Tecniche di Deep Learning per la classificazione predittiva dei guasti in sistemi off-grid alimentati da pannelli solari

Facoltà di Ingegneria dell'informazione, informatica e statistica

Dipartimento di Ingegneria dell'Informazione, Elettronica e Telecomunicazioni  
Corso di laurea in Ingegneria Elettronica

Lorenzo Giraldi

1649909

Relatore Correlatore

Massimo Panella Daniele Nerone

A.A. 2021-2022

[I. Introduzione 3](#_Toc102654050)

[II. Deep Learning 6](#_Toc102654051)

[A. Recursive Neural Network - RNN 6](#_Toc102654052)

[B. Long Short Term Memory - LSTM 12](#_Toc102654053)

[III. Manutenzione predittiva 15](#_Toc102654054)

[A. Che cos’è e perché si usa 15](#_Toc102654055)

[B. Stato dell’arte 19](#_Toc102654056)

[IV. Architettura del sistema 21](#_Toc102654057)

[A. Acquisizione e trasmissione dei dati 21](#_Toc102654058)

[B. Salvataggio dei campioni nel database 26](#_Toc102654059)

[V. Elaborazione dei dati 27](#_Toc102654060)

[A. Query 27](#_Toc102654061)

[B. Prima preelaborazione dei dati (Sincronizzazione) 29](#_Toc102654062)

[C. Estrazione di sequenze lunghe 3 giorni 37](#_Toc102654063)

[1. Estrazione delle sequenze 39](#_Toc102654064)

[2. Identificazione delle sequenze sane e patologiche 46](#_Toc102654065)

[3. Normalizzazione delle sequenze 58](#_Toc102654066)

[4. Etichettatura delle sequenze 60](#_Toc102654067)

[5. Salvataggio 62](#_Toc102654068)

[6. Creazione del dataset con le sequenze di tutti i tralicci 62](#_Toc102654069)

[7. Salvataggio del dataset 70](#_Toc102654070)

[D. Estrazione di sequenze lunghe 1 giorno 71](#_Toc102654071)

[VI. Rete Neurale 72](#_Toc102654072)

[A. Layers 72](#_Toc102654073)

[B. Parametri 73](#_Toc102654074)

[C. Training Options 73](#_Toc102654075)

[D. Training e classificazione 75](#_Toc102654076)

[E. K-Folding 77](#_Toc102654077)

[VII. Risultati 79](#_Toc102654078)

[A. Tensione della cella minima e potenza del pannello 79](#_Toc102654079)

[1. Predizione ad 1 giorno con sequenze lunghe 3 giorni 79](#_Toc102654080)

[2. Predizione ad 1 giorno con sequenze lunghe 1 giorno 80](#_Toc102654081)

[3. Predizione a 3 giorni con sequenze lunghe 3 giorni 81](#_Toc102654082)

[4. Predizione a 3 giorni con sequenze lunghe 1 giorno 81](#_Toc102654083)

[5. Predizione a 7 giorni con sequenze lunghe 3 giorni 82](#_Toc102654084)

[6. Predizione a 7 giorni con sequenze lunghe 1 giorno 83](#_Toc102654085)

[1.1.1 Considerazioni 84](#_Toc102654086)

[B. Tensione della cella minima, potenza del pannello, SOC e irradiazione 87](#_Toc102654087)

[1. Predizione ad 1 giorno con sequenze lunghe 3 giorni 87](#_Toc102654088)

[2. Predizione ad 1 giorno con sequenze lunghe 1 giorno 88](#_Toc102654089)

[3. Predizione a 3 giorni con sequenze lunghe 3 giorni 89](#_Toc102654090)

[4. Predizione a 3 giorni con sequenze lunghe 1 giorno 89](#_Toc102654091)

[5. Predizione a 7 giorni con sequenze lunghe 3 giorni 90](#_Toc102654092)

[6. Predizione a 7 giorni con sequenze lunghe 1 giorno 91](#_Toc102654093)

[7. Considerazioni 91](#_Toc102654094)

[C. Tensione della cella minima, potenza del pannello, SOC, irradiazione e corrente della batteria 94](#_Toc102654095)

[1. Predizione ad 1 giorno con sequenze lunghe 3 giorni 94](#_Toc102654096)

[2. Predizione ad 1 giorno con sequenze lunghe 1 giorno 95](#_Toc102654097)

[3. Predizione a 3 giorni con sequenze lunghe 3 giorni 96](#_Toc102654098)

[4. Predizione a 3 giorni con sequenze lunghe 1 giorno 96](#_Toc102654099)

[5. Predizione a 7 giorni con sequenze lunghe 3 giorni 97](#_Toc102654100)

[6. Predizione a 7 giorni con sequenze lunghe 1 giorno 98](#_Toc102654101)

[7. Considerazioni 99](#_Toc102654102)

[VIII. Conclusione 103](#_Toc102654103)

[IX. Appendice 103](#_Toc102654104)

[X. Indice delle figure 103](#_Toc102654105)

[XI. Bibliografia 108](#_Toc102654106)

# Introduzione

Negli ultimi anni, nel quadro dell’efficientamento dei processi industriali, l'interesse per la stima delle condizioni delle apparecchiature è fortemente aumentato. Questo, nell’ottica di evitare guasti/imprevisti la cui risoluzione porterebbe a forti perdite economiche (arresto della catena di produzione per individuare la causa del guasto e infine la rispettiva riparazione). Dato che, ogni apparecchiatura è in sostanza un sistema costituito da componenti che funzionano insieme per eseguire una o più attività, è facile capire come la sua affidabilità dipenda dalle condizioni di lavoro dei componenti. Nel caso in cui un componente si rompa, potrebbe essere necessario arrestare tutto il sistema per effettuare la manutenzione/riparazione e, nel peggiore dei casi, l'arresto improvviso di un sistema potrebbe causare danni gravi a cose e/o persone. Per questo motivo la manutenzione dei componenti è un aspetto essenziale per il corretto funzionamento di un sistema.

A questo proposito, l’avvento del internet of things ha favorito la raccolta di grandi moli di dati facilitando il monitoraggio di sistemi complessi e permettendo di visualizzarli da remoto. Tipicamente, l’andamento dei dati viene rappresentato tramite interfacce grafiche come gli SCADA, che permettono ad un utente specializzato di avere una panoramica sull’andamento delle variabili critiche di un sistema. In questo modo, qualora il sistema subisca un guasto, è possibile risalire alla causa di quest’ultimo ed effettuare la manutenzione. Tuttavia, questo tipo di monitoraggio presenta forti limiti, dato che non è in grado di riconoscere e segnalare all’utente una potenziale situazione di guasto in anticipo, ma solo individuarlo una volta accaduto. Nelle Smart Industries, la manutenzione predittiva è una delle tecniche più utilizzate per prevedere i guasti, dal momento che permette di valutare le condizioni di una specifica attrezzatura, con l'obiettivo di prevederne una possibile rottura prima che accada. Utilizzando questo tipo di tecnica, è possibile fare delle manutenzioni non ordinarie mirate per prevenire dei guasti e quindi ottimizzare il processo in questione.

Per predire lo stato di salute di un sistema esistono tecniche che fanno uso di sensori montati direttamente sui componenti, e che comparano le misurazioni dei sensori con i valori di lavoro standard scritti nelle schede tecniche. In particolare, grazie alla crescita dell’affidabilità dei sistemi di intelligenza artificiale, è possibile creare dei modelli che sappiano prevedere automaticamente se la situazione presente possa portare ad un guasto futuro oppure se sia una situazione di lavoro normale. Ciò è possibile grazie all’estrazione automatica di correlazioni tra il valore delle grandezze e gli eventi di guasto.

