefficiente di correlazione di Kendall** [35]:

  Il coefficiente (o tau) di Kendall è una misura non parametrica della relazione tra due variabili, basata sul confronto tra le probabilità di concordanza e discordanza delle coppie di osservazioni X_i, Y_i. A differenza del coefficiente di Pearson, che cattura solo relazioni lineari, e di Spearman, che si basa sui ranghi, Kendall valuta direttamente la coerenza dell'ordine relativo tra le osservazioni. I valori del coefficiente di Kendall sono compresi tra -1 e $+1$. Si calcola come:

$$\rho_\tau(X,Y) = \frac{C-D}{\frac{1}{2}n(n-1)}\,, \tag{3.17}$$

  dove C è il numero di coppie concordanti, D il numero di coppie discordanti e n il numero totale di osservazioni.

L'uso combinato di questi tre coefficienti consente un'analisi più completa della dipendenza tra le variabili, identificando sia relazioni lineari sia monotone non lineari.

La correlazione tra le variabili è rappresentata attraverso una matrice, detta appunto **matrice di correlazione**, in cui ogni cella indica il coefficiente di correlazione tra le due variabili corrispondenti. Per rendere più immediata l'interpretazione dei dati, è stata impiegata una heatmap che utilizza una scala cromatica per evidenziare i livelli di correlazione. Successivamente è stata applicata una strategia di feature selection basata sulla correlazione per identificare e rimuovere le variabili ridondanti. A tal fine sono state utilizzate tre soglie di correlazione: alta (0.95, 0.90, 0.85), media (0.90, 0.85, 0.75) e bassa (0.85, 0.80, 0.65) in base alla tipologia di correlazione (Pearson, Spearman e Kendall). Sono state considerate come ridondanti le coppie di variabili con correlazione superiore a queste soglie di riferimento.

Per i dataset con un numero relativamente basso di colonne, le variabili ridondanti sono state rimosse iterativamente, eliminando una colonna per volta e aggiornando la matrice di correlazione a ogni passaggio. Per i dataset con molte colonne, invece, la matrice delle correlazioni è stata calcolata una sola volta all'inizio, rimuovendo tutte le variabili ridondanti in contemporanea. Il modo corretto sarebbe solamente il primo metodo; il secondo, infatti, è stato denominato nella tesi come *greedy*.

I risultati sperimentali mostrano che, in generale, questo metodo basato sulle matrici di correlazione è in grado di catturare efficacemente sia le relazioni lineari che quelle non

lineari, garantendo una selezione sfumata e preservando l'interpretabilità delle variabili. Tuttavia rimane lo svantaggio che, per dataset molto grandi (ad esempio, più grandi del dataset WADI [36]), le matrici di correlazione sono molto costose da costruire, pur utilizzando l'approccio *greedy*. In Figura 3.5 sono mostrate le matrici di correlazione relative ai tre diversi coefficienti per il dataset SKAB [25]. In questo esempio si può osservare che solo la soglia bassa (0.85, 0.80, 0.65) rimuoverebbe delle variabili.

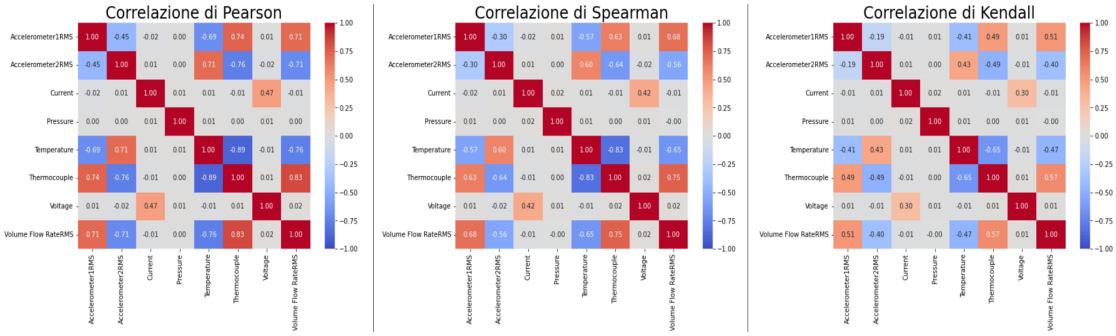

Figura 3.5: Matrici di correlazione per le variabili del dataset SKAB.

Un altro approccio molto utilizzato per rimuovere le variabili ridondanti è l'**analisi delle componenti principali (PCA)**, che viene approfondito nella Sezione 4.1.2. Tuttavia è stato escluso dalla pipeline finale per preservare l'interpretabilità del dataset, un aspetto essenziale nelle applicazioni industriali come quella trattata in questo studio. La PCA, infatti, sostituisce le variabili originali con combinazioni lineari chiamate componenti principali, rendendo più difficile comprendere il significato delle nuove variabili e la loro relazione con i fenomeni industriali analizzati.

Capitolo 4

Metodi

Il capitolo in esame analizza i metodi specifici adottati per affrontare l'anomaly detection in serie temporali multivariate, in cui il dataset di addestramento è composto esclusivamente da dati non anomali, come discusso nel Capitolo introduttivo 1. Una panoramica della classificazione di questi metodi è riportata in Figura 2.1 e, più in generale, nella Sezione 2.1. I metodi considerati, ad eccezione dei metodi di baseline (Sezione 4.1), sono illustrati nella seguente Tabella 4.1.

Metodo	**Autoencoder** 4.3	**Recurrent neural network** 4.4	**LSTM-autoencoder** 4.7
Approccio	Approccio di ricostruzione	Approccio predittivo	Approccio di ricostruzione
Modelli testati	• Autoencoder • Denoising autoencoder • Sparse autoencoder • Convolutional autoencoder	• Long short-term memory • Gated recurrent unit	• Configurazione con strati LSTM e uno strato lineare finale.

Tabella 4.1: Panoramica dei metodi di anomaly detection considerati.

4.1 Metodi per la baseline

In questa tesi sono stati selezionati due **metodi statistici** come baseline (ossia come metodi di riferimento) per valutare le prestazioni in termini di F1 score. Entrambi sono metodi consolidati nell'analisi delle serie temporali e nel machine learning. Il primo, VARIMA, è tra i modelli più utilizzati per la previsione di serie temporali multivariate, mentre il secondo, PCA, è uno dei metodi più diffusi per la ricostruzione non supervisionata. Entrambi si basano su combinazioni lineari e non fanno uso di reti neurali. Questa scelta permette di mettere a confronto l'efficacia degli approcci tradizionali con quella dei modelli avanzati basati su reti neurali. Un'analisi dettagliata delle prestazioni di ciascun metodo è riportata nel Capitolo dei risultati 6.

4.1.1 ARIMA ed Estensioni

Il modello ARIMA [37], [22], acronimo di AutoRegressive Integrated Moving Average, è uno dei modelli più utilizzati per l'analisi e la previsione delle serie temporali univariate, e può essere applicato anche all'anomaly detection.

Il modello ARIMA è stato sviluppato come evoluzione dei modelli autoregressivi (AR) e dei modelli di media mobile (MA), che insieme formano il modello ARMA. L'ARIMA combina le caratteristiche di AR e MA e aggiunge anche la capacità di gestire la non stazionarietà attraverso la differenziazione. Una serie temporale è stazionaria (debolmente) se le sue proprietà statistiche (media, varianza, autocovarianza) non cambiano nel tempo, come visto in 2.3.3. Quando una serie è non stazionaria, risulta più difficile per un modello identificare una relazione stabile tra i dati passati e futuri.

Il modello ARIMA si basa sulla combinazione di **tre componenti**: la componente autoregressiva (AR), la componente di integrazione (I) e la componente di media mobile (MA).

1. AR (AutoRegressione)

La componente autoregressiva del modello ARIMA utilizza i valori passati della serie temporale per prevedere i valori futuri. In altre parole, si cerca di scrivere lo stato della serie storica in t come una regressione lineare multipla delle osservazioni passate. Il termine autoregressione fa riferimento al fatto che si tratta di una regressione di ogni variabile contro se stessa. La formula generale per un modello AR di ordine p è:

$$x_t = c + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + \epsilon_t \,. \tag{4.1}$$

Dove x_t è il valore attuale della serie temporale, c è la componente di tendenza (trend), ϕ_i sono i coefficienti (o pesi) autoregressivi per i lag i (con $i = 1, 2, \ldots, p$) e ϵ_t è il termine di errore (residuo) al tempo t .

Un lag (ritardo) rappresenta il numero di periodi di tempo che separano due osservazioni di una serie temporale.

2. MA (Media Mobile)

La componente di media mobile utilizza i residui del passato per spiegare l'osservazione all'istante t. La formula generale per un modello MA di ordine q è:

$$x_t = c + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q} + \epsilon_t \,. \tag{4.2}$$

Dove c è la componente di trend, θ_i sono i coefficienti (pesi) della media mobile per i lag i (con $i = 1, 2, \ldots, q$) e ϵ_{t-i} è il residuo al tempo $t - i$.

Non si tratta di una regressione ordinaria perché gli ϵ non sono l'osservazione di un processo vero, ma la differenza tra l'insieme dei dati e il modo di spiegarlo del modello. Grazie alla proprietà di invertibilità, sotto particolari vincoli, ogni modello MA di ordine q può

essere scritto come un modello AR di ordine ∞ e ogni modello AR di ordine p può essere scritto come un modello MA di ordine ∞ [22].

3. I (Integrazione)

La componente di integrazione si occupa di rendere la serie temporale stazionaria attraverso la differenziazione. Se una serie temporale non è stazionaria, si può applicare la differenziazione per rimuovere le tendenze. La differenziazione di ordine d è definita come:

$$\Delta^d x_t = x_t - x_{t-d}\,, \tag{4.3}$$

dove Δ^d rappresenta l'operazione di differenziazione applicata d volte. Ad esempio, per $d = 1$ (rimozione trend lineare):

$$\Delta x_t = x_t - x_{t-1}\,.$$

Combinando queste tre componenti il **modello ARIMA** può essere espresso come:

$$x_t = c + \phi_1 \cdot \Delta^d x_{t-1} + \ldots + \phi_p \cdot \Delta^d x_{t-p} + \theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q} + \epsilon_t\,. \tag{4.4}$$

In notazione ARIMA, il modello è denotato come ARIMA(p, d, q), dove:

- p è l'ordine della parte autoregressiva, cioè il numero di osservazioni passate usate per prevedere il valore attuale;
- d è l'ordine della differenziazione;
- q è l'ordine della parte media mobile, ossia il numero di termini dei residui passati inclusi nel modello per correggere la previsione attuale.

La scelta di questi tre parametri è di fondamentale importanza e può essere effettuata in modo automatico, quindi senza l'utilizzo di funzioni di autocorrelazione, tramite la funzione *autoarima* della libreria Pmdarima [38].

Nel modello ARIMA e nelle sue varianti, i vari coefficienti o pesi vengono stimati mediante il metodo della massima verosimiglianza [39]. Inoltre, si fa l'ipotesi che i residui ϵ_t siano variabili casuali distribuite secondo una normale con media zero e varianza costante (rumore bianco), ovvero $\epsilon_t \sim N(0, \sigma^2)$. Questo implica che i residui non devono presentare autocorrelazione, devono essere indipendenti nel tempo e mantenere una varianza costante (omoschedasticità). Se queste condizioni non sono rispettate, il modello potrebbe generare stime distorte e risultare inefficace.

L'ARIMA è utilizzato per l'anomaly detection nelle serie temporali grazie alla sua capacità di modellare e prevedere l'andamento dei dati. Una volta addestrato, il modello genera previsioni basate sui valori storici, e le anomalie vengono individuate analizzando i residui, ossia la differenza tra i valori osservati e quelli previsti. Se questi superano una soglia predefinita (discussione approfondita nella Sezione 5.3), si può ritenere che l'osservazione in questione sia anomala [40].

Nel corso della tesi, sono stati analizzati diversi dataset (maggiori dettagli nella Sezione 5.1), tutti costituiti da serie temporali multivariate (maggiori dettagli in 2.3.2). In questi casi si ricorre all'utilizzo di un modello ARIMA separato per ogni singola variabile del dataset. Tuttavia, tale approccio prevede la modellazione di ogni serie temporale separatamente, senza considerare le possibili interazioni tra le variabili. Tale scelta risulta pertanto adeguata nel caso in cui le variabili risultino quasi indipendenti tra loro, e quindi modellarle separatamente potrebbe essere sufficiente. Nei dataset presi in considerazione nella tesi, e nella maggior parte dei dataset industriali, non è possibile avanzare questa forte ipotesi; pertanto si ricorre alle varianti vettoriali del modello ARIMA.

VARIMA

VARIMA (Vector Autoregressive Integrated Moving Average) [37] rappresenta l'estensione multivariata di ARIMA, permettendo di analizzare simultaneamente più serie temporali. Pertanto, oltre alle componenti autoregressive e di media mobile di ciascuna variabile, vengono considerate anche le interazioni tra le diverse variabili. In questo modo ogni variabile dipende non solo dai propri valori passati, ma anche da quelli delle altre variabili.

La formulazione generale del modello VARIMA combina le componenti autoregressive, di media mobile e di integrazione in forma vettoriale. Può essere vista come un sistema di equazioni che descrivono come le variabili si influenzano reciprocamente nel tempo. Ecco una panoramica delle formule matematiche che definiscono il modello.

Il modello VARIMA(p, d, q) per una serie temporale multivariata viene espresso come:

$$X_t = \mathbf{c} + \mathbf{A_1} \cdot \Delta^d X_{t-1} + \mathbf{A_2} \cdot \Delta^d X_{t-2} + \ldots + \mathbf{A_p} \cdot \Delta^d X_{t-p} + \mathbf{B_1} \epsilon_{t-1} + \mathbf{B_2} \epsilon_{t-2} + \ldots + \mathbf{B_q} \epsilon_{t-q} + \epsilon_t \,.$$

Dove:

- $\mathbf{c}$ è il vettore dei trend.
- X_t è il vettore delle variabili osservate al tempo t.
- $\mathbf{A}_i$ sono le matrici dei coefficienti (pesi) autoregressivi per i lag i (con $i = 1, 2, \ldots, p$).
- $\mathbf{B}_i$ sono le matrici dei coefficienti (pesi) della media mobile per i lag i (con $i = 1, 2, \ldots, q$).
- Δ^d rappresenta l'operazione di differenziazione (vettoriale) applicata d volte
- ϵ_t è il vettore dei residui al tempo t.

Componenti del Modello

1. **Forma Vettoriale (V)**: ogni entrata o riga del vettore X_t corrisponde ad una variabile del dataset.

2. **Autoregressione (AR)**: la parte autoregressiva del modello è costituita dai termini $\mathbf{A}_i X_{t-i}$. Ogni variabile nel vettore X_t è influenzata dai propri valori passati (stessa riga) e da quelli delle altre variabili (altre righe).

3. **Media Mobile (MA)**: la parte di media mobile è rappresentata dai termini $\mathbf{B}_i \epsilon_{t-i}$. Questi termini catturano l'influenza degli errori passati sulle variabili attuali.

4. **Integrazione (I)**: la parte di integrazione è gestita attraverso la differenziazione delle serie temporali, che viene espressa come $\Delta^d X_t = X_t - X_{t-1}$.

Nella tesi il modello VARIMA è stato pertanto scelto come principale approccio statistico predittivo per l'analisi di serie temporali multivariate applicate all'anomaly detection. Tuttavia, come evidenziato nel Capitolo dei risultati 6, tale modello si è rivelato inefficiente dal punto di vista computazionale: su dataset contenenti più di 25 variabili e 100.000 osservazioni, l'addestramento del modello non è stato completato in meno di 12 ore.

A titolo esemplificativo, la Figura 4.1 mostra un confronto tra le predizioni (in arancione) dei modelli ARIMA (sinistra) e VARIMA (destra) per la variabile *Accelerometer1RMS* all'interno del dataset SKAB [25]. È possibile osservare come il modello VARIMA fornisca una rappresentazione meno approssimativa della variabile rispetto ad ARIMA nello stesso intervallo temporale. Questo comportamento può essere attribuito principalmente alla maggiore complessità del modello VARIMA, sebbene ciò comporti una maggiore complessità computazionale.

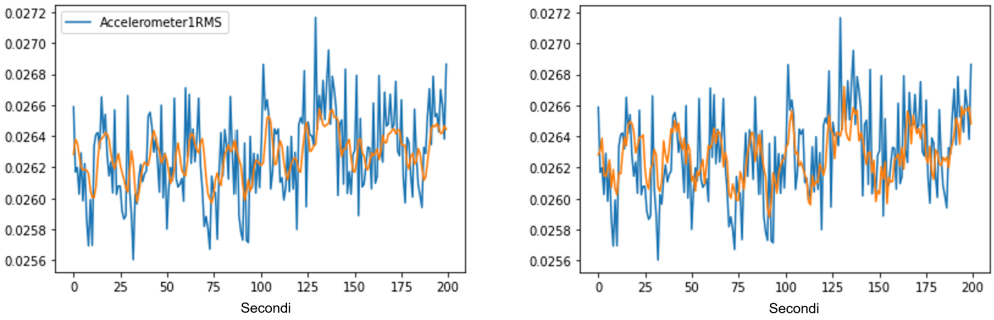

Figura 4.1: Predizioni (in arancione) dei modelli ARIMA (sinistra) e VARIMA (destra) per la variabile *Accelerometer1RMS* del dataset SKAB.

4.1.2 Analisi delle Componenti Principali (PCA)

L'Analisi delle componenti principali (PCA) [41], [42], [43] è una tecnica non supervisionata ampiamente utilizzata per la riduzione della dimensionalità e la ricostruzione. Il suo obiettivo è trasformare un insieme di variabili correlate in un numero inferiore di variabili non correlate, chiamate componenti principali. La PCA può essere impiegata sia come tecnica di feature selection (come discusso nella Sezione 3.6), selezionando le

componenti principali più rilevanti, sia come metodo di ricostruzione nell'ambito dell'anomaly detection, confrontando i dati veri con quelli ricostruiti per identificare eventuali anomalie.

La PCA ha come obiettivo l'individuazione delle direzioni in cui i dati presentano la massima varianza per la riduzione della dimensionalità (numero di colonne N) del dataset N-dimensionale proiettandolo in un sottospazio M-dimensionale (con $M < N$). Le proiezioni ortogonali lineari sono note come componenti principali (PC). La direzione di una buona proiezione è parallela alla direzione di massima varianza, la quale è l'autovettore corrispondente all'autovalore più grande della matrice di varianza e covarianza campionaria Σ.

In altre parole la PCA si basa su un **procedimento matematico** rigoroso che trasforma il dataset $X \in \mathbb{R}^{T \times N}$, composto da T osservazioni e N variabili, in uno spazio ridotto di M componenti principali. I passaggi chiave sono i seguenti:

- **Standardizzazione dei dati**:

  Prima di applicare la PCA i dati devono essere standardizzati (vedi la Sottosezione 3.5.2 per maggiori dettagli), poiché variabili con scale diverse potrebbero influenzare erroneamente il calcolo delle componenti principali. Ogni osservazione t della variabile n viene trasformata nel seguente modo:

$$X'_{t,n} = \frac{X_{t,n} - \mu_n}{\sigma_n}, \quad \text{con} \quad \begin{cases} \mu_n = \frac{1}{T} \sum_{t=1}^{T} X_{t,n} \\ \sigma_n = \sqrt{\frac{1}{T-1} \sum_{t=1}^{T} (X_{t,n} - \mu_n)^2} \end{cases}, \tag{4.5}$$

  dove μ_n è la media campionaria della n-esima variabile, σ_n è la deviazione standard campionaria della n-esima variabile e X' è il dataset standardizzato (maggiori dettagli in Rice et al. [44]).

- **Calcolo della matrice di covarianza**:

  La PCA si basa sull'analisi della matrice di covarianza Σ dello spazio dei dati, che rappresenta le relazioni tra le variabili X_n, $n = 1, \ldots, N$. Matematicamente si esprime come:

$$\Sigma = \frac{1}{T} (X')^\top X', \quad \text{con} \quad \Sigma \in \mathbb{R}^{N \times N}, \tag{4.6}$$

  dove si assume che Σ sia una matrice simmetrica e definita positiva.

- **Calcolo autovalori e autovettori**:

  Gli autovalori λ e gli autovettori w della matrice di covarianza Σ sono elementi fondamentali per determinare le componenti principali. Infatti, risolvendo l'equazione agli autovalori:

$$\Sigma \mathrm{w} = \lambda \mathrm{w}, \tag{4.7}$$

si ottengono gli autovettori w che identificano le **componenti principali**, ovvero le direzioni nello spazio dei dati lungo cui la varianza è massima; e gli autovalori λ che indicano la varianza spiegata da ciascuna componente principale (calcolo completo in James et al. [42]).

Gli autovettori vengono ordinati in base ai corrispondenti autovalori in ordine decrescente ($\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$).

- **Proiezione dei dati**:

  Il dataset X' viene proiettato nello spazio ridotto delle componenti principali utilizzando i primi M autovettori (quelli con i maggiori autovalori). La proiezione è data da:

  $$Z = X' W_M \,, \tag{4.8}$$

  dove $W_M \in \mathbb{R}^{N \times M}$ è la matrice contenente i primi M autovettori come colonne; e $Z \in \mathbb{R}^{T \times M}$ è il dataset proiettato nello spazio ridotto, in cui ogni riga rappresenta un'osservazione nello spazio delle componenti principali.

- **Ricostruzione dei dati**:

  Se si utilizza la PCA per la ricostruzione, il dataset X può essere approssimato proiettando Z nello spazio dei dati di partenza:

  $$R = Z W_M^\top \,, \tag{4.9}$$

  dove $R \in \mathbb{R}^{T \times N}$ è il dataset ricostruito.

  La differenza tra il dataset X' e quello ricostruito R può essere utilizzata per calcolare l'errore di ricostruzione:

  $$\mathcal{L} = \|X' - R\| \,. \tag{4.10}$$

- **Varianza spiegata**:

  La qualità della riduzione della dimensionalità viene valutata attraverso la varianza spiegata, che misura quanta parte della varianza totale è catturata da ciascuna componente principale. La varianza spiegata (in percentuale) per la m-esima componente principale è calcolata come:

  $$V_m = \frac{\lambda_m}{\sum_{i=1}^{N} \lambda_i} \,. \tag{4.11}$$

Nell'applicazione pratica della PCA, un passaggio cruciale è la determinazione del **numero ottimale di componenti principali da conservare**. Questo valore può essere scelto in base alla percentuale di varianza spiegata dalle componenti V_M. L'obiettivo è individuare il numero minimo di componenti che permetta di catturare una percentuale significativa

della varianza totale. In questa tesi è stato adottato come criterio il mantenimento delle componenti che spiegano almeno il 95% della varianza. Sebbene questo approccio sia ampiamente utilizzato, non è l'unico possibile; altre strategie potrebbero essere considerate in base alle specifiche esigenze dell'analisi (Murphy et al. [45] per ulteriori dettagli). Un metodo grafico utile per tale scelta è rappresentato da uno scree plot cumulativo come in Figura 4.2, che mostra la somma progressiva della varianza spiegata dalle componenti principali.

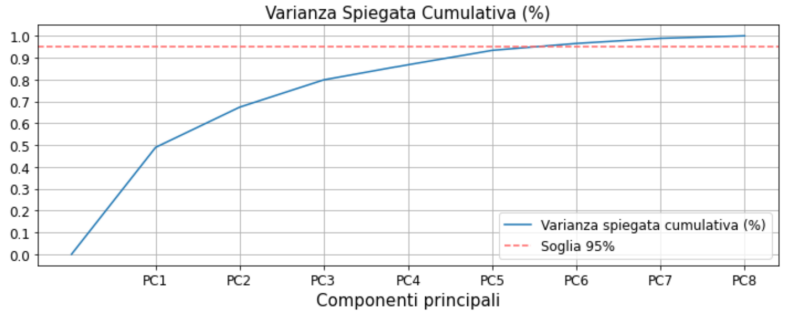

Figura 4.2: Scree plot della PCA sul dataset SKAB.

Per interpretare quali variabili del dataset di partenza X contribuiscono maggiormente a ciascuna componente principale si possono utilizzare dei barplot. Questi grafici mostrano i coefficienti di ciascuna variabile all'interno delle componenti principali, evidenziando il peso di ogni variabile nelle diverse componenti principali. In Figura 4.4 sono mostrati i barplot relativi alle 6 componenti principali che spiegano il 95% della varianza.

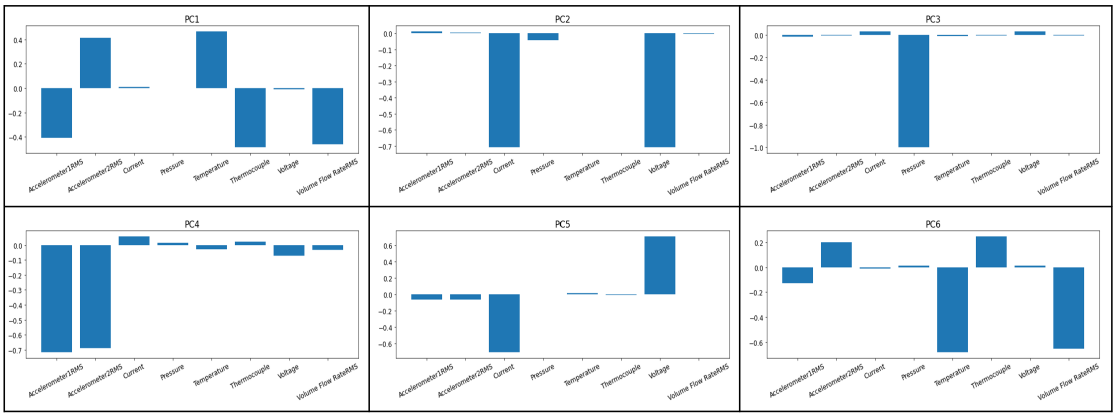

Figura 4.3: Barplots delle componenti principali della PCA sul dataset SKAB.

In questa tesi, la PCA è stata adottata come principale metodo non supervisionato per la ricostruzione nelle serie temporali multivariate applicate all'anomaly detection. Tuttavia, come evidenziato nel Capitolo dei risultati 6, le sue prestazioni si sono rivelate inferiori alla maggior parte degli altri modelli. Il motivo principale è la difficoltà del modello nel catturare relazioni altamente non lineari e complesse nei dati.

4.2 Reti Neurali

Le reti neurali rappresentano una delle innovazioni più significative nel campo dell'intelligenza artificiale e del machine learning. Questi modelli, ispirati al funzionamento del cervello umano, sono progettati per riconoscere schemi e apprendere da grandi quantità di dati. Le reti neurali sono state sviluppate sin dagli anni '40, ma hanno guadagnato popolarità solo recentemente, grazie ai progressi nella potenza computazionale e alla disponibilità di grandi dataset [15].

Una rete neurale è composta da strati di nodi (layer), o *neuroni*, che elaborano le informazioni. Ogni neurone riceve l'input $x \in \mathbb{R}^{T \times N}$, applica una funzione di attivazione ϕ e produce un output $y \in \mathbb{R}^{T \times N}$ che viene passato ai neuroni dello strato successivo. Matematicamente, nel caso più semplice di strato lineare, l'output y di un neurone può essere espresso come:

$$y = \phi \left(\sum_{n=1}^{N} w_n x_n + b \right) , \tag{4.12}$$

dove:

- x_n sono gli input della variabile n-esima del neurone;
- w_n sono i pesi associati a ciascun input;
- b è il bias del neurone;
- ϕ è la funzione di attivazione, che introduce non linearità.