Nella fattispecie, l’obbiettivo di questa tesi è di usare il machine learning per proporre un sistema in grado di prevenire un guasto. Il modello viene allenato a distinguere le situazioni critiche da quelle normali, usando sequenze temporali di dati storici appositamente etichettate come sane o patologiche. Una volta allenato, il modello sarà in grado di generalizzare le conoscenze acquisite per classificare in modo automatico sequenze sconosciute provenienti dal campo e segnalare in anticipo situazioni di potenziale spegnimento. L’obbiettivo è di rilevare un evento critico prima che accada ed avere il tempo necessario per intervenire prima che si verifichi.

Nel nostro caso, si tratta di apparecchiature di monitoraggio posizionate in zone più o meno remote, monitorate tramite SCADA, alimentate da pannelli solari e dotate di autonomia grazie ad un pacco batterie. È di grande importanza che le apparecchiature rimangano continuamente accese ed è per questo che vogliamo sviluppare un algoritmo che, in base allo stato dell’apparato, predica gli spegnimenti dovuti ad una scarsa irradiazione del pannello. Infatti, qualora un sistema si trovi in situazioni di potenziale guasto e venga segnalato all’utente, egli può operare da remoto per diminuire il consumo dell’apparato ed aumentare la sua autonomia, ma anche programmare un intervento di manutenzione sul campo in tempi tali che il sistema non si spenga, o che lo spegnimento del sistema non sia irreversibile. Quest’ultimo caso, si verificherebbe qualora il sistema rimanga spento per troppo tempo, e quindi la tensione della batteria scenda sotto una soglia critica oltre la quale diventi impossibile ricaricarla.

Data la complessità e la varietà dei dati useremo il deep learning. Infine, analizzeremo i risultati ottenuti variando il periodo di predizione, la lunghezza in giorni delle sequenze e la combinazione di grandezze date in ingresso alla rete neurale.

La tesi contiene una prima parte in cui facciamo un’introduzione teorica al machine learning e in particolare al deep learning, una seconda parte in cui riassumiamo lo stato dell’arte della manutenzione predittiva nel quadro dell’industria contemporanea, una terza parte in cui presentiamo il caso di studio, ed infine due sezioni in cui presentiamo l’algoritmo implementato e i risultati ottenuti.

# Deep Learning

## Recursive Neural Network - RNN

Il Deep Learning è una branchia del Machine Learning che s’inspira alla struttura del cervello umano, in particolare al neurone, per creare degli algoritmi di apprendimento. Il neurone è una cellula che raccoglie, conduce gli impulsi nervosi e si divide in neuroni sensori, motori e intermedi. Ce ne sono più di 100.000 nel nostro sistema nervoso e sono fondamentali per ricevere e trasmettere segnali. Una rete neurale cerca di riprodurre il funzionamento del neurone umano, ovvero tutti quei processi che avvengono nel cervello durante la fase di apprendimento e quella successiva del riconoscimento. Come nel cervello, questi algoritmi hanno una struttura a strati gerarchica (layers), in questo modo viene scomposta la complessità del problema. Gli strati più bassi permettono a quelli più alti l’apprendimento di concetti man mano sempre più astratti. Se immaginiamo che una rete neurale controlli il comportamento di un semplice organismo posto nell'ambiente, le unità di input della rete emulano gli organi sensoriali dell'organismo e le unità di output gli organi motori. L'insieme dei livelli di attivazione delle unità di input codifica lo stato dei diversi tipi di agenti fisici e chimici (luminosi, sonori, meccanici, termici, odori, sapori) presenti in un dato momento nell'ambiente circostante e a cui l'organismo è sensibile. L'attivazione si propaga dalle unità di input alle unità interne e da queste alle unità di output. L'insieme dei livelli di attivazione delle unità di output codifica i movimenti con cui l'organismo risponde all'informazione sensoriale proveniente dall'esterno. Ogni rete ha un'architettura costituita dal numero di unità e dallo schema delle interconnessioni.

In Figura II‑1 vediamo la struttura di un neurone artificiale (12):

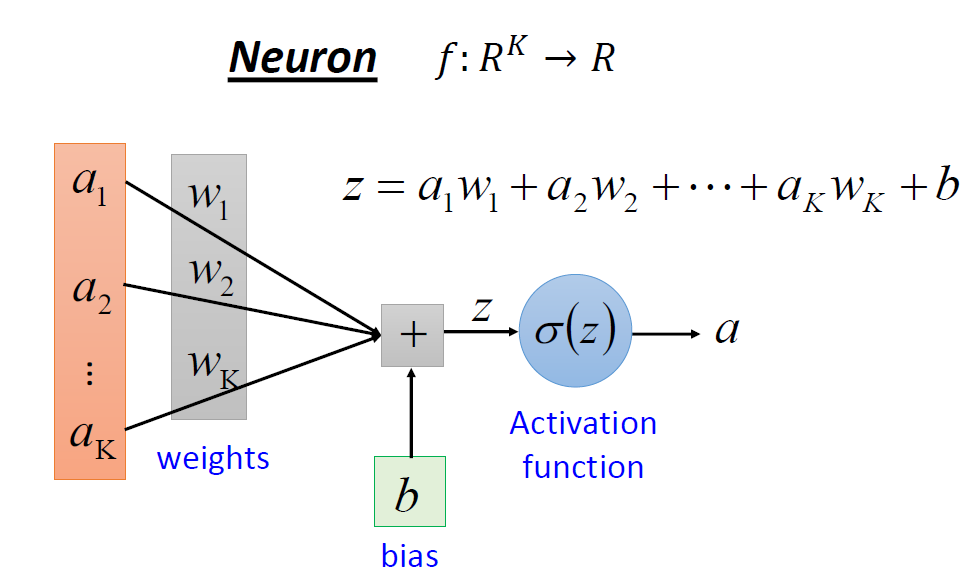


Figura II‑1: Neurone artificiale

Vediamo che l’uscita del neurone è la combinazione lineare degli ingressi moltiplicati per dei coefficienti che vengono chiamati pesi. A questo termine va sommato un termine di bias. La funzione di attivazione che si trova dopo l’uscita ha il compito di pesare l’output del neurone prima che venga passato ad un altro elemento della rete.

Fra i vari tipi di architetture, distinguiamo le architetture feedforward (a propagazione in avanti) e le architetture ricorrenti. Un'architettura feedforward è formata da un insieme di unità di input (input layer), da uno strato di unità di output (output layer) e da uno o più strati intermedi di unità interne o nascoste (hidden layers). Ogni unità di ciascuno strato è connessa con tutte le unità dello strato successivo, a partire dallo strato di input fino a quello di output. In ogni ciclo, l'attivazione si propaga in avanti dalle unità di input ai successivi strati di unità interne fino a raggiungere le unità di output (11).