Le **funzioni di attivazione** più comunemente utilizzate includono (in Figura 4.4 sono mostrati i rispettivi grafici):

- **Sigmoide**: $\sigma(x) = \frac{1}{1+e^{-x}}$. Comprime l'output in un intervallo compreso tra 0 e 1, risultando particolarmente adatta ai problemi di classificazione binaria;
- **ReLU** (Rectified Linear Unit): $\text{ReLU}(x) = \max(0, x)$. Porta molti neuroni a rimanere inattivi, ossia con un output pari a zero, e accelera il processo di addestramento nelle reti neurali profonde;
- **Tanh** (Tangente iperbolica): $\tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$. Funziona in modo simile alla sigmoide, ma restituisce valori compresi tra -1 e 1, contribuendo a centrare i dati intorno allo zero.

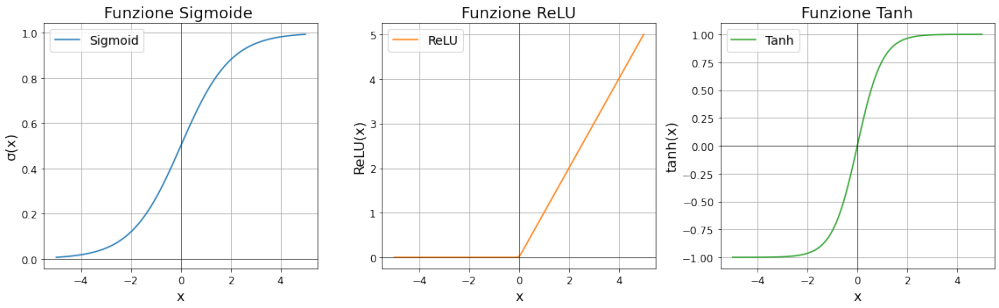

Figura 4.4: Grafici delle funzioni di attivazione più utilizzate.

Il processo di apprendimento si basa sull'ottimizzazione dei pesi delle connessioni tra i neuroni, con l'obiettivo di ridurre la discrepanza tra le previsioni del modello e i valori reali. Questo avviene tramite algoritmi come la retropropagazione (backpropagation), che aggiornano iterativamente i pesi minimizzando l'errore. Quest'ultimo, indicato come $\mathcal{L}$, può essere rappresentato da una **funzione di perdita**, come l'errore quadratico medio (MSE), definito da:

$$\mathcal{L} = \frac{1}{T} \sum_{t=1}^{T} \|x_t - \hat{x}_t\|^2 \,, \tag{4.13}$$

dove T è il numero totale di osservazioni nel dataset, x_t è il valore reale, $\hat{x}_t$ è il valore previsto dal modello.

L'ottimizzazione dei pesi avviene attraverso la discesa del gradiente, dove i pesi vengono aggiornati secondo la regola:

$$w_n \leftarrow w_n - \alpha \frac{\partial Err}{\partial w_n} \,, \quad n = 1, \ldots, N \quad , \tag{4.14}$$

dove α è il tasso di apprendimento (learning rate). Questo processo consente alla rete di apprendere dai dati e migliorare le sue previsioni nel tempo [15]. Questa è una trattazione piuttosto sintetica del processo di apprendimento di una rete neurale; ulteriori dettagli sono presenti in [15], [45], [46].

Le reti neurali possono essere classificate in diverse categorie, tra cui: reti neurali profonde, reti convoluzionali e reti ricorrenti. Ognuna di queste architetture è adatta a compiti specifici, come il riconoscimento delle immagini o l'elaborazione del linguaggio naturale [15]. In particolare, nel contesto dell'anomaly detection su serie temporali multivariate, le reti ricorrenti, e in particolare le LSTM (Long Short-Term Memory), si rivelano particolarmente efficaci nel gestire la complessità e la variabilità dei dati temporali, come attestato dai risultati nel Capitolo 6 e anche nei lavori di Belay et al. [7] e di Pota et al. [2]. Un'altra architettura efficace e ampiamente utilizzata per l'anomaly detection è quella degli autoencoder. Si fa riferimento alla Sezione 2.1 per una spiegazione più approfondita della classificazione degli approcci di deep learning applicati all'anomaly detection.

4.3 Autoencoder

Gli autoencoder [15], [47], [2] sono reti neurali progettate per ricostruire i dati in input in modo non supervisionato, attraverso un processo di codifica e decodifica che consente di impararne una rappresentazione compatta. La loro architettura pertanto si compone di due elementi principali: l'encoder, che trasforma l'input in uno spazio latente a dimensione ridotta, e il decoder, che tenta di ricostruire l'input originale a partire da questa rappresentazione. Gli autoencoder possono essere considerati un'estensione non lineare della PCA [15].

Più precisamente l'**encoder**, rappresentato dalla funzione f, è costituito da una serie di strati con pesi e funzioni di attivazione. Il suo compito è trasformare l'input $x \in \mathbb{R}^{T\times N}$ in una rappresentazione interna o latente (embedding), indicata con $h \in \mathbb{R}^{T\times M}$, $M < N$, che solitamente ha una dimensionalità inferiore rispetto a quella dell'input originale. Questo processo può essere descritto matematicamente come:

$$h = f(x)\,. \tag{4.15}$$

In un autoencoder *tradizionale*, l'encoder è costituito da una serie di strati completamente connessi (fully connected), con alla fine una funzione di attivazione non lineare (maggiori dettagli nella Sezione precedente 4.2):

$$h = \phi(W_e x + b_e)\,, \tag{4.16}$$

dove $W_e \in \mathbb{R}^{M\times N}$ è la matrice dei pesi dell'encoder, $b_e \in \mathbb{R}^M$ è il vettore di bias e ϕ è la funzione di attivazione.

Successivamente, il **decoder**, rappresentato dalla funzione g, è anch'esso costituito da una serie di strati con pesi e funzioni di attivazione. Il suo compito è utilizzare la rappresentazione latente h per generare un output $r \in \mathbb{R}^{T\times N}$, denominato ricostruzione, che dovrebbe riprodurre fedelmente l'input originale x. Questo processo di decodifica può essere espresso matematicamente come:

$$r = g(h)\,. \tag{4.17}$$

Come per l'encoder, in un autoencoder tradizionale, il decoder è costituito da una serie di strati completamente connessi, con alla fine una funzione di attivazione non lineare:

$$r = \phi(W_d h + b_d)\,, \tag{4.18}$$

dove $W_d \in \mathbb{R}^{N\times M}$ è la matrice dei pesi del decoder e $b_d \in \mathbb{R}^N$ è il vettore di bias.

Insieme, encoder e decoder formano il **processo di codifica e decodifica** dell'autoencoder, che può essere rappresentato matematicamente come:

$$g(f(x)) = g(h) = r\,. \tag{4.19}$$

L'obiettivo è minimizzare la differenza tra l'input originale x e la ricostruzione r. A tal scopo l'autoencoder viene addestrato ottimizzando una funzione di perdita, come l'errore quadratico medio (MSE):

$$\mathcal{L} = \frac{1}{T} \sum_{t=1}^{T} \|x_t - r_t\|^2. \tag{4.20}$$

dove x_t è l'input della t-esima osservazione, r_t è la ricostruzione della t-esima osservazione e T è il numero di osservazioni dell'input x.
Minimizzando la funzione di perdita $\mathcal{L}$ il modello impara a comprimere e ricostruire i dati in modo efficace. La struttura generale di un autoencoder è mostrata in Figura 4.5.

Figura 4.5: Struttura generale autoencoder.

La Figura 4.7 mostra un esempio schematico dell'architettura di un autoencoder tradizionale, evidenziando la disposizione degli strati dell'encoder, della rappresentazione latente e del decoder, insieme alle connessioni completamente connesse che permettono il flusso dei dati dal livello di input alla ricostruzione finale.

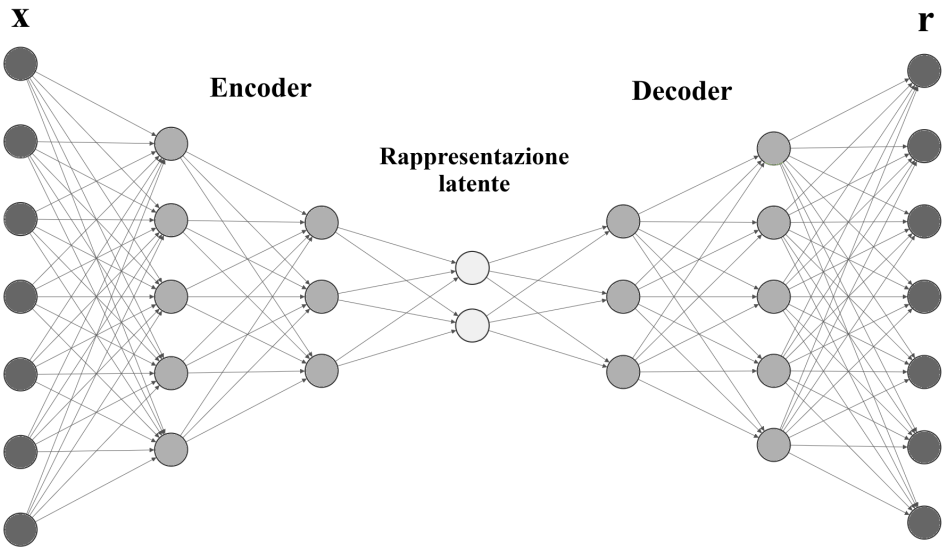

Figura 4.6: Architettura autoencoder tradizionale (con strati completamente connessi).

In un contesto di anomaly detection l'autoencoder viene addestrato a ricostruire i dati che riflettono il comportamento tipico/normale del sistema, ovvero senza anomalie. Di conseguenza, quando il modello viene applicato a dati anomali, fatica a ricostruirli con

precisione, portando a un errore di ricostruzione $\mathcal{L}$ più elevato. Questo errore viene utilizzato per assegnare un punteggio di anomalia. Se l'errore di ricostruzione supera una soglia prestabilita δ, il dato viene classificato come anomalo. La determinazione di questa soglia è un passaggio critico e viene approfondita nella Sezione 4.7. Questo approccio, basato sulla ricostruzione, è alla base di numerosi metodi derivati, come l'LSTM-autoencoder (LSTM-AE) (Vedi Sezione 4.7), che sfruttano lo stesso principio. Un esempio dell'output di un autoencoder applicato al dataset SKAB [25] è riportato in Figura 4.7.

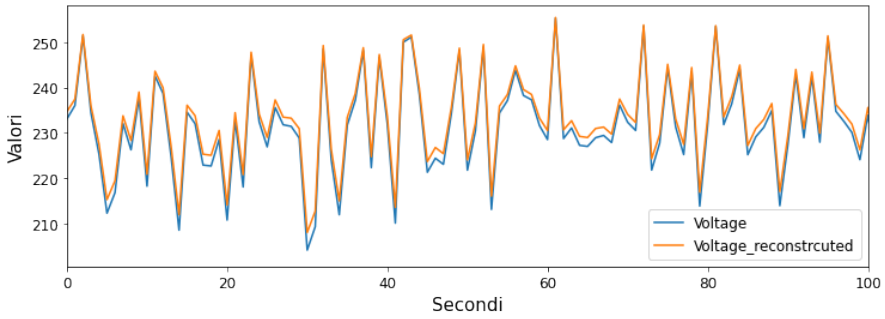

Figura 4.7: L'autoencoder funziona come metodo di ricostruzione.

Inoltre, esiste un'ampia gamma di varianti dell'autoencoder tradizionale che differiscono per alcune piccole modifiche, che non incidono in modo significativo sui risultati (maggiori dettagli nella Tabella 6.2). Alcune di queste varianti sono state incluse nell'analisi e sono discusse di seguito.

4.3.1 Denoising Autoencoder (DAE)

Il denoising autoencoder (DAE) [48] è una variante dell'autoencoder classico progettata per migliorare la robustezza del modello rispetto a input rumorosi o dati corrotti. L'approccio dell'autoencoder tradizionale non è ottimale in presenza di rumore nei dati, poiché il modello potrebbe apprendere anche informazioni di disturbo. Nel denoising autoencoder, invece, il modello non ricostruisce direttamente l'input originale x, ma impara a ricostruirlo a partire da una sua versione corrotta $\tilde{x}$, ottenuta applicando del rumore al dato di input. La relazione di codifica e decodifica 4.19 diventa quindi:

$$g(f(\tilde{x})) = g(h) = r. \tag{4.21}$$

L'obiettivo del modello diventa quindi minimizzare la seguente funzione di perdita:

$$\mathcal{L}_{DAE} = \|x - g(f(\tilde{x}))\|^2. \tag{4.22}$$

Questa modifica obbliga il modello a non dipendere troppo da dettagli specifici dell'input, imparando così una rappresentazione più robusta. Specialmente in presenza di poche osservazioni, il DAE aiuta a evitare che il modello memorizzi il dataset invece di apprenderne le caratteristiche generali.

Nel processo di corruzione dell'input x, sono stati presi in considerazione i seguenti **tipi di rumore**, comunemente utilizzati nell'addestramento dei denoising autoencoder:

- **Rumore gaussiano**: viene aggiunto un rumore casuale con distribuzione normale $\mathcal{N}(0, \sigma^2)$ all'input:

$$\tilde{x} = x + \eta, \quad \eta \sim \mathcal{N}(0, \sigma^2)\,,$$

  dove σ è il parametro che determina la varianza della distribuzione e controlla l'intensità del rumore.

- **Rumore salt and pepper**: un numero casuale di pixel viene impostato arbitrariamente a 0 (salt) o a 1 (pepper), simulando perdita di informazioni o distorsioni causate da errori di trasmissione.

$$\tilde{x}_{t,n} = \begin{cases} 0, & \text{con probabilità } p/2 \\ 1, & \text{con probabilità } p/2 \\ x_{t,n}, & \text{altrimenti} \end{cases},$$

  dove p è la probabilità di corruzione.

- **Rumore di Poisson**: viene aggiunto un rumore casuale con distribuzione di Poisson Poisson$(\lambda \cdot x)$ all'input:

$$\tilde{x} = x + \eta, \quad \eta \sim \text{Poisson}(\lambda \cdot x)\,,$$

  dove λ è il parametro che determina la varianza della distribuzione e controlla l'intensità del rumore.

4.3.2 Sparse Autoencoder (SAE)

Lo sparse autoencoder (SAE) [49] è una variante dell'autoencoder tradizionale, in cui il numero di neuroni attivati simultaneamente viene limitato mediante l'aggiunta di un termine alla funzione di perdita da minimizzare. Nell'autoencoder tradizionale, come visto in precedenza, il modello è addestrato a ricostruire l'input originale x minimizzando la funzione di perdita:

$$\mathcal{L} = \|x - g(f(x))\|^2\,.$$

Tuttavia, senza ulteriori vincoli, il modello potrebbe apprendere una rappresentazione densa in cui tutti i neuroni della rappresentazione latente h risultano attivati contemporaneamente. Tale scenario comporterebbe una riduzione della capacità del modello di estrarre caratteristiche generali dai dati.

Lo sparse autoencoder risolve questo problema imponendo una penalizzazione sulle attivazioni, favorendo una rappresentazione in cui solo un sottoinsieme dei neuroni nel livello

latente è attivato. Per imporre questa sparsità, viene introdotto alla funzione di perdita $\mathcal{L}$ un termine di penalizzazione basato sulla divergenza di Kullback-Leibler (KL) [41] tra la distribuzione effettiva delle attivazioni e una distribuzione desiderata. Più precisamente, per garantire che la media delle attivazioni reali $\hat{\rho}_j$ si avvicini al valore desiderato della media delle attivazioni ρ (solitamente un valore piccolo, come $\rho = 0.05$), viene utilizzata la divergenza di Kullback-Leibler (KL):

$$D_{KL}(\rho \| \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j} \quad . \tag{4.23}$$

Il termine di penalizzazione totale per la sparsità è dato dalla somma delle divergenze di Kullback-Leibler su tutti i neuroni della rappresentazione latente:

$$\mathcal{L}_{sparse} = \beta \cdot \sum_{j=1}^{n_h} D_{KL}(\rho \| \hat{\rho}_j) \quad , \tag{4.24}$$

dove n_h è il numero di neuroni nel livello latente e β è il peso della penalizzazione (valori alti impongono una maggiore sparsità).

Quindi, in conclusione, la funzione di perdita complessiva dello sparse autoencoder può essere espressa come:

$$\mathcal{L}_{SAE} = \mathcal{L} + \mathcal{L}_{sparse} = \|x - g(f(x))\|^2 + \beta \sum_{j=1}^{n_h} D_{KL}(\rho \| \hat{\rho}_j) \quad . \tag{4.25}$$

4.3.3 Convolutional Autoencoder (CAE)

I convolutional autoencoder (CAE) [50] sono una variante degli autoencoder che utilizza strati convoluzionali per elaborare dati con strutture spaziali o temporali, come immagini e serie temporali. A differenza degli autoencoder tradizionali, che impiegano esclusivamente strati completamente connessi, i CAE sfruttano le convoluzioni per estrarre informazioni locali dai dati, mantenendo le relazioni tra le variabili.

Il semplice autoencoder con strati lineari non tiene conto delle strutture spaziali nei dati. Il motivo è che ogni neurone è connesso a tutti gli altri del livello successivo, e ciò può comportare la perdita di caratteristiche locali nei dati, come la struttura/schema della serie. Inoltre, a causa dell'alto numero di connessioni, si ha un maggior numero di parametri. I convolutional autoencoder risolvono questi problemi adottando strati convoluzionali, che filtrano i dati estraendo solo le caratteristiche più rilevanti.

Nel caso di serie temporali multivariate, il dataset può essere rappresentato come una matrice di dimensioni $T \times N$, con T il numero di istanti temporali e N il numero di variabili. Gli strati convoluzionali vengono utilizzati per catturare pattern sia lungo la dimensione temporale sia tra le variabili.

Uno strato convoluzionale applica un filtro w all'input x, producendo un'output y. Per serie temporali multivariate, l'operazione di **convoluzione** [46], [15] è calcolata come:

$$y_{t,n,k} = \phi\left(\sum_{c=1}^{C}\sum_{\tau=1}^{k_t}\sum_{\nu=1}^{k_n} x_{t+\tau,n+\nu,c} \cdot w_{\tau,\nu,c,k} + b_k\right) . \tag{4.26}$$

Dove:

- $x_{t,n,c}$ è il valore del dato all'istante temporale t, per la variabile n, al canale c;
- $w_{\tau,\nu,c,k}$ è il peso del filtro convoluzionale associato al ritardo temporale τ, alla variabile ν, e al filtro k;
- b_k è il bias associato al filtro k;
- ϕ è la funzione di attivazione;
- y è l'output risultante, con ogni elemento $y_{t,n,k}$ rappresentante un pattern locale estratto.

La struttura di un CAE è composta da un encoder e un decoder convoluzionali. L'encoder convoluzionale f_{conv} utilizza questa operazione per comprimere i dati, riducendo la dimensionalità temporale e il numero di variabili, mentre il decoder convoluzionale g_{conv} cerca di ricostruire il dataset X a partire dalla rappresentazione latente. Il processo intero può essere descritto dalla stessa equazione dell'autoencoder tradizionale 4.19, sostituendo però f e g con le rispettive versioni convoluzionali f_{conv} e g_{conv}.
L'architettura dei convolutional autoencoder è illustrata in Figura 4.8. A differenza degli autoencoder tradizionali, rappresentati in Figura 4.7, i CAE non prevedono connessioni dense tra tutti i neuroni, ma utilizzano al loro posto gli strati convoluzionali per l'elaborazione dei dati.

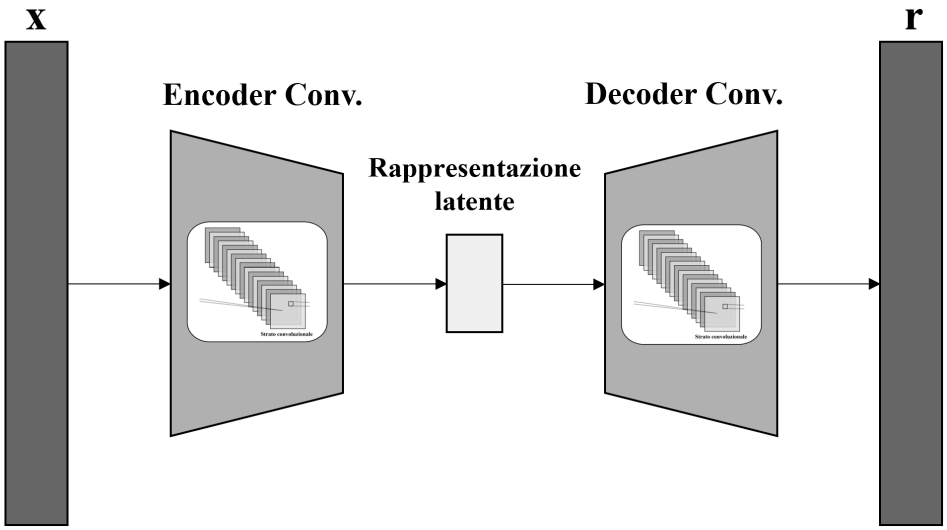

Figura 4.8: Architettura convolutional autoencoder (CAE).

Questa caratteristica conferisce ai CAE importanti vantaggi rispetto agli autoencoder standard, in particolare nella gestione di dati con strutture spaziali o temporali. Inoltre, il numero ridotto di parametri rispetto agli autoencoder completamente connessi aiuta a contenere il rischio di overfitting. Tuttavia, i CAE risultano meno efficaci su dati non strutturati e, in caso di compressione eccessiva, rischiano di perdere dettagli rilevanti nella fase di ricostruzione. Inoltre, sono particolarmente sensibili al processo di ottimizzazione, richiedendo una selezione attenta di filtri e altri parametri per ottenere risultati ottimali. L'analisi dei risultati nel Capitolo 6 mostra le loro prestazioni in confronto con le altre varianti dell'autoencoder.

4.4 Reti Neurali Ricorrenti (RNN)

Le Recurrent Neural Networks (RNN) [15], [51], [45] rappresentano una delle classi fondamentali di architetture nel campo del deep learning, progettate specificamente per l'elaborazione di dati sequenziali. Un esempio di dato sequenziale è una serie storica di valori di temperatura rilevati da una stazione meteorologica, in quanto ogni valore rappresenta la temperatura registrata a intervalli di tempo regolari. Tutte le serie storiche multivariate considerate nella tesi sono dati sequenziali.

Una delle caratteristiche distintive delle RNN è la loro capacità di elaborare sequenze di lunghezza variabile, il che le rende estremamente versatili. Le RNN possono essere impiegate per generare output in tempo reale, come nel caso della generazione di testo o della sintesi vocale. A differenza delle reti neurali tradizionali, che elaborano ciascun input separatamente, le RNN sono in grado di mantenere una **memoria interna**, consentendo di tenere traccia delle informazioni passate durante l'elaborazione di una sequenza. Questo aspetto le rende particolarmente adatte per compiti come l'analisi delle serie temporali, dove il contesto fornito dai dati passati è fondamentale per prevedere i valori futuri [15]. La considerazione di dati passati è una caratteristica comune dei modelli statistici autoregressivi, come gli ARIMA (Sezione 4.1.1). La novità introdotta dalle RNN consiste nella capacità di gestire sequenze più complesse che presentano relazioni non lineari e a lungo termine.

Il termine *recurrent* deriva dalla struttura caratteristica di queste reti, che presentano connessioni cicliche tra i nodi. Questo meccanismo consente di riutilizzare l'output di una cella come input per i calcoli successivi, permettendo alla rete di conservare informazioni nel tempo. Grazie a questa ricorsività, l'output di una cella al tempo t dipende non solo dall'input corrente, ma anche dallo stato nascosto della stessa unità al tempo precedente $t-1$. Questo comportamento è descritto matematicamente dalle seguenti equazioni [8] che caratterizzano le RNN:

1. **Aggiornamento dello stato nascosto**:
   $o_t = \tanh(W_o o_{t-1} + W_x x_t + b)$ $\quad \leftarrow$ ottenuto ricorsivamente (rispetto a t)
2. **Output**: $y_t = \sigma(W_y o_t + b_y)$

Dove:

- o_t è l'output dello stato nascosto al tempo t;
- tanh è la funzione di attivazione tangente iperbolica;
- W_o è la matrice dei pesi che mettono in relazione lo stato nascosto precedente o_{t-1} con quello corrente;
- W_x è la matrice dei pesi per l'input corrente x_t;
- b è il bias;
- y_t è l'output al tempo t;
- σ è la funzione di attivazione sigmoide;
- W_y è la matrice dei pesi per l'output;
- b_y è il bias per l'output.

La Figura 4.9 illustra la struttura generale di una rete neurale ricorrente. Nella parte sinistra della figura la rete è presentata nella sua forma compatta, con le connessioni ricorsive tra le unità. Nella parte destra, è mostrata la versione *unrolled*, in cui la ricorsione viene espansa per rendere più chiaro il passaggio dei dati tra i vari stati.

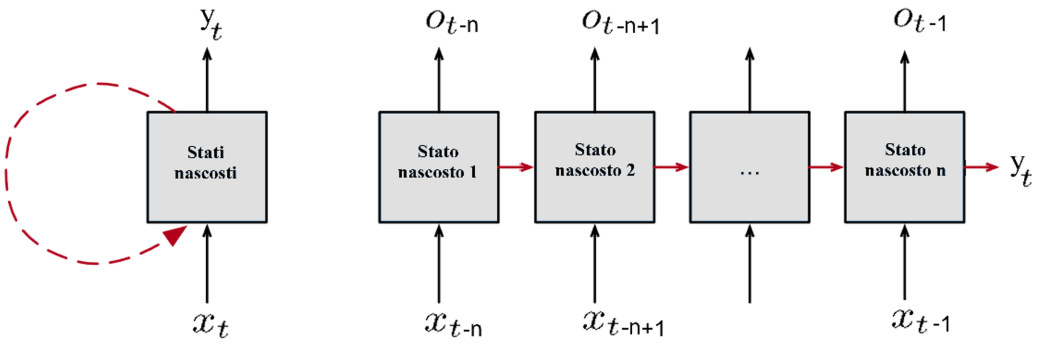

Figura 4.9: Struttura generale di una RNN.

Ogni unità di stato nascosto presenta la struttura interna illustrata in Figura 4.10.

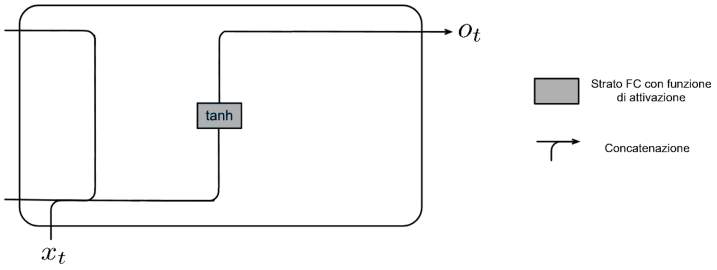

Figura 4.10: Stato nascosto di una RNN.

Le RNN tradizionali tuttavia presentano delle limitazioni, come la difficoltà nel gestire sequenze molto lunghe a causa della loro architettura fissa. In teoria, selezionando opportunamente i parametri, le RNN dovrebbero essere in grado di gestire tali dipendenze a lungo termine. Purtroppo nella pratica sembrano avere difficoltà ad apprenderli, come esaminato nel lavoro di S. Hochreiter [52] e di Y. Bengio et al. [53].