In Figura II‑2 vediamo lo schema di una rete feedforward con 2 layers nascosti (12):

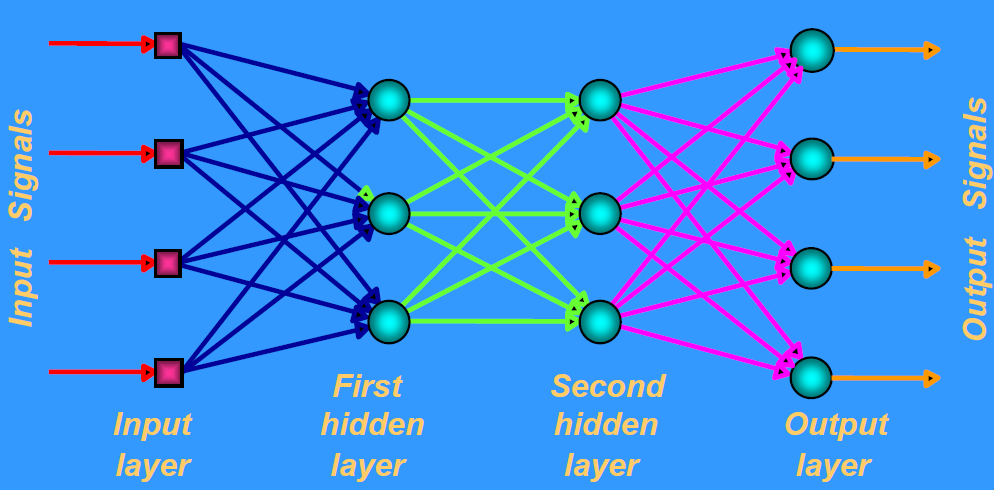


Figura II‑2: Rete feedforward

1. A differenza delle architetture feedforward, le architetture ricorrenti presentano connessioni che partono da una data unità e ritornano, direttamente o indirettamente, alla stessa unità. Un tipo di architettura ricorrente può servire a dotare una rete neurale di una memoria che conserva una traccia di quanto è avvenuto nei cicli precedenti. In ogni ciclo input-output, il pattern di attivazione delle unità interne viene memorizzato in uno speciale insieme di unità chiamate [unità di memoria](https://www.treccani.it/enciclopedia/unita-di-memoria_(Enciclopedia_della_Scienza_e_della_Tecnica)/), collegate alle unità interne tramite normali connessioni. Pertanto, in ogni ciclo, il livello di attivazione delle unità interne, e quindi anche l'output della rete, viene a dipendere non solo dall’input in quell’istante, ma anche dalla traccia dello stato della rete nel ciclo precedente, conservata nelle unità di memoria (11).

In Figura II‑3 vediamo lo schema di una rete ricorrente con 1 layer nascosto (12):

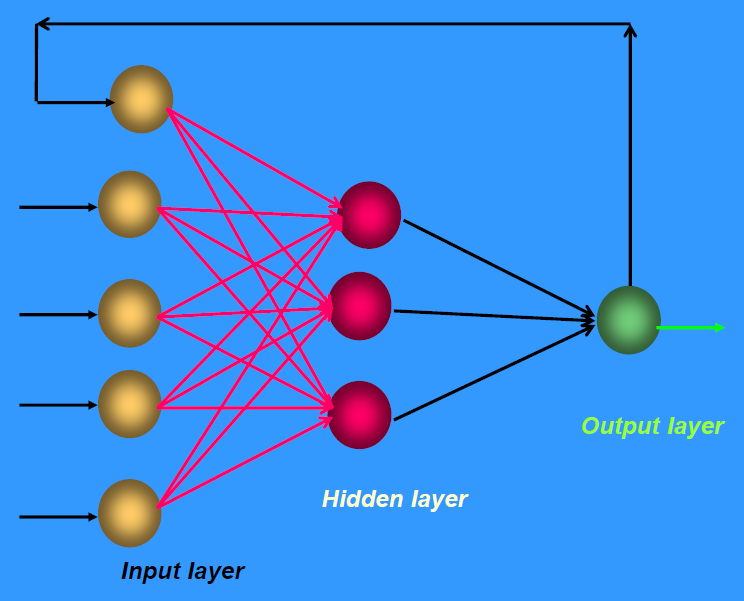


Figura II‑3: Recurrent Neural Network

In Figura II‑3 la freccia che parte dallo strato di uscita per tornare allo strato di ingresso rappresenta il feedback, ossia l’informazione che permette di contestualizzare i nuovi dati in ingresso rispetto agli istanti precedenti.

Per creare una rete ben dimensionata è necessario fare attenzione alla complessità della rete portata dal numero di neuroni in ogni strato. In particolare, il numero di neuroni corrisponde alla quantità di informazioni memorizzate ad ogni step. Aumentandone il numero aumenta la complessità della rete in termini di numeri di parametri. Creare una rete neurale complessa può portare la rete ad avere risultati eccellenti, in termini di accuratezza, sul set di dati su cui è stata allenata e di avere risultati pessimi su un set di dati sconosciuti. Questo fenomeno si chiama overfitting, ed è causato dal fatto che durante l’allenamento la rete ha un numero di parametri tale da permettergli da imparare “a memoria” la relazione tra i dati del training set e le sue etichette di riferimento, senza però sviluppare la capacità di generalizzare le conoscenze acquisite a discapito dell’accuratezza su dati sconosciuti (12).

In Figura II‑4 vediamo uno schema dettagliato di una rete RNN srotolata (13):

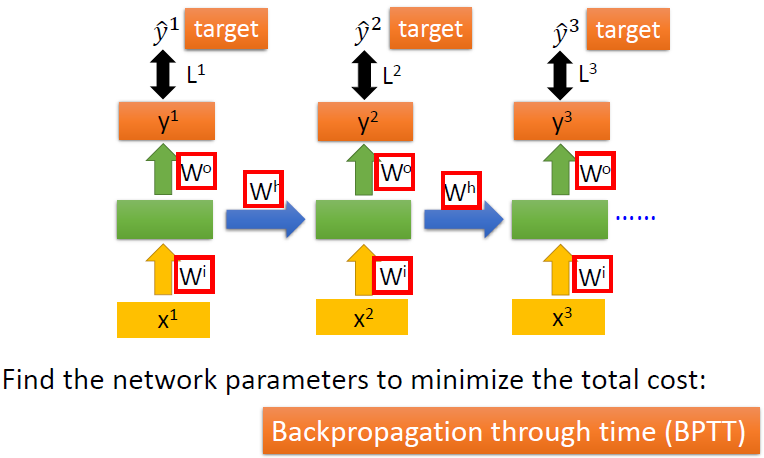


Figura II‑4: Schema RNN srotolato

In Figura II‑4, vediamo la stessa cella a diversi istanti di tempo. La rete è rappresentata in modo srotolato, ossia per ogni istante di tempo si rappresenta una cella ma è sempre la stessa. Con intendiamo gli ingressi, con gli output stimati dalla rete e con i pesi/parametri per l’input, l’output e lo stato. I target rappresentano l’etichetta di riferimento del campione e il discostamento in modulo tra l’output della rete e il riferimento. Vediamo che ad ogni istante di tempo la rete ha due ingressi: . La memoria della rete è rappresentata dallo stato al tempo , ciò implica che dipende da tutti gli ingressi compresi tra l’istante zero e l’istante i.

In queste reti ricoprono un ruolo di fondamentale importanza i suddetti parametri/pesi. Per ottimizzarli, essi vengono aggiornati ad ogni epoca, ossia ogni volta che tutti i dati contenuti nel training sono stati analizzati dalla rete. Il feedback consiste nel comparare il valore di riferimento della sequenza () con il valore predetto dalla rete () e fare le correzioni ai parametri di conseguenza. Più specificatamente, ad ogni epoca ogni parametro del modello riceve un aggiornamento proporzionale alla [derivata parziale](https://it.wikipedia.org/wiki/Derivata_parziale) della funzione di costo rispetto al parametro stesso (13)(14). L’obbiettivo che si vuole raggiungere è quello di è trovare il minimo globale della funzione di costo. Il costo può essere una distanza Euclidea oppure la cross entropia dell‘uscita della rete rispetto al riferimento (13). Per minimizzare la funzione di costo è molto importante che tutti i parametri vengano ottimizzati tramite il meccanismo di retro-propagazione del gradiente.