Per affrontare queste problematiche sono state sviluppate varianti più avanzate delle RNN, come le Long Short-Term Memory (LSTM) e le Gated Recurrent Unit (GRU) [8].

4.5 Long Short-Term Memory (LSTM)

Le Long Short-Term Memory (LSTM) [52], [8], [51] sono una variante avanzata delle reti neurali ricorrenti, sviluppata da S. Hochreiter e J. Schmidhuber [54] per superare la limitazione legata alla capacità di apprendere dipendenze a lungo termine nei dati sequenziali. Una delle innovazioni chiave riguarda l'introduzione di celle di memoria controllate da meccanismi di gating, che regolano quali informazioni devono essere mantenute, aggiornate o dimenticate. Nelle RNN tradizionali, il termine *stato nascosto* veniva utilizzato per descrivere il meccanismo principale di memorizzazione; con le LSTM, invece, si fa una distinzione chiara tra lo stato nascosto (o_t), ovvero l'output immediato della cella, e la cella di memoria (C_t), che funge da memoria a lungo termine della rete. Questo consente alla rete di superare il problema del *vanishing gradient* che affligge le RNN.

Dal punto di vista matematico la struttura delle LSTM è governata dalle seguenti equazioni fondamentali [8]:

1. **Input Gate**:
$$i_t = \sigma(W_i \cdot [o_{t-1}, x_t] + b_i)$$
L'input gate i_t determina quali nuove informazioni dall'input corrente x_t devono essere aggiunte allo stato della cella.
Nella formula σ è la funzione di attivazione sigmoide, W_i è la matrice dei pesi per il gate di input, o_{t-1} è l'output dello stato nascosto precedente, x_t è l'input corrente, e b_i è il bias per il gate di input.

2. **Forget Gate**:
$$f_t = \sigma(W_f \cdot [o_{t-1}, x_t] + b_f)$$
Il forget gate f_t determina quali informazioni mantenere dalla cella di memoria precedente. Agisce come se fosse un filtro delle informazioni.
Nella formula W_f è la matrice dei pesi per il forget gate e b_f è il bias per il forget gate.

3. **Aggiornamento dello stato della cella**:
$$\tilde{C}_t = \tanh(W_C \cdot [o_{t-1}, x_t] + b_C)$$

Il nuovo stato della cella $\tilde{C}_t$ viene generato a partire dall'input corrente x_t e dallo stato nascosto precedente o_{t-1}.

Nella formula tanh è la funzione di attivazione tangente iperbolica, W_C è la matrice dei pesi per il nuovo stato della cella e b_C è il bias per il nuovo stato della cella.

4. **Cella di Memoria**:
$$C_t = (f_t \otimes C_{t-1}) \oplus (i_t \otimes \tilde{C}_t)$$

In questa equazione C_t rappresenta lo stato della cella al tempo t, che viene aggiornato combinando le informazioni dalla cella precedente e il nuovo stato della cella tramite la moltiplicazione puntuale $\otimes$ e l'addizione puntuale $\oplus$.

5. **Output Gate**:
$$Z_t = \sigma(W_Z \cdot [o_{t-1}, x_t] + b_Z)$$

Il gate di output Z_t determina quali informazioni dallo stato della cella devono essere passate come output.

Nella formula W_C è la matrice dei pesi per l'output gate e b_C è il bias per l'output gate.

6. **Output Finale**:

$o_t = Z_t \otimes \tanh(C_t)$ $\quad \leftarrow$ ottenuto ricorsivamente (rispetto a t)

$y_t = o_t \otimes \tanh(C_t)$

Dove y_t è l'output della LSTM al tempo t .

Queste equazioni consentono alle LSTM di gestire sequenze complesse, conservando le informazioni utili e scartando quelle superflue. Tuttavia, a causa della loro struttura complessa, le LSTM richiedono un'attenta messa a punto che può renderle difficili da ottimizzare. Inoltre, le LSTM sono strati versatili che possono essere integrati in diverse architetture di reti neurali, facilitando l'inserimento di una componente sensibile al tempo all'interno di altri modelli.

Le LSTM sono particolarmente adatte per l'anomaly detection nelle serie temporali grazie alla loro capacità di gestire le dipendenze a lungo termine e di apprendere rappresentazioni complesse e non lineari. Dopo aver appreso il comportamento tipico/normale della serie temporale, le LSTM confrontano i valori previsti con quelli effettivi e classificano un punto come anomalo quando la differenza tra i due supera una soglia prestabilita (discussione approfondita nella Sezione 5.3).

La Figura precedente 4.9, che illustra la struttura di una RNN, può essere utilizzata anche per descrivere la struttura generale delle LSTM. In aggiunta, la Figura 4.11 mostra in modo dettagliato la struttura interna di una cella di memoria LSTM, che corrisponde a ciò che nella Figura 4.9 viene indicato come *stato nascosto*.

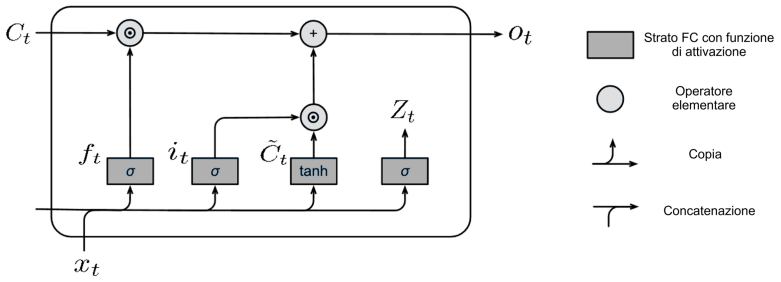

Figura 4.11: Cella di memoria di una LSTM.

4.6 Gated Recurrent Units (GRU)

Le Gated Recurrent Unit (GRU) [55], [8], [51], introdotte da Cho et al. [56], rappresentano un'evoluzione delle reti neurali ricorrenti, analogamente alle Long Short-Term Memory (LSTM). Nate per superare le limitazioni delle unità ricorrenti tradizionali, come le unità tanh, offrono un'alternativa più semplice ed efficiente rispetto alle LSTM.

La Figura 4.9, che illustra la struttura di una RNN, può essere utilizzata anche per rappresentare la struttura generale delle GRU. La Figura 4.12, invece, mostra nel dettaglio la struttura interna di una cella di memoria GRU, che nella Figura 4.9 corrisponde alla parte chiamata *stato nascosto*. Tale struttura può essere confrontata con quella della cella di memoria delle LSTM, rappresentata in Figura 4.11.

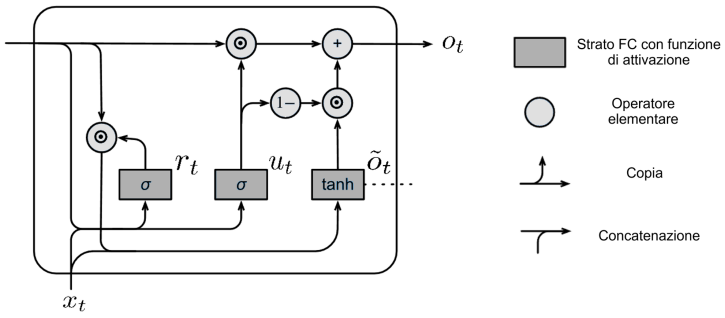

Figura 4.12: Cella di memoria di una GRU.

Le LSTM elaborano le sequenze in input utilizzando tre gate distinti (input, output e forget gate). Le GRU, invece, semplificano questo meccanismo combinando i gate di input e di forget in un unico gate, noto come gate di update. Inoltre introducono un gate di reset, che controlla quante sono le informazioni passate da dimenticare durante il calcolo del nuovo stato. Questa semplificazione consente di preservare la capacità di apprendimento delle dipendenze a lungo termine, riducendo al contempo la complessità computazionale.

Le equazioni fondamentali che governano il funzionamento delle GRU sono le seguenti [8]:

1. **Update Gate**:
$$u_t = \sigma(W_u \cdot [o_{t-1}, x_t] + b_u)$$

   Il gate di update u_t controlla quanto dello stato precedente o_{t-1} deve essere mantenuto e quanto deve essere aggiornato con nuove informazioni dell'input x_t, nel nuovo stato o_t. Se u_t è vicino a 1, significa che il modello decide di mantenere la maggior parte delle informazioni dallo stato precedente. Viceversa se uguale a 0.

   Nella formula σ è la funzione di attivazione sigmoide, W_u è la matrice dei pesi per il gate di update, o_{t-1} è lo stato nascosto precedente, x_t è l'input corrente, e b_u è il bias per il gate di update.

2. **Reset Gate**:
$$r_t = \sigma(W_r \cdot [o_{t-1}, x_t] + b_r)$$

   Il gate di reset r_t determina quanto delle informazioni passate dovrebbe essere rimosso quando si calcola l'attivazione candidata $\tilde{o}_t$. Permette al modello di resettare le informazioni passate in situazioni in cui non sono più rilevanti.

   Nella formula W_r è la matrice dei pesi per il reset gate e b_r è il bias per il reset gate.

3. **Attivazione Candidata**:

$$\tilde{o}_t = \tanh(W_{\tilde{o}} \cdot [r_t \otimes o_{t-1}, x_t] + b_{\tilde{o}})$$

   L'attivazione candidata $\tilde{o}_t$ è generata dall'input corrente x_t, e dallo stato nascosto precedente o_{t-1} modificato dal reset gate r_t.

   Nella formula tanh è la funzione di attivazione tangente iperbolica, $\otimes$ è l'operazione di moltiplicazione puntuale, $W_{\tilde{o}}$ è la matrice dei pesi per l'attivazione candidata e $b_{\tilde{o}}$ è il bias per l'attivazione candidata.

4. **Output Finale**:

   $o_t = [(1 - u_t) \otimes o_{t-1}] \oplus [u_t \otimes \tilde{o}_t] \quad \leftarrow$ ottenuto ricorsivamente (rispetto a t)

   $y_t = o_t$

   Nella prima equazione, o_t rappresenta lo stato nascosto al tempo t, aggiornato combinando le informazioni dallo stato nascosto precedente e l'attivazione candidata.

   Nella seconda equazione y_t è l'output della GRU al tempo t.

Di conseguenza le GRU hanno un numero minore di parametri rispetto alle LSTM, risultando particolarmente vantaggiose in scenari con risorse computazionali limitate. Gli studi di Chung et al. [55] evidenziano che, in alcune circostanze, le GRU possono offrire una convergenza più rapida e una migliore capacità di generalizzazione rispetto alle LSTM, a seconda del dataset e del compito specifico. Tuttavia, i risultati di questo lavoro (Capitolo 6) mostrano che le differenze tra LSTM e GRU sono minime, sia in termini di F1 score (approfondito in Sezione 5.2) che di tempi di calcolo.

4.7 LSTM-Autoencoder (LSTM-AE)

Gli LSTM-autoencoder (LSTM-AE) [56], [57], [58], [59] rappresentano un modello ibrido che combina le potenzialità delle reti LSTM nella gestione delle sequenze temporali con l'efficacia degli autoencoder nella compressione e ricostruzione dei dati, offrendo così un approccio più robusto per l'elaborazione di dati sequenziali.

Come si è visto nella Sezione 4.3, gli autoencoder sono costruiti per apprendere una rappresentazione compressa dei dati, riducendo la dimensionalità e preservando le caratteristiche più significative. Tuttavia, essi presentano limitazioni nel catturare le dipendenze temporali presenti in dati sequenziali complessi, come quelli derivanti dalle vibrazioni di macchine elettriche analizzati nel lavoro di Lachekhab et al. [58]. Questo avviene perché gli autoencoder non sono modelli autoregressivi e non dispongono di una memoria delle informazioni passate. È in questo contesto che l'architettura LSTM, descritta nella Sezione 4.5, si rivela particolarmente efficace grazie alla sua capacità di essere sensibile al tempo.

L'integrazione delle LSTM all'interno di un'architettura autoencoder permette di sfruttare i punti di forza di entrambi i modelli. L'autoencoder contribuisce con la sua capacità di apprendimento di rappresentazioni compatte dei dati, mentre gli strati LSTM garantiscono la conservazione delle informazioni nel tempo. Questa sinergia consente al modello di analizzare grandi quantità di dati sequenziali in modo più efficace [58].

L'architettura degli LSTM-autoencoder può essere rappresentata graficamente come nella Figura 4.13.

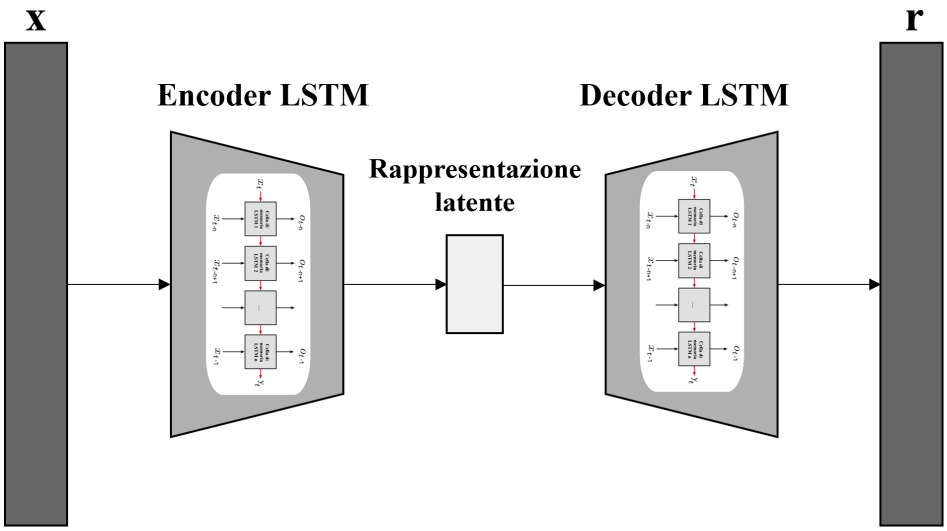

Figura 4.13: Architettura LSTM-autoencoder.

Anche in questo caso, il modello si compone di un encoder e un decoder, ma la loro struttura interna si differenzia da quella degli autoencoder tradizionali, come analizzato di seguito.

- **Encoder LSTM**: utilizza unità LSTM al posto degli strati lineari che, oltre a elaborare le sequenze temporali, permettono di catturare le dipendenze e le dinamiche temporali presenti nei dati di input.
- **Decoder LSTM**: dopo la compressione operata dall'encoder, il decoder, anch'esso composto da strati LSTM, ricostruisce la sequenza originale. Come visto nella Sezione 4.3, il decoder sfrutta la rappresentazione latente generata dall'encoder per generare l'output, cercando di minimizzare la differenza tra l'input originale e l'output ricostruito.

La differenza rispetto all'architettura degli autoencoder tradizionali illustrata nella Figura 4.7 consiste nel fatto che non ci sono connessioni tra tutti i neuroni, ma al loro posto ci sono degli strati ricorrenti (Figura 4.9) composti dalle celle di memoria di una LSTM (Figura 4.11). Dal punto di vista matematico, il processo può essere descritto dalla stessa equazione 4.19, sostituendo però f e g con le rispettive versioni LSTM f_{LSTM} e g_{LSTM}.

In un contesto di anomaly detection, l'LSTM-autoencoder viene addestrato a ricostruire i dati che riflettono il comportamento tipico/normale del sistema [59], analogamente all'autoencoder, come descritto nella Sezione 4.3.

Gli LSTM-autoencoder presentano vantaggi significativi rispetto agli autoencoder e agli LSTM tradizionali nella gestione di dati temporali complessi. Tuttavia, l'impiego di più livelli LSTM richiede un'attenta regolazione degli iperparametri per evitare l'overfitting e l'esplosione dei gradienti. Inoltre, questi modelli sono impegnativi dal punto di vista computazionale e spesso richiedono potenti GPU per garantire un addestramento efficiente.

Capitolo 5

Configurazione degli esperimenti

Questa sezione fornisce una descrizione dettagliata della configurazione sperimentale adottata per valutare i metodi di rilevamento delle anomalie proposti nel Capitolo 4 e le tecniche di preprocessing applicate ai dati nel Capitolo 3. L'obiettivo è garantire una comprensione chiara delle condizioni sperimentali e dei criteri di valutazione utilizzati.

5.1 Dataset

Gli esperimenti sono stati condotti su **quattro dataset**, ciascuno con caratteristiche distinte che riflettono vari aspetti dell'industria di processo. Gli algoritmi di rilevamento delle anomalie sono stati valutati nel contesto pratico specifico di questo lavoro, in cui non sono disponibili etichette di anomalia precise, ma i dati sono suddivisi in periodi noti per essere privi di anomalie e periodi noti per contenerle (maggiori dettagli nel Capitolo 1). I dataset sono strutturati in modo specifico per riflettere questo scenario: i file dei periodi privi di anomalie sono utilizzati per l'addestramento, mentre i file dei periodi con anomalie sono riservati ai test (come illustrato in Figura 1.1).

L'elenco puntato che segue fornisce una breve descrizione delle caratteristiche principali dei quattro dataset utilizzati nel presente studio. La Tabella 5.1 offre una sintesi delle informazioni principali.

- **SKAB** [25]: Lo Skoltech Anomaly Benchmark (SKAB) raccoglie dati da sensori installati su un banco di prova dotato di un sistema di circolazione dell'acqua, di un sistema di monitoraggio per la valutazione delle condizioni e di un sistema di controllo per la regolazione del funzionamento.

- **HAI** [60]: Il dataset sulla sicurezza HIL-based augmented ICS (HAI) raccoglie i dati di un testbed di un sistema di controllo industriale. Questo sistema è integrato

con un simulatore che emula la generazione di energia con turbine a vapore e la generazione di energia idroelettrica con pompaggio.

- **PSM** [18]: Il dataset Pooled Server Metrics (PSM) misura le variabili che monitorano le metriche interne dei nodi di un server in un ambiente applicativo condiviso.

- **WADI** [36]: Il dataset Water Distribution (WADI) raccoglie i dati di 16 giorni di funzionamento di un sistema di distribuzione dell'acqua, di cui 14 giorni in condizioni normali e 2 giorni con scenari di attacco.

Dataset	SKAB	HAI	PSM	WADI
Numero di variabili (colonne)	9	80	26	130
Numero di osservazioni (righe)	9 405	921 603	132 481	784 570
Frequenza di campionamento	1 secondo	1 secondo	1 minuto	1 minuto
Fonte	`github`	`github`	`github`	`iTrust`

Tabella 5.1: Panoramica dei dataset (il numero di osservazioni è relativo al dataset di training).

I dati PSM [18] utilizzati in questa tesi sono concessi da eBay secondo le condizioni della licenza `Creative Commons Attribution 4.0 International License`, mentre i dati WADI sono accreditati a "iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design".

5.2 Metriche

Per analizzare le prestazioni di un modello di anomaly detection è utile introdurre il concetto di **matrice di confusione** [41], [42], uno strumento che riassume il confronto tra le previsioni del modello e le effettive classi o etichette dei dati. Nella matrice ogni riga rappresenta la classe reale dell'osservazione, distinguendo tra dati normali (0, assenza di anomalia) e dati anomali (1, presenza di anomalia), mentre ogni colonna indica la classe assegnata dal modello. L'elemento in posizione (i, j) indica quante volte un'osservazione della classe effettiva i è stata classificata come appartenente alla classe predetta j. Questo strumento consente di individuare eventuali errori ricorrenti nelle previsioni, evidenziando la tendenza del modello a confondere le due categorie. La matrice di confusione è strutturata nel seguente modo in Figura 5.1:

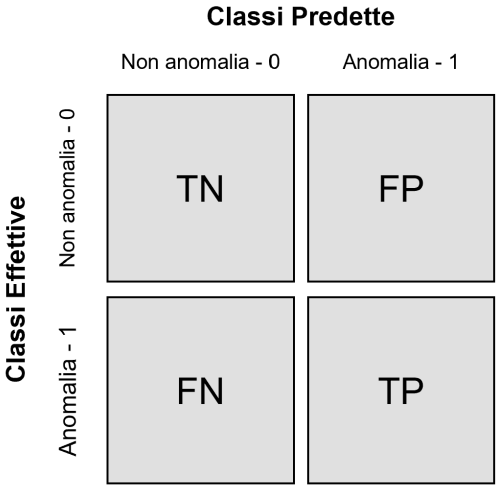

Figura 5.1: Matrice di confusione.

Dove:

- **True Negatives (TN)**: numero di osservazioni normali correttamente classificate come tali;
- **False Positives (FP)**: numero di osservazioni normali erroneamente classificate come anomalie (falsi allarmi),
- **False Negatives (FN)**: numero di anomalie che il modello non ha rilevato (anomalie mancate);
- **True Positives (TP)**: numero di anomalie correttamente identificate dal modello.

La Figura 5.2 mostra un esempio di matrice di confusione ottenuta applicando un autoencoder al dataset SKAB [25].

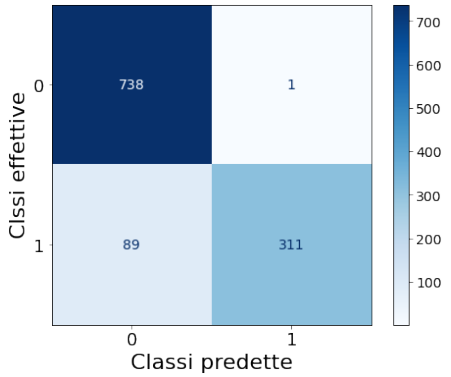

Figura 5.2: Esempio di matrice di confusione, applicata al dataset SKAB.

A partire dalla matrice di confusione è possibile definire diverse metriche di valutazione, tra cui accuracy, precision, recall e F1 score [41], [42], ciascuna con un ruolo specifico nell'analisi delle prestazioni del modello. Tuttavia, nel contesto dell'anomaly detection, l'accuracy può risultare fuorviante, poiché le anomalie sono spesso eventi rari e un modello che classifica quasi tutte le osservazioni come normali potrebbe comunque ottenere un'elevata accuratezza (maggiori dettagli in Kroese et al. [41]).

Per questo motivo in questa tesi è stato adottato l'**F1 score** [41] come metrica principale per la valutazione delle prestazioni dei modelli. L'F1 score combina precision e recall in un unico valore, offrendo un indicatore bilanciato che consente di misurare l'efficacia del modello nel riconoscere le anomalie senza trascurare il compromesso tra falsi positivi e falsi negativi. Diversi studi nel campo dell'anomaly detection adottano un approccio simile, tra cui quelli di Choi et al. [9] e Darban et al. [8]. Altri lavori, come quelli di Schmidl et al. [6] e Belay et al. [7], includono sempre l'F1 score tra le metriche di valutazione, ma affiancandolo anche a indicatori come AUC e AUPR (per un approfondimento su queste metriche, si veda James et al. [42]).

L'F1 score è definito come la media armonica tra precision e recall:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{5.1}$$

dove:

- La **precision** [41] misura l'accuratezza delle previsioni positive di un modello, ovvero la proporzione di anomalie identificate correttamente (TP) rispetto al totale delle istanze etichettate dal modello come anomale (TP + FP). È definita come:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{5.2}$$

  Un valore elevato di precision indica un numero ridotto di falsi positivi, ossia minori segnalazioni errate di anomalie. Come nel caso dell'esempio in Figura 5.2.

- La **recall** [41] misura la capacità di identificare tutte le anomalie effettive, ovvero la proporzione di anomalie identificate correttamente (TP) rispetto al totale delle istanze realmente anomale (TP + FN). È definita come:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{5.3}$$

  Un valore alto di recall indica un numero minore di falsi negativi, ossia un minor numero di anomalie mancate.

Per quantificare l'errore sia nella ricostruzione che nella previsione (maggiori dettagli nella Sezione 2.1) si utilizza l'**errore quadratico medio (MSE)**, che rappresenta la scelta più diffusa per questo tipo di compiti. L'MSE è definito come la media dei quadrati degli errori:

$$\text{MSE} = \frac{1}{T}\sum_{t=1}^{T} \|x_t - \hat{x}_t\|^2 ,$$

dove x_t rappresenta il valore effettivo, $\hat{x}_t$ il valore ricostruito o previsto e T rappresenta il numero totale di osservazioni.

5.3 Criterio di rilevamento delle anomalie

Il rilevamento delle anomalie (discusso anche nella Sezione 2.1) si basa su una **soglia** stabilita in base al **Mean Squared Error (MSE)** calcolato nella fase di addestramento. Il modello viene allenato su dati privi di anomalie, dove impara a ricostruire o prevedere il comportamento normale della serie temporale. Una volta completato l'addestramento l'MSE viene calcolato per ogni osservazione, misurando la differenza tra i valori osservati e quelli ricostruiti o predetti dal modello. Se l'errore supera la soglia prestabilita il campione è classificato come anomalo, poiché un errore elevato indica che il modello non riesce a riprodurre correttamente il comportamento normale. Per determinare questa soglia sono stati presi in considerazione **due metodi**:

- **Metodo basato sul percentile (95%)**:

  Questo approccio stabilisce la soglia di anomalia al 95° percentile dei valori di MSE calcolati sul dataset di addestramento. In pratica, si assume che il 95% dei dati di addestramento rappresenti il comportamento normale del sistema, mentre il restante 5%, caratterizzato dagli errori più elevati, possa contenere anomalie. Durante la fase di test qualsiasi campione con un MSE superiore a questa soglia viene classificato come anomalo.

  La scelta del percentile, in questo caso fissato al 95%, può essere modificata a seconda delle esigenze: un percentile più elevato privilegia la precision, riducendo i falsi positivi, mentre uno inferiore consente di rilevare più anomalie, aumentando la recall.

- **Metodo del gomito (elbow method)** [61]:

  Questa tecnica euristica consiste nel tracciare il grafico dei valori di MSE e individuare il punto in cui la curva cambia drasticamente pendenza. Tale punto segna il passaggio da una crescita graduale dell'errore a un aumento più marcato ed è noto come *gomito* (elbow). Esso rappresenta un confine naturale tra i dati normali e quelli anomali, poiché oltre questa soglia l'errore tende a crescere in modo significativo. Sebbene l'identificazione visiva del gomito sia una pratica comune, il metodo può essere formalizzato matematicamente. In Satopaa et al. [61] viene approfondita la metodologia utilizzata per individuare il punto ottimale.

Si sceglie il **massimo** tra le due soglie individuate dai due metodi per seguire un approccio conservativo, riducendo il rischio di falsi positivi. In questo modo vengono classificate come anomalie solo le osservazioni con un errore di ricostruzione particolarmente elevato,

evitando che piccole deviazioni dal comportamento atteso vengano erroneamente segnalate. Per garantire l'affidabilità della soglia, il modello deve essere in grado di ricostruire i dati normali con un'accuratezza sufficiente, ossia con una funzione di perdita (equazione 4.13) prossima a zero durante la fase di addestramento. In caso contrario, un errore elevato potrebbe derivare da una scarsa capacità di apprendimento anziché dalla presenza effettiva di un'anomalia.

Un esempio grafico dell'applicazione di queste soglie al dataset PSM [18] è riportato in Figura 5.3.