A tal proposito, uno dei principali colli di bottiglia delle reti RNN riguarda il meccanismo di retro-propagazione del gradiente, ed è il fenomeno della scomparsa del gradiente (14). Questo fenomeno si presenta quando si usano funzioni di attivazione non lineari classiche (la [tangente iperbolica](https://it.wikipedia.org/wiki/Tangente_iperbolica) o la f[unzione logistica](https://it.wikipedia.org/wiki/Funzione_logistica)) che hanno gradiente a valori nell'intervallo [-1,1] o [0,1], e si hanno dei dati costituiti da sequenze temporali di più di 1000 campioni. Se si verificano queste due condizioni, nell'algoritmo di retro-propagazione, quando i gradienti ai vari strati vengono moltiplicati tramite la [regola della catena](https://it.wikipedia.org/wiki/Regola_della_catena), dopo un certo numero di strati, il gradiente si approssima a zero. Questo è dovuto al fatto che, il prodotto di n numeri in [0,1] decresce esponenzialmente rispetto ad n (la profondità della rete). Questo fenomeno porta all’annullamento dell’effetto di correzione dei pesi negli strati più vicini all’inizio della sequenza e fa sì che la rete non riesca ad imparare le correlazioni tra due istanti distanti di tempo distanti. Quando invece il gradiente delle funzioni di attivazione assume valori elevati, un problema analogo che può manifestarsi è quello dell'esplosione del gradiente (14). Il problema della scomparsa del gradiente limita fortemente la capacità di apprendimento della rete. Questo problema può essere risolto parzialmente tramite l’uso di funzioni di attivazioni lineari come la ReLu (14).

Un’importante innovazione per le reti RNN fu introdotta nel 1997 da Mike Schuster e Kuldip K. Paliwal che presentarono per la prima volta il concetto di RNN bidirezionale (15). Il concetto di queste reti è di avere un ingresso che arriva dall’istante successivo, al tempo (t+1), e uno dall’istante precedente, al tempo (t-1), oltre che all’input al tempo t.

La figura esplicita il concetto (13):

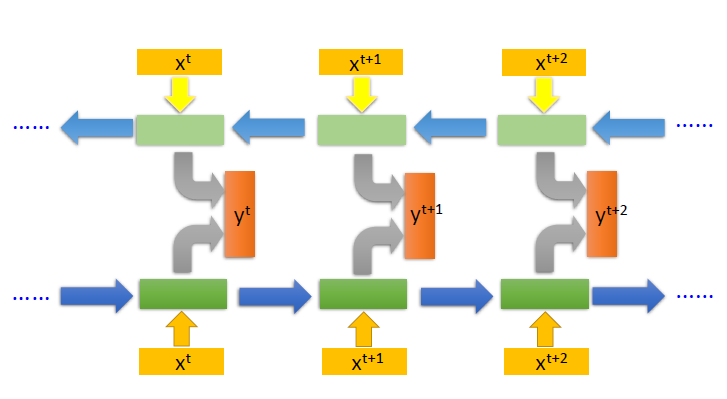


Figura II‑5: Rete RNN bidirezionale

In Figura II‑5 abbiamo due reti RNN sovrapposte che prendono in ingresso gli stessi dati ma nell’ordine opposto. Ciò permette di classificare l’elemento all’istante t grazie ai campioni precedenti ma anche a quelli futuri. Le sequenze vengono lette dal primo all’ultimo e contemporaneamente dall’ultimo al primo elemento. Per poter avere una rete bidirezionale bisogna avere accesso a tutta la sequenza al momento della predizione.

Come vediamo nel seguente esempio, una rete bidirezionale può essere utile nel caso in cui si debba prevedere le parole in una frase (18):

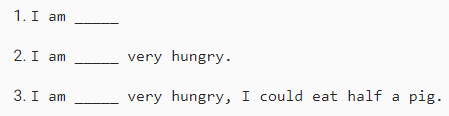


Figura II‑6: Predizione delle parole in una frase in base al resto della frase

In Figura II‑6, le soluzioni sono “hungry”, “not” e “very”. Per saperlo, nella prima riga ci bastano le informazioni precedenti, mentre nei due casi seguenti diventa fondamentale il resto della frase per poter contestualizzare.

## Long Short Term Memory - LSTM

Per risolvere il problema della scomparsa del gradiente, nel 1997 [Sepp Hochreiter](https://it.wikipedia.org/w/index.php?title=Sepp_Hochreiter&action=edit&redlink=1) e [Jürgen Schmidhuber](https://it.wikipedia.org/wiki/J%C3%BCrgen_Schmidhuber) creano le reti LSTM (Long-Short Term Memory) (16).

In Figura II‑7 vediamo uno schema esplicativo di una rete LSTM (13):

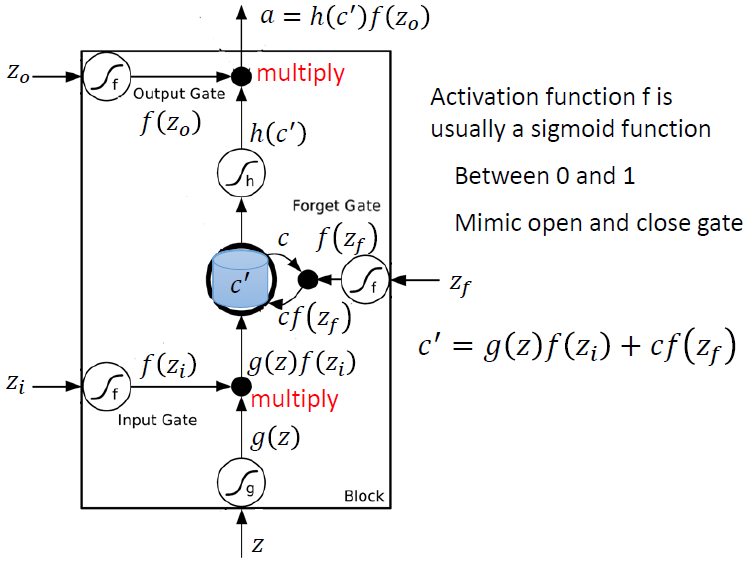
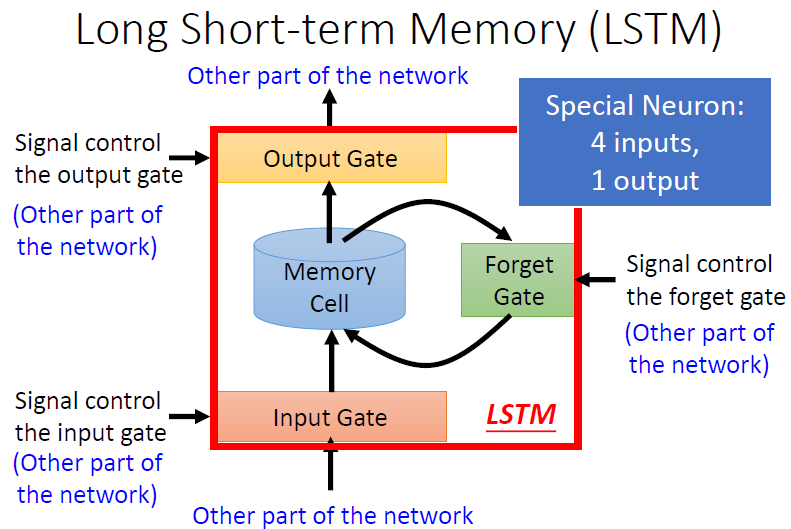


Figura II‑7: Rete LSTM

In Figura II‑7 vediamo lo schema di un singolo neurone, anche chiamato cella. Come nel caso delle reti RNN, le celle LSTM sono dotate di uno stato interno che li consente di tenere memoria dei campioni precedenti, e che verrà dato in ingresso al neurone all’istante di tempo successivo. Ogni neurone ha 4 ingressi ed 1 uscita. Inoltre, vediamo la presenza di vari gate. Ad ogni gate è associata una rete neurale, allenata durante il training, che permette di controllare quale informazione viene salvata nello stato.

In Figura II‑8 andiamo a vedere più nel dettaglio a cosa servono i gate e come funzionano (17):

Immagine che contiene testo, orologio

Descrizione generata automaticamente

Figura II‑8: Cella LSTM

Come possiamo vedere in Figura II‑8, la cella prende in ingresso l’uscita della cella al tempo t-1 (), l’input al tempo t () e lo stato della cella al tempo t-1 (), e ritorna in uscita lo stato al tempo t () e l’uscita al tempo t (). Inoltre, la cella fa uso di funzioni di attivazione non lineari come la sigmoide e la tangente iperbolica. E da notare, come i vettori e vengono concatenati prima di entrare nei vari gate.

Vediamo le funzionalità dei gate della cella LSTM 109(17):

* Il *Forget Gate* prende in ingresso l’input al tempo t e l’output della cella al tempo (t-1). Grazie ad una funzione di attivazione sigmoide nell’intervallo [0,1] stabilisce quali informazione salvare o meno nello stato della cella. In particolare, tramite una moltiplicazione con la sigmoide, l’informazione viene cancellata (sigmoide uguale a 0), salvata (sigmoide uguale a 1), oppure attenuata (sigmoide con valore compreso tra 0 e 1). Ciò permette alla rete di selezionare quali sono le informazioni rilevanti e quali invece vanno scartate, cosa che non accade nelle RNN. Il coefficiente risultante va moltiplicato con lo stato della cella al tempo (t-1).
* L’*Input Gate* prende in ingresso l’input al tempo t e l’output della cella al tempo (t-1). Essi passano in due rami diversi, uno con una sigmoide e uno con una funzione di attivazione tangente iperbolica. I risultati di questi due rami vengono moltiplicati tra loro. La tangente iperbolica normalizza i valori in [-1,1] così da non subire l’esplosione di valori che durante il processo iterativo potrebbero essere incrementati/moltiplicati diverse volte andando a sregolare il processo di apprendimento. La sigmoide seleziona i valori normalizzati. L’output si somma allo stato della cella al tempo (t-1) per creare lo stato corrente.
* L’*Output Gate* prende in ingresso l’input al tempo t e l’output della cella al tempo (t-1) e li passa attraverso una sigmoide. La risultante viene moltiplicata per lo stato della cella normalizzato da una tangente iperbolica e forma l’output della cella*.*

Le equazioni che legano i rami tra loro sono le seguenti (17):

Vediamo che l’equazione dello stato corrente ( della cella dipende dai parametri del Forget Gate (, dell’Input Gate (, e dello stato precedente (. L’output corrente della cella è il prodotto dell’output gate con la tangente iperbolica dello stato corrente.

Le equazioni dei vari gate sono le seguenti (17):

Dove e rappresentano rispettivamente il peso e il bias per ogni gate. La rappresenta la funzione di attivazione sigmoide.

I gate sono un’innovazione significativa, che permette alla rete LSTM di ricordare solo le informazioni rilevanti contenute all’interno di una sequenza, e quindi permette alla rete di apprendere correlazioni tra istanti di tempo lontani tra loro.

Come per le RNN anche le reti LSTM possono essere bidirezionali ed apportano lo stesso tipo di vantaggio che le RNN. Nel caso delle reti BiLSTM avremmo due layers LSTM consecutivi e quindi il numero di parametri ottenuti dalla rete sarà doppio rispetto al caso LSTM semplice.

# Manutenzione predittiva

## Che cos’è e perché si usa

Negli ultimi anni l’avvento dell’IoT ha permesso, grazie a sensori posizionati sempre più diffusamente su tutti i componenti di un sistema, di accedere a misurazioni e dati in grado di definire/descrivere in modo accurato tutte le fasi di un processo. Questa disponibilità di dati, campionati in modo discreto o continuo a seconda dei casi, sta portando una quantità tale di informazioni da influenzare fortemente il modo di prendere decisioni. L’informazione portata da questi dati, se sfruttata e decifrata correttamente, permette di avere un quadro migliore dello stato di salute dei componenti di un sistema e dunque di ottimizzarne la manutenzione e rendere più efficienti di processi.

Nella fattispecie, la norma UNE-EN 13306. 2018 definisce la manutenzione come una “Combinazione di tutte le azioni tecniche, amministrative e gestionali, durante il ciclo di vita di un’entità, destinate a mantenerla o riportarla in uno stato in cui possa eseguire la funzione richiesta”(19). A partire da questa definizione, possiamo differenziare i diversi tipi di manutenzione come in Figura III‑1 (1):

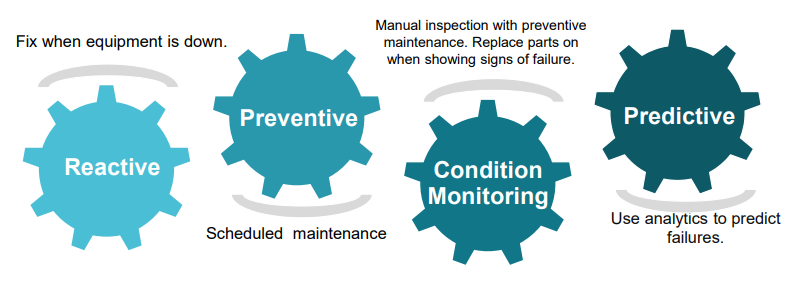


Figura III‑1: Diversi tipi di manutenzione

In Figura III‑1 vediamo i 4 tipi di manutenzione: reattiva, preventiva, via monitoraggio dei componenti e predittiva. Le prime tre tecniche di manutenzione sono convenzionali, poiché non vengono effettuate in modo automatico e consistono nella manutenzione di un componente, dopo che si sia rotto, prima che si rompa anche se non dà segni di malfunzionamento (manutenzione programmata), oppure consiste nel monitoraggio da parte di un utente delle condizioni del componente per rilevare comportamenti anomali e sostituirlo/ripararlo prima che si rompa. Questo tipo di approccio è caratterizzato da perdite economiche dovute alla sostituzione di componenti ancora funzionanti, da periodi di fermo degli apparati mediamente lunghi, e nei casi di manutenzione reattiva può portare anche a danni a persone e/o cose. Per evitare questo tipo di imprevisti è stato introdotto l’uso di strumenti analitici automatici per la manutenzione predittiva in grado di sfruttare la grande mole di dati altrimenti difficilmente interpretabile da un utente. Inoltre, l’utilizzo di sistemi automatizzati permette di eliminare l’incertezza e l’errore portato dal lavoro umano poiché implementa delle procedure sistematiche.

A seguire uno schema della processazione dei dati per effettuare manutenzione predittiva:

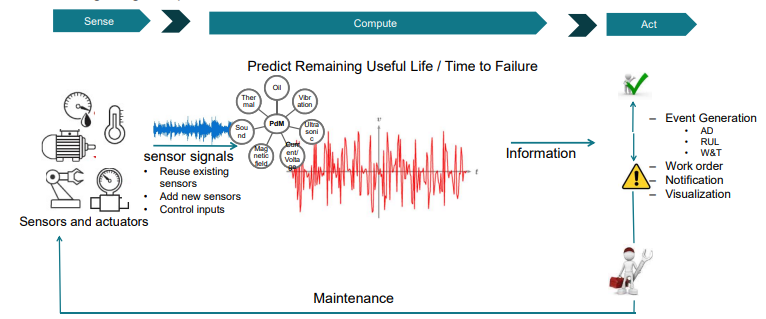


Figura III‑2: Schema di manutenzione predittiva

Come vediamo in Figura III‑2, i dati vengono raccolti dal campo, passati all’algoritmo di predizione ed elaborati. Una volta estratte le informazioni necessarie dai dati, delle notifiche vengono mandate all’utente per informarlo sullo stato di salute dei componenti.

È stato dimostrato che la manutenzione facente uso del machine learning (ML) fornisce sempre più soluzioni efficaci grazie alla crescita delle funzionalità hardware dei sensori, alla potenza di calcolo accessibile via cloud e all’introduzione di nuovi algoritmi all'avanguardia (3). Per questo motivo il ML sta diventando essenziale per una gestione efficiente delle attività di manutenzione e per ridurre i costi associati ai tempi di fermo e ai prodotti difettosi.

Tra i vari approcci alla manutenzione predittiva distinguiamo tre categorie: La manutenzione predittiva basata su un modello fisico del sistema, quella basata sui dati (data-driven) e quella ibrida. Il primo si basa sulla conoscenza del sistema per costruire una modello matematico/fisico dell’andamento nel tempo della degradazione del componente. In questo caso è facile comprendere il significato fisico dei risultati forniti, ma è difficile implementare un modello accurato in sistemi complessi perché diventa molto difficile conoscere e modellizzare tutte le correlazioni tra le variabili. L’approccio data-driven, si basa sull’analisi di dati storici descriventi il funzionamento del sistema. Confrontando questi dati con i dati ricevuti si può predire il futuro stato del sistema, senza però averne un modello fisico. Questo tipo di approccio si basa su metodi statistici, funzioni di affidabilità e metodi di intelligenza artificiale. Un vantaggio dei sistemi data-driven è che sono più abbordabili per sistemi complessi poiché permettono di ottenere risultati utili anche se non si ha una completa conoscenza della complessità del sistema. Tuttavia, per questo stesso motivo, le correlazioni tra i risultati ottenuti ed il problema sono più difficili da interpretare da un punto di vista fisico. La maggiore accessibilità ai dati, favorita dalla crescita tecnologica dell’IoT, sta permettendo ai sistemi data-driven di guadagnare popolarità dato che permettono di avere accesso a quella grande mole di dati che richiedono per essere affidabili. In particolare, la branchia facente uso di reti di deep learning sta emergendo poiché riesce ad estrarre correlazioni complesse dai dati. Infine, l'approccio ibrido combina i due approcci sopra citati.

L’introduzione di tali sistemi implica l’impiego di risorse e una serie di costi (2). In Figura III‑3 ne possiamo vedere un riassunto:

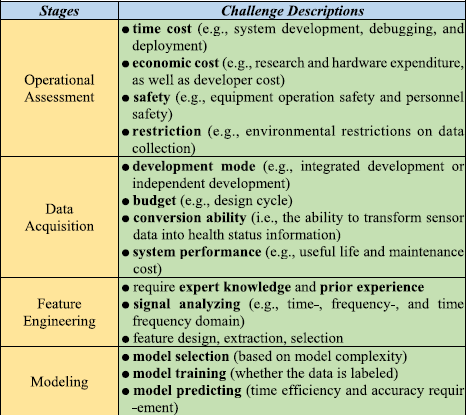


Figura III‑3: Costi dell'implementazione di un sistema di manutenzione predittiva

In particolare, vediamo come la transizione da una manutenzione tradizionale ad un sistema di manutenzione predittiva porta un impiego di risorse economiche, ma anche di tempo e di persone. Le spese economiche vanno dal design del progetto, all’acquisto dell’hardware (sensori, microcontrollori, etc..), all’acquisizione e l’elaborazione dei dati, fino allo sviluppo del modello e al test. Dato che l’approccio è data-driven l’acquisizione del dato è cruciale per il corretto funzionamento del sistema. Il trade-off tra i costi dello sviluppo del sistema di manutenzione predittiva e le perdite economiche dovute a guasti dei componenti è solitamente positivo e porta un valore aggiunto al processo. In particolare, esso permette di avere una panoramica più chiara dello stato di salute del sistema.

Per giustificare maggiormente l’investimento che rappresenta una sistema di manutenzione predittiva, forniamo a seguire qualche esempio di casi gravi che si sarebbero potuti prevenire, ed in cui il guasto di uno o più componenti abbia portato a gravi conseguenze. Nel giugno del 2009, una linea della metropolitana di Washington DC si è schiantata uccidendo 9 persone e ferendo dozzine di persone. Il guasto è stato attribuito ad anomalie della sensoristica posta sotto i binari della metropolitana (5). In Brasile, nel novembre del 2009, diversi blackout hanno causato disagi a più di 60 milioni di persone e arrestato diversi servizi, tra cui la metropolitana e l’illuminazione pubblica. Nonostante la causa di questi imprevisti sia stata attribuita a fattori metereologici come fulmini, vento e pioggia, è chiaro che è stato complice anche un guasto tecnico (6). Un ultimo caso si riscontra nel famoso malfunzionamento delle automobili Toyota che subivano accelerazioni involontarie e repentine causando, problemi per la sicurezza degli utenti, diversi morti, ed il ritiro di circa 10 milioni di automobili dal mercato da parte dell’azienda, senza contare i danni d’immagine (7).

## Stato dell’arte

Andiamo ora a presentare lo stato dell’arte della manutenzione predittiva nel caso di approcci data-driven facenti uso del machine learning. Come detto, l’approccio data-driven permette di analizzare dei sistemi senza averne un modello fisico e per questo motivo sta trovando sbocchi in diversi settori.

Nel settore dell’agricoltura, la manutenzione predittiva viene usata per monitorare lo stato di salute di una piantagione. In questo modo si vuole predire ed evitare una possibile infezione della piantagione che potrebbe portare alla perdita del raccolto. Tramite sensori di temperatura sparsi nella piantagione, lo stato di salute viene monitorato di modo da predire 8 giorni prima del suo avvenimento, un’infezione della coltura. Facendo una classificazione è possibile risalire all’entità del problema che si sta prevedendo tramite dei livelli di gravità e quindi è possibile decidere con precisione all’operazione da fare. I risultati ottenuti in termini di accuratezza sono del 99,97% (20).

Nel settore ferroviario troviamo un caso di studio riguardante la detezione di potenziali difetti/anomalie dei binari dei treni che possano portare a guasti. Il caso è stato ampiamente studiato dato che i modelli fisici dei binari non bastavano per prevedere la natura stocastica di alcune anomalie che non potevano essere evitate grazie a manutenzioni regolari. Un modo per risolvere questo quesito è di monitorare continuamente le condizioni dei binari. In questo caso la rete usata è una rete RNN. I risultati ottenuti riguardano la predizione del numero e del tipo di difetti presenti sul binario rispetto all’ultima manutenzione. I risultati mostrano un’accuratezza del 82% (21).

Un caso di particolare interesse, potrebbe essere quello della manutenzione predittiva applicata a motori industriali (22). L’applicazione in questione fa uso della grande mole di dati fornita dai sensori elettrochimici per allenare una rete di deep learning a classificare le sequenze come patologiche o sane. Il dataset fornisce una serie di misurazioni relative a varie turboventole di un motore per aerei (temperatura, pressione, velocità della turboventola, etc…) in diverse condizioni ambientali, al fine di monitorarne le condizioni di degradazione. L’obbiettivo è quello classificare le sequenze in ingresso per predire un guasto con largo anticipo. I risultati ottenuti variano in base all’anticipo con cui si vuole prevedere il guasto. L’intervallo di tempo in questo caso viene specificato in numero di cicli del motore rimanenti prima del guasto. Usando una rete LSTM e posizionandoci 50 cicli prima del guasto, otteniamo un’accuratezza del 91%, mentre a 5 cicli dal guasto l’accuratezza è del 100% (23). Altri tipi di approccio per predire i guasti sono quelli di creare una rete in grado di calcolare il tempo di vita utile rimanente del componente, anche detto RUL (Remaining Useful Life), oppure di creare una rete in grado di prevedere i futuri valori delle grandezze in base ai dati storici (22).

Un’altra applicazione industriale che troviamo rilevante, viene proposta per lo studio dello stato di funzionamento delle macchine utensili a controllo numerico computerizzato (CNC). Esse svolgono un ruolo importante nel settore moderno manufatturiero. Queste macchine sono utilizzate per la sagomatura o la lavorazione di materiali rigidi come il metallo, di solito tagliando, alesando, rettificando, o deformando in altre forme. A causa del prolungato funzionamento, sul lungo termine le abrasioni e lo stress dei suoi componenti meccanici sono inevitabili, con conseguente scarsa qualità del prodotto e bassa efficienza. Nell'industria moderna, quasi il 79,6% dei tempi di inattività di una macchina utensile è causata da guasti meccanici. Per ridurre al minimo il tempo di fermo degli apparati si misurano e analizzano le vibrazioni dei componenti riuscendo ad avere discreti risultati. Il problema che si riscontra all’atto pratico è che le vibrazioni corrispondenti ad un cambio di modalità o condizioni di lavoro del macchinario spesso possono essere confuse con quelle dovute a guasti. A questo proposito, in questo articolo si allena un algoritmo di deep learning a distinguere questi due tipi di vibrazioni in modo automatico. Anche qui i risultati ottenuti dalla classificazione sono soddisfacenti e pari ad un’accuratezza del 98,1%.

# Architettura del sistema

## Acquisizione e trasmissione dei dati

Gli apparati di campo sono costituiti da una serie di sensori, una centralina meteo e un pacchetto di telecomunicazione, il tutto alimentato da batterie ricaricate tramite pannelli solari. Gli apparati si distinguono in tre diverse tipologie (A, B e C). Le prime due sono costituite da un pannello solare che alimenta un pacco batterie, un BMS, sensori e stazione meteo, mentre la terza non possiede né i sensori né la stazione meteo. Per contro i tipi B e C hanno pannelli fotovoltaici e batterie maggiori del tipo A.

A seguire vediamo uno schema a blocchi di un dispositivo di tipo A:

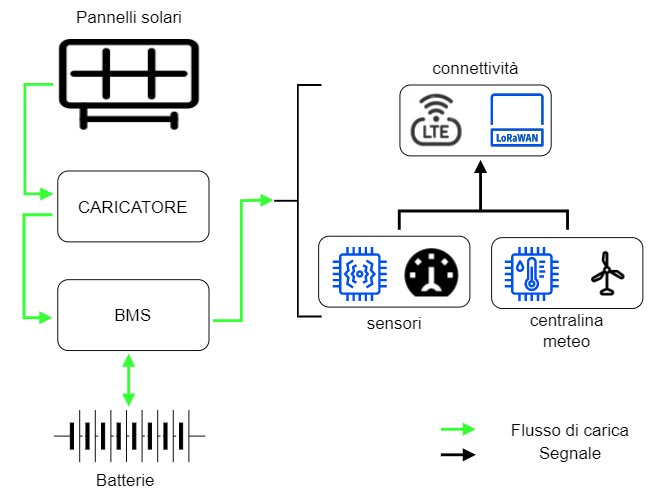


Figura IV‑1: Schema a blocchi di un dispositivo di tipo A

Vediamo che il dispositivo è costituito da un pannello solare collegato ad un caricatore. Tra il caricatore e il pacco batterie abbiamo un BMS (Battery Management System) incaricato di ottimizzare la gestione e il tempo di vita delle celle. Come si evince in Tabella 1, i dispositivi di tipo A sono dotati di un pacco batterie contenente 4 celle a litio. Il BMS è il sistema che ne gestisce la carica e la scarica in modo da garantirne il corretto funzionamento e la miglior durata di vita. Inoltre, calcola lo stato di carico (SOC), ossia la percentuale della piena capacità di carica del pacco batterie ancora disponibile (10). E il BMS a decidere se e quando caricare o scaricare la batteria. Nel nostro caso, la decisione viene presa prendendo in considerazione la potenza fornita dal pannello solare e il SOC della batteria: Se il SOC non è al 100% e il pannello sta fornendo una potenza maggiore rispetto al consumo dal sistema, il BMS passa direttamente al sistema la carica necessaria e stocca il surplus nel pacco batterie. Nel caso in cui il pannello non stia producendo abbastanza potenza per alimentare da solo il sistema, la carica necessaria viene prelevata dal pacco batterie. Inoltre, qualora il pannello stia producendo carica in eccesso ma il SOC stia già al 100%, il pacco batterie non viene ricaricato dal BMS.

Come si evince dalla Figura IV‑1, il consumo del dispositivo è dovuto ai sensori, la centralina meteo e alla connettività. Le considerazioni fatte sui dispositivi di tipo A si applicano ai dispositivi di tipo B, e anche a quelli di tipo C se non si considerano i sensori e la centralina meteo.

A seguire vediamo una foto di un dispositivo di tipo B con tutti i suoi componenti:

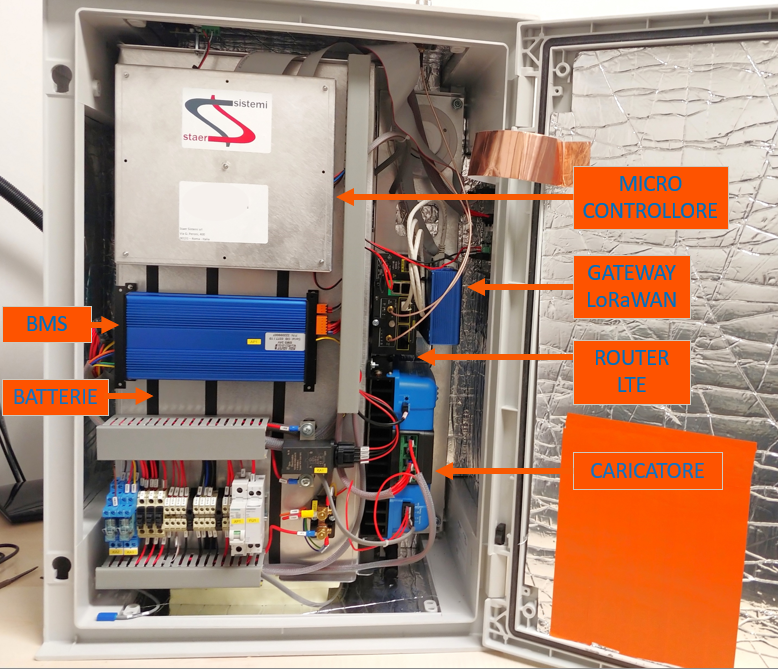


Figura IV‑2: Implementazione hardware di un dispositivo di tipo B

In Figura IV‑2 vediamo un’implementazione reale dello schema a blocchi presentato in Figura IV‑1. Dato che i dispositivi saranno collocati in ambienti difficilmente raggiungibili e che saranno esposti alle intemperie, vengono posizionati in un armadietto protettivo.

A seguire vediamo una tabella ricapitolativa delle caratteristiche dei dispositivi:

Tabella 1: Caratteristiche hardware degli apparati

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | A | B | C |
| BATTERIE  (Litio) | Tensione nominale | 12.8 V | 25.6 V | 25.6 V |
| Carica elettrica | 100 Ah | 160 Ah | 160 Ah |
| Energia immagazzinabile | 1280 Wh | 4096 Wh | 4096 Wh |
| Celle per unità | 4 | 8 | 8 |
| PANNELLI SOLARI | Potenza nominale | 115 W | 400 W | 400 W |
| Dimensioni | 1015 x 668 x 30 mm | 1690 x 1046 x 40 mm | 1690 x 1046 x 40 mm |
| SENSORI | | SI | SI | NO |
| CENTRALINA METEO | | SI | SI | NO |
| PROTOCOLLI DI COMUNICAZIONE | | LORAWAN e LTE | LTE | LTE |

Dalla Tabella 1 possiamo osservare come i dispositivi A abbiano due protocolli di comunicazione. Il primo è il protocollo LoRaWAN (Long Range Wide Area Network), un protocollo di trasmissione dei dati ad ampio spettro e a lunga portata che si basa sulla tecnologia LoRa. Quest’ultima è una tecnica di modulazione che deriva dalla tecnologia Chirp Spread Spectrum, utilizzata fin dal secondo dopoguerra nel campo militare. L’architettura include standard di protocollo e funzionalità che supportano la comunicazione bidirezionale a basso costo, mobile e sicura ed ottima per dispositivi IoT. L’architettura LoRaWAN è ottimizzata per un basso consumo energetico ed è progettata per scalare da un’installazione di un singolo gateway, fino a grandi reti globali con miliardi di dispositivi (8). Il protocollo LoRaWAN è usato di default dai dispositivi A per ridurre i consumi poiché hanno pacchi batterie più piccoli rispetto agli altri dispositivi. La rete LTE (Long Term Evolution) viene usata come soluzione provvisoria qualora si dovesse verificare un problema temporaneo sulla rete LoRaWAN. Quest’ultimo è uno standard di connessione che permette l'accesso a internet a banda larga anche su reti mobili e ha prestazioni migliori di LoRaWAN a fronte di un consumo energetico maggiore (9).

I dispositivi B e C invece, fanno uso solamente della rete LTE poiché possiedono pacchi batterie e pannelli solari di dimensioni maggiori. In particolare, come vediamo in Tabella 1, le batterie e i pannelli solari montati sui dispositivi B e C sono identici. Notiamo inoltre, che i dispositivi C non possiedono né stazione meteo né sensori poiché assolvono solo il compito del ricevitore per i segnali LORAWAN dei dispositivi A nelle vicinanze.

Una considerazione importante è che tutti i dispositivi (A, B e C) hanno un consumo approssimativamente costante causato sia dalla trasmissione dei dati sia, per i dispositivi A e B, dalla centralina meteo e dai sensori. Questo ci permette di valutare a priori il consumo energetico del dispositivo. Un esempio di caso in cui si verifica una forte variazione nei consumi è quello in cui un dispositivo di tipo A dovesse passare alla comunicazione via LTE per i suddetti motivi.

Vediamo ora una rappresentazione della rete che collega i dispositivi in campo al database in cui vengono salvati i dati che in seguito andremmo ad analizzare:



Figura IV‑3: Connessione bidirezionale tra i dispositivi in campo e il databse

I dispositivi di tipo A sono rappresentati dai triangoli mentre i riquadri contenenti i gateway LoRaWAN e i router LTE sono i dispositivi di tipo B e C. Dalle frecce bidirezionali si evince che la comunicazione tra i dispositivi è bidirezionale, ossia in un senso vengono mandati al database i dati dal campo, e nell’altro vengono mandati ai dispositivi in campo i segnali di controllo. Vediamo che i dispositivi di tipo A si connettono tramite la rete LoRaWAN ai dispositivi di tipo B o C nei dintorni e mandano i dati dal campo. I dati raccolti dai gateway LoRaWAN vengono poi passati al router LTE. Dallo schema, notiamo che i dispositivi di tipo A possono collegarsi a diversi gateway nelle vicinanze, in questo modo ci si accerta che anche se un gateway non è disponibile il segnale può comunque essere trasmesso usandone un altro. I dati passati al router LTE vengono poi mandati al cloud in cui viene fatta una cernita dei dati da salvare, e poi vengono scritti nel database SQL. E da notare, che anche i dispositivi di tipo B e C mandano dati al database, chiaramente in questo caso le misure dal campo vengono passate direttamente al router LTE, spediti al cloud e poi salvati nel database, senza passare dalla connessione LoRaWAN.

## Salvataggio dei campioni nel database

Per semplicità di ragionamento, facendo riferimento a Figura IV‑1, andiamo a suddividere le misurazioni in 2 pacchetti/categorie: il pacchetto energy management contenente tutti i dati provenienti dal caricatore, pannello solare, BMS e pacco batterie; il pacchetto sensoristica contenente i dati della centralina meteo e dei sensori.

La frequenza di campionamento è impostata a 15 minuti. Dato l’alto numero di sensori e la poca banda a disposizione nel protocollo LoRaWAN, i dati non vengono mandati tutti insieme, ma bensì sfalsati. Tutti i dati provenienti dal pacchetto energy management vengono mandati insieme, mentre sfalsati di 7,5 minuti, vengono mandati quelli del pacchetto sensoristica. Inoltre, le scelte di progetto impongono di sovrascrivere le variabili nel database solo in caso di variazione rispetto al campione precedente, altrimenti non viene scritto. Questa cernita di dati viene fatta nel cloud, ossia qualora il dato ricevuto dal campo sia uguale a quello ricevuto 15 minuti prima, allora non viene scritto nel database. Per questo motivo, il dataset non contiene i valori delle variabili per ogni istante di campionamento, ma soltanto le loro variazioni.

Si possono verificare delle situazioni in cui ci sia un problema nella trasmissione o ricezione dei dati. In particolare, qualora il dato non venga ricevuto dal cloud entro un intervallo massimo di 20 minuti (5 minuti dopo l’arrivo previsto del dato), viene salvato nel database l’ultimo campione, modificando però il bit di diagnostica da basso ad alto, segnalando così che l’apparato non è riuscito a comunicare con il cloud. Questa condizione riflette la mancata ricezione del campione, ma non ne determina la causa. La variabile di diagnostica torna al valore basso quando viene ricevuto un nuovo campione. Nella maggior parte dei casi ciò accade per un problema di connessione. Qualora il problema di connessione non si risolva nel breve termine una procedura automatizzata viene implementata in base alla tipologia di apparato. I dispositivi di tipo A in questi casi funzionano con il router finché non torna la connessione LoRaWAN, mentre i dispositivi di tipo B e C riavviano il router LTE.

In conclusione, nel dataset la scrittura del campione non avviene in modo regolare dato che essa dipende dalla sua variazione. Per esempio, la potenza del pannello viene campionata solamente nelle ore di luce poiché la notte la sua potenza è sempre nulla. Per questo motivo, durante la notte non sono presenti campioni riguardanti la potenza del pannello solare. E da considerare tra le possibili cause della mancanza di dati, anche i problemi e imprevisti provenienti dal campo.

# Elaborazione dei dati

## Query

I dati vengono acquisiti dal database (SQL Server) tramite Database Toolbox, il plugin di Matlab che permette di effettuare delle query direttamente dall’editor. Questo perché altrimenti per esportare i dati dal database dovremmo, per ogni query, passare per l’interfaccia di SQL server, esportare i dati in csv e poi caricarli manualmente dentro Matlab. Il plugin invece permette di scrivere direttamente da Matlab la query come da SQL server tramite un’interfaccia grafica e poi la converte in codice Matlab. Una volta lanciata la routine i dati vengono salvati direttamente nel workspace (Vedi un esempio di query in appendice in Figura IX‑1).

È stata fatta la scelta di progetto di prelevare per ogni dispositivo i dati partendo dal 01-05-2021fino alla data di spegnimento. Questa scelta è stata fatta per coprire sia la stagione estiva, in cui vi sono più ore di luce, e quindi ci sono meno spegnimenti, sia la stagione invernale in cui si verificano maggiormente i problemi