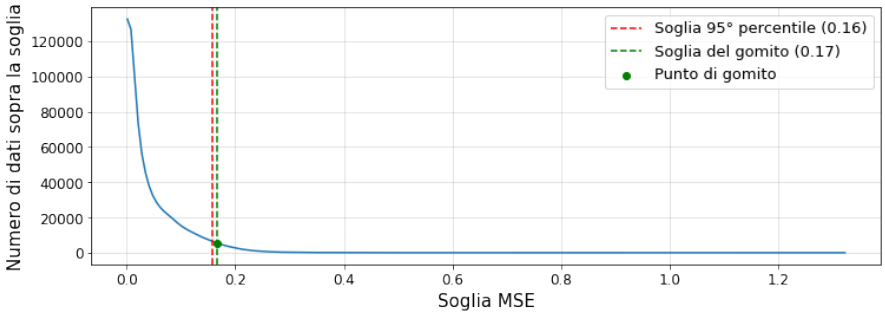

Figura 5.3: Esempio delle soglie MSE utilizzate per rilevare le anomalie.

Capitolo 6

Risultati

Questa sezione presenta i risultati e l'analisi delle ricerche a griglia condotte sui metodi di rilevamento delle anomalie proposti nel Capitolo 4, valutati utilizzando l'impostazione sperimentale descritta nel Capitolo 5 e impiegando le tecniche di preprocessing illustrate nel Capitolo 3.

6.1 Preprocessing

Per valutare l'impatto del preprocessing nella pipeline di rilevamento delle anomalie seguita (Figura 1.2), sono state eseguite delle ricerche a griglia complete sui quattro dataset considerati (Tabella 5.1).
La Figura 3.1 del Capitolo 3 riporta le diverse tecniche di preprocessing sottoposte a inclusione/esclusione nelle ricerche a griglia. Più precisamente sono state testate sistematicamente tutte le combinazioni riportate nella Tabella 6.1.

Fase di preprocessing:	Impiegata (sì ; no)	Iperparametri
Data imputation (3.1)	sì (se necessaria)	-
Rimozione colonne costanti (3.2)	sì ; no	-
Savitky-Golay (SG) (3.3)	sì ; no	Dimensione finestra: 5 ; 6 ; 7; Ordine polinomio: 2 ; 3 ; 4
Hodrick-Prescott (3.4)	sì ; no	1600 ; 144000 ; 129600
Normalizzazione (3.5)	sì ; no	StandardScaler ; RobustScaler ; MinMaxScaler
Feature selection (3.6)	sì ; no	Soglia: 'low' ; 'med' ; 'high'

Tabella 6.1: Fasi e iperparametri di preprocessing considerati.

Dall'analisi della Tabella (6.1) si suggeriscono indicativamente le due pipeline di preprocessing mostrate nella Figura 6.1, distinte in modelli sensibili al tempo (VARIMA, LSTM, GRU, LSTM-AE) e non sensibili al tempo (PCA, AE, DAE, SAE, CAE). Si precisa che

la pipeline per i modelli sensibili al tempo (o autoregressivi) viene suggerita quando il dataset di addestramento ha almeno 500 000 righe.

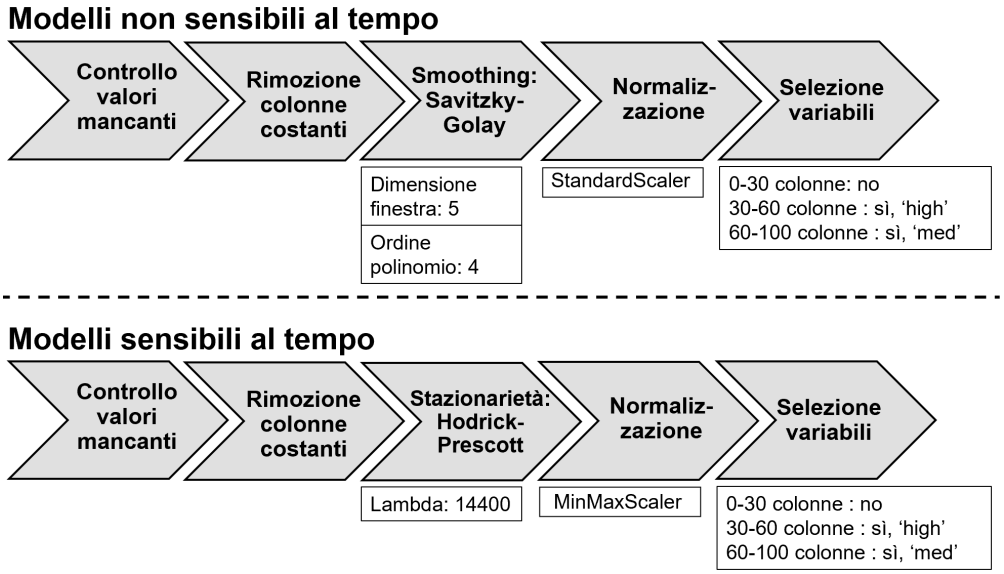

Figura 6.1: Pipeline di preprocessing proposte.

6.2 Metodi

Le configurazioni ottimali di preprocessing identificate nella Sezione precedente (6.1) sono state impiegate per confrontare le prestazioni, in termini di F1 score, dei metodi sui quattro dataset. Poiché i metodi si basano sul deep learning, è necessario regolare con precisione i parametri di addestramento [15], [62] per minimizzare la funzione di perdita. I parametri testati, sempre tramite ricerche a griglia complete, sono riassunti nella Tabella 6.2. Le configurazioni in **grassetto** sono linee guida generali che possono essere utilizzate quando non sono disponibili dataset etichettati e non è possibile l'ottimizzazione basata sulle prestazioni.

Learning rate	**0.001**	0.0001	-
Weight decay	0.0	**0.00001**	0.0001
Optimizer	**Adam**	SGD	-
Loss	MAE	**MSE**	-
Scheduler	None	**Reduce LR On Plateau**	Cosine Annealing LR
Batch size	**512**	1024	2048

Tabella 6.2: Parametri di addestramento.

In merito ai modelli basati su LSTM e GRU, un aspetto fondamentale è la lunghezza della sequenza, ovvero il numero di osservazioni precedenti considerate dal modello. Sono state testate sequenze di lunghezza pari a 10, 25, 50 e 100, con le migliori prestazioni complessive ottenute per una lunghezza di 50. Tuttavia, per dataset più semplici, in particolare quelli univariati, la scelta ottimale può essere guidata dall'analisi del grafico dell'autocorrelazione (approfondita nella Sezione 2.3.1).

La Tabella 6.3 riassume i **risultati finali**, riportando le prestazioni dei metodi ottimizzati in termini di F1 score. Come baseline (descritte nella Sezione 4.1) sono stati adottati due approcci statistici tradizionali: l'analisi delle componenti principali (PCA) per la ricostruzione e il modello vettoriale autoregressivo integrato con media mobile (VARIMA) per la previsione.

Dataset	SKAB	HAI	PSM	WADI
AE	0.882	0.390	0.537	0.381
DAE	0.897	0.422	0.535	**0.385**
SAE	0.887	0.404	0.567	0.377
CAE	0.873	0.372	0.518	0.295
LSTM	0.914	**0.679**	**0.623**	0.380
GRU	0.911	0.651	0.619	0.376
LSTM-AE	**0.926**	0.617	0.575	0.271
VARIMA	0.626	*	*	*
PCA	0.607	0.367	0.435	0.137

* *Indica dove il processo non converge in meno di 12 ore.*

Tabella 6.3: Risultati in termini di F1 score.

Per verificare la significatività dei risultati riportati nella tabella è stato eseguito un **test t di Student**. L'analisi ha mostrato che una differenza negli F1 score pari o superiore a 0.015 è statisticamente significativa, indicando una reale differenza tra i valori confrontati.

Il test t di Student (maggiori dettagli in Montgomery et al. [63]) è un test statistico utilizzato per confrontare le medie di due campioni e verificare se la loro differenza è significativa. Si basa sull'ipotesi nulla (H_0), che assume l'uguaglianza tra le due medie, ossia:

$$H_0 : \bar{x}_1 = \bar{x}_2 \, .$$

Il test calcola il valore t confrontando la differenza tra le medie con la variabilità dei dati. La formula generale per il t-test per campioni indipendenti (con stessa lunghezza) è:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2 + s_2^2}{n}}}, \tag{6.1}$$

dove $\bar{x}_1$, $\bar{x}_2$ sono le medie campionarie dei due campioni, n è la loro lunghezza, e s_1, s_2 sono le loro deviazioni standard campionarie.

Il valore t ottenuto viene confrontato con la soglia della distribuzione t di Student determinata dai gradi di libertà $(n - 1)$ e dal livello di significatività scelto (95%). Si rifiuta l'ipotesi nulla se il valore t supera questa soglia, concludendo che la differenza tra le due medie è statisticamente significativa.

In questo caso ogni valore in tabella è stato calcolato 50 volte, modificando il seme del generatore di numeri casuali, così da ottenere la media e la varianza campionaria di ciascun risultato.

6.3 Discussione dei risultati

I risultati evidenziano l'importanza del preprocessing nell'analisi delle serie temporali multivariate, come già sottolineato da Tawakuli et al. [14]. In tutte le ricerche a griglia, infatti, i modelli che includevano fasi di preprocessing sono stati sistematicamente selezionati come i più performanti. Inoltre, i modelli predittivi LSTM e GRU si sono rivelati particolarmente efficaci nel contesto industriale di broad classification, confermando la necessità di valutare sia approcci di ricostruzione che di previsione. Le loro prestazioni migliorano ulteriormente quando affiancati da tecniche di preprocessing che garantiscono la stazionarietà, in questo caso il filtro di Hodrick-Prescott.

Nessun modello ha dominato sugli altri, in linea con quanto riportato da Zope et al. [12], dove tecniche come PCA, AE e LSTM-AE hanno mostrato risultati variabili a seconda del dataset. Tuttavia, a differenza di quel lavoro, in questo studio i metodi basati su autoencoder e reti LSTM/GRU hanno costantemente superato i modelli statistici di base (PCA) nei dataset industriali analizzati.

Nel complesso questi risultati confermano che le architetture basate su deep learning, in particolare autoencoder e reti ricorrenti come LSTM e GRU, abbinate a un preprocessing ottimizzato, rappresentano una soluzione efficace per il rilevamento delle anomalie in scenari industriali reali.

Capitolo 7

Conclusione

Il corrente studio presenta una pipeline ottimizzata per il rilevamento delle anomalie multivariate in ambito industriale, che integra tecniche avanzate di preprocessing, regolazione automatica della soglia di errore e l'utilizzo di reti neurali. La metodologia è stata testata su dataset provenienti dall'industria di processo con caratteristiche e scale diverse, rivelandosi particolarmente utile per l'applicazione a dataset privi di dati etichettati. L'intero studio è stato infatti sviluppato per massimizzare l'efficacia di questa applicazione, guidando ogni scelta metodologica verso una soluzione funzionale e generalizzabile.

L'analisi condotta si concentra sulla selezione, valutazione e ottimizzazione delle tecniche di preprocessing proposte da Tawakuli et al. [14]. Questo lavoro colma la mancanza di pipeline dettagliate e ottimizzate nella letteratura sull'anomaly detection in ambito industriale. Inoltre, affronta il problema della selezione della soglia di errore, spesso trascurato negli studi esistenti. Ad esempio, Pota et al. [2] determinano la soglia in modo non generalizzabile ottimizzandola in base all'F1 score di un singolo dataset etichettato. L'approccio adottato in questo lavoro, invece, è progettato per essere applicabile anche in assenza di etichette. A differenza delle soglie basate sui percentili adottate da Panza et al. [11] e Zope et al. [12], il metodo proposto (descritto nella Sezione 5.3) offre una strategia più robusta, adattandosi meglio alle caratteristiche specifiche del dataset di addestramento. Il metodo del gomito si dimostra infatti vantaggioso in situazioni in cui ci sono numerosi outlier o nessun outlier, poiché tiene conto dell'errore di ricostruzione (o previsione) più elevato per questi punti.

Il lavoro contribuisce inoltre alle future direzioni di ricerca suggerite da Panza et al. [11]. Infatti, il metodo di soglia che seleziona il valore massimo tra due soglie aiuta a ridurre i falsi positivi nei modelli non supervisionati, mentre il filtro Savitzky-Golay migliora l'estrazione di informazioni rilevanti dai segnali industriali grezzi. Infine, i risultati ottenuti confermano una delle principali conclusioni di West et al. [13], evidenziando la necessità di un preprocessing accurato e di un'ottimizzazione attenta dei parametri per migliorare le prestazioni dei modelli nelle applicazioni industriali.

7.1 Estensioni e Lavoro Futuro

Il lavoro futuro si può concentrare sull'ampliamento di questo studio, testando nuovi dataset e metodi per rafforzare ulteriormente la validità dell'approccio proposto. Un possibile dataset di interesse è SWaT [64], che raccoglie 11 giorni di dati operativi continui di un sistema di trattamento delle acque. Per quanto riguarda i metodi, l'esplorazione di altre architetture, come SAE-1SVM [65], GRU-Autoencoder [66], Variational Autoencoder (VAE) [8], e LSTM-VAE [8] rappresenta uno sviluppo naturale. Infine, l'integrazione di Optuna [67], un framework per l'ottimizzazione automatica degli iperparametri, potrebbe rendere più efficiente la fase di tuning rispetto alle tradizionali ricerche a griglia.

Nei quattro dataset analizzati non è stato necessario eliminare i periodi di *ramp-up* e *ramp-down*, ovvero le fasi transitorie in cui il sistema non ha ancora raggiunto il regime operativo e quindi il suo comportamento non riflette le normali condizioni di funzionamento [2]. Tuttavia, in futuri dataset che includano tali periodi, la loro rimozione potrebbe migliorare le prestazioni dei modelli applicati successivamente. In questi casi la pipeline di preprocessing da testare seguirebbe lo schema illustrato in Figura 7.1.

Figura 7.1: Pipeline di pre-processing per dataset con periodi di ramp-up e ramp-down.

Inoltre, un'altra possibilità di lavoro futuro riguarda l'identificazione delle cause delle anomalie. La pipeline proposta mantiene l'interpretabilità del dataset, permettendo di applicare tecniche di analisi delle cause profonde (RCA) [68], [69], un aspetto essenziale nelle applicazioni industriali. Un approccio per rendere le anomalie più interpretabili consiste nell'analizzare il contributo delle singole variabili all'errore di ricostruzione nei momenti in cui si verificano eventi anomali [70]. Tuttavia, questa tecnica tende a evidenziare gli effetti più evidenti delle anomalie piuttosto che le loro cause reali. Per validare ulteriormente le metodologie RCA si potrebbe utilizzare il dataset HAI [60], che include una documentazione dettagliata delle anomalie, fornendo informazioni sia sulle loro cause che sui relativi effetti. L'impiego di dataset ben documentati consentirebbe un'analisi più approfondita delle tecniche RCA, contribuendo allo sviluppo di strumenti di anomaly detection più robusti e interpretabili, in grado di individuare non solo le anomalie, ma anche le loro origini.

Bibliografia

[1] M. de Castro-Cros et al., "Machine-learning-based condition assessment of gas turbines—a review," *Energies*, vol. 14, no. 24, 2021. [Online]. Available: https://www.mdpi.com/1996-1073/14/24/8468

[2] M. Pota, G. De Pietro, and M. Esposito, "Real-time anomaly detection on time series of industrial furnaces: A comparison of autoencoder architectures," *Engineering Applications of Artificial Intelligence*, vol. 124, p. 106597, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197623007819

[3] Z. T. et al., "Anomaly detection using spatial and temporal information in multivariate time series," *Scientific Reports*, 2023. [Online]. Available: https://doi.org/10.1038/s41598-023-31193-8

[4] S. C. et al., "Anomaly detection in financial time series by principal component analysis and neural networks," *Algorithms*, vol. 15, no. 10, 2022. [Online]. Available: https://www.mdpi.com/1999-4893/15/10/385

[5] X. Yang, X. Qi, and X. Zhou, "Deep learning technologies for time series anomaly detection in healthcare: A review," *IEEE Access*, vol. 11, pp. 117 788–117 799, 2023.

[6] S. Schmidl, P. Wenig, and T. Papenbrock, "Anomaly detection in time series: a comprehensive evaluation," *Proc. VLDB Endow.*, vol. 15, no. 9, p. 1779–1797, May 2022. [Online]. Available: https://doi.org/10.14778/3538598.3538602

[7] M. A. Belay, S. S. Blakseth, A. Rasheed, and P. Salvo Rossi, "Unsupervised anomaly detection for iot-based multivariate time series: Existing solutions, performance analysis and future directions," *Sensors*, vol. 23, no. 5, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/5/2844

[8] Z. Z. D. et al., "Deep learning for time series anomaly detection: A survey," *ACM Computing Surveys*, vol. 57, no. 1, p. 1–42, 2024. [Online]. Available: http://dx.doi.org/10.1145/3691338

[9] K. C. et al., "Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines," *IEEE Access*, vol. 9, pp. 120 043–120 065, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3107975

[10] L. R. et al., "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, 2021.

[11] M. A. Panza, M. Pota, and M. Esposito, "Anomaly detection methods for industrial applications: A comparative study," *Electronics*, vol. 12, no. 18, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/18/3971

[12] K. Z. et al., "Anomaly detection and diagnosis in manufacturing systems: A comparative study of statistical, machine learning and deep learning techniques," *Annual Conference of the PHM Society*, 2019.

[13] N. West, "A comparative study of machine learning approaches for anomaly detection in industrial screw driving data," in *57th Hawaii International Conference on System Sciences*, 2024.

[14] A. Tawakuli, B. Havers, V. Gulisano, D. Kaiser, and T. Engel, "Survey:time-series data preprocessing: A survey and an empirical analysis," *Journal of Engineering Research*, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2307187724000452

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[16] V. C. et al., "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, Jul. 2009. [Online]. Available: https://doi.org/10.1145/1541880.1541882

[17] G. P. et al., "Deep learning for anomaly detection: A review," *ACM Computing Surveys*, vol. 54, no. 2, p. 1–38, Mar. 2021. [Online]. Available: http://dx.doi.org/10.1145/3439950

[18] eBay AI Research, "Ransyncoders - data repository," https://github.com/eBay/RANSynCoders/tree/main/data, 2023, accessed: 2025-02-07.

[19] Y. Dodge, *The Concise Encyclopedia of Statistics*. Springer New York, NY, 2008. [Online]. Available: https://doi.org/10.1007/978-0-387-32833-1

[20] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[21] R. H. S. et al., *Time Series Analysis and Its Applications With R Examples*. Springer, 2017. [Online]. Available: https://doi.org/10.1007/978-3-319-52452-8

[22] R. J. H. et al., *Forecasting: Principles and Practice*. OTexts: Melbourne, Australia, 2021. [Online]. Available: https://otexts.com/fpp3/

[23] R. S. et al., "Nonstationary time series transformation methods: An experimental review," *Knowledge-Based Systems*, 2019. [Online]. Available: https://doi.org/10.1016/j.knosys.2018.10.041

[24] W. W. S. W. et al., *Multivariate Time Series Analysis and Applications*. Wiley, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/book/10.1002/9781119502951

[25] I. D. Katser and V. O. Kozitsin, "Skoltech anomaly benchmark (skab)," https://www.kaggle.com/dsv/1693952, 2020.

[26] R. B. C. et al., "Stl: A seasonal-trend decomposition procedure based on loess," *Journal of Official Statistics*, 1990. [Online]. Available: https://www.scb.se/contentassets/ca21efb41fee47d293bbee5bf7be7fb3/stl-a-seasonal-trend-decomposition-procedure-based-on-loess.pdf

[27] B. Kim, M. A. Alawami, E. Kim, S. Oh, J. Park, and H. Kim, "A comparative study of time series anomaly detection models for industrial control systems," *Sensors*, vol. 23, no. 3, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/3/1310

[28] J. M. Blackledge, *Digital Signal Processing*. Horwood Publishing, 2006. [Online]. Available: https://www.sciencedirect.com/book/9781904275268/digital-signal-processing

[29] R. W. Schafer, "What is a savitzky-golay filter? [lecture notes]," *IEEE Signal Processing Magazine*, vol. 28, no. 4, pp. 111–117, 2011.

[30] K. Berezowski, "Transistor chaining with integrated dynamic folding for 1-d leaf cell synthesis," in *Proceedings Euromicro Symposium on Digital Systems Design*, 2001, pp. 422–429.

[31] R. J. Hodrick and E. C. Prescott, "Postwar u.s. business cycles: An empirical investigation," *Journal of Money, Credit and Banking*, 1997. [Online]. Available: https://doi.org/10.2307/2953682

[32] M. O. Ravn and H. Uhlig, "On adjusting the hp-filter for the frequency of observations," Munich, CESifo Working Paper 479, 2001. [Online]. Available: https://hdl.handle.net/10419/75742

[33] S. G. et al., *Data Preprocessing in Data Mining*. Springer Cham, 2014. [Online]. Available: https://doi.org/10.1007/978-3-319-10247-4

[34] K. Michalak and H. Kwasnicka, "Correlation-based feature selection strategy in neural classification," in *Sixth International Conference on Intelligent Systems Design and Applications*, vol. 1, 2006, pp. 741–746.

[35] E. F. E.-H. et al., "A comparison of the pearson, spearman rank and kendall tau correlation coefficients using quantitative variables," *Asian Journal of Probability and Statistics*, 2022. [Online]. Available: https://doi.org/10.9734/ajpas/2022/v20i3425

[36] C. M. Ahmed, V. R. Palleti, and A. P. Mathur, "Wadi: a water distribution testbed for research in the design of secure cyber physical systems," in *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*, ser. CySWATER '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 25–28. [Online]. Available: https://doi.org/10.1145/3055366.3055375

[37] J. Korstanje, *Advanced Forecasting with Python*. Springer, 2021. [Online]. Available: https://doi.org/10.1007/978-1-4842-7150-6

[38] P. Developers, "pmdarima.arima.autoarima documentation," https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html, 2023.

[39] S. Team, "State space models - sarimax example," https://www.statsmodels.org/dev/examples/notebooks/generated/statespace_sarimax_stata.html, 2025.

[40] H. H. Kishan G. Mehrotra, Chilukuri K. Mohan, *Anomaly Detection Principles and Algorithms*. Springer Cham, 2017. [Online]. Available: https://doi.org/10.1007/978-3-319-67526-8

[41] D. P. K. et al., *Data Science and Machine Learning: Mathematical and Statistical Methods*, ser. Chapman & Hall/CRC machine learning & pattern recognition. Boca Raton: CRC Press, 2019. [Online]. Available: https://people.smp.uq.edu.au/DirkKroese/DSML/

[42] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning with Applications in Python*. Springer, 2023. [Online]. Available: https://www.statlearning.com/

[43] A. Dhalla. (2021) The math of principal component analysis (pca). [Online]. Available: https://medium.com/analytics-vidhya/the-math-of-principal-component-analysis-pca-bf7da48247fc

[44] J. A. Rice, *Mathematical Statistics and Data Analysis*. Duxbury, 2007. [Online]. Available: https://www.stat.berkeley.edu/~rice/Book3ed/index.html

[45] K. P. Murphy, *Machine Learning: a Probabilistic Perspective*. MIT Press, 2012. [Online]. Available: https://probml.github.io/pml-book/book0.html

[46] S. J. Prince, *Understanding Deep Learning*. The MIT Press, 2023. [Online]. Available: http://udlbook.com

[47] C. M. Bishop and H. Bishop, *Deep Learning Foundations and Concepts*. Springer, 2024. [Online]. Available: https://www.bishopbook.com/

[48] P. V. et al., "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 1096–1103. [Online]. Available: https://doi.org/10.1145/1390156.1390294

[49] A. Ng, "Sparse autoencoder," *Stanford University*, 2011. [Online]. Available: https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf

[50] G. L. et al., "Unsupervised anomaly detection of the gas turbine operation via convolutional auto-encoder," in *2020 IEEE International Conference on Prognostics and Health Management (ICPHM)*, 2020, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9187054

[51] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into Deep Learning*. Cambridge University Press, 2023, https://D2L.ai.

[52] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen netzen," Ph.D. dissertation, Institut fur Informatik, Technische Universitat, Munchen, 1991. [Online]. Available: https://people.idsia.ch/~juergen/SeppHochreiter1991ThesisAdvisorSchmidhuber.pdf

[53] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[54] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, 1997.

[55] J. C. et al., "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014. [Online]. Available: https://arxiv.org/abs/1412.3555

[56] K. C. et al., "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014. [Online]. Available: https://arxiv.org/abs/1406.1078

[57] O. I. P. et al., "Unsupervised anomaly detection in time series using lstm-based autoencoders," in *2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT)*, 2019, pp. 513–517.

[58] F. L. et al., "Lstm-autoencoder deep learning model for anomaly detection in electric motor," *Energies*, 2024. [Online]. Available: https://doi.org/10.3390/en17102340

[59] Y. W. et al., "Reconstruction-based lstm-autoencoder for anomaly-based ddos attack detection over multivariate time-series data," 2023. [Online]. Available: https://arxiv.org/abs/2305.09475

[60] H.-K. Shin, W. Lee, J.-H. Yun, and B.-G. Min, "Two ics security datasets and anomaly detection contest on the hil-based augmented ics testbed," in *Cyber Security Experimentation and Test Workshop*, ser. CSET '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 36–40. [Online]. Available: https://doi.org/10.1145/3474718.3474719

[61] V. S. et al., "Finding a "kneedle" in a haystack: Detecting knee points in system behavior," in *2011 31st International Conference on Distributed Computing Systems Workshops*, 2011, pp. 166–171. [Online]. Available: https://doi.org/10.1109/ICDCSW.2011.20

[62] A. P. et al., "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[63] G. C. R. D. C. Montgomery, *Applied Statistics and Probability for Engineers*. Wiley, 2018. [Online]. Available: https://www.wiley.com/en-us/Applied+Statistics+and+Probability+for+Engineers%2C+7th+Edition-p-9781119400363

[64] A. P. Mathur and N. O. Tippenhauer, "Swat: a water treatment testbed for research and training on ics security," in *2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*, 2016, pp. 31–36.

[65] L. Mhamdi, D. McLernon, F. El-moussa, S. A. Raza Zaidi, M. Ghogho, and T. Tang, "A deep learning approach combining autoencoder with one-class svm for ddos attack detection in sdns," in *2020 IEEE Eighth International Conference on Communications and Networking (ComNet)*, 2020, pp. 1–6.

[66] J. Cowton, I. Kyriazakis, T. Plötz, and J. Bacardit, "A combined deep learning gru-autoencoder for the early detection of respiratory disease in pigs using multiple environmental sensors," *Sensors*, vol. 18, no. 8, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/8/2521

[67] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[68] M. Solé, V. Muntés-Mulero, A. I. Rana, and G. Estrada, "Survey on models and techniques for root-cause analysis," 2017. [Online]. Available: https://arxiv.org/abs/1701.08546

[69] K. Papageorgiou, T. Theodosiou, A. Rapti, E. I. Papageorgiou, N. Dimitriou, D. Tzovaras, and G. Margetis, "A systematic review on machine learning methods for root cause analysis towards zero-defect manufacturing," *Frontiers in Manufacturing Technology*, vol. 2, 2022. [Online]. Available: https://www.frontiersin.org/journals/manufacturing-technology/articles/10.3389/fmtec.2022.972712

[70] C. M. Roelofs, M.-A. Lutz, S. Faulstich, and S. Vogt, "Autoencoder-based anomaly root cause analysis for wind turbines," *Energy and AI*, vol. 4, p. 100065, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666546821000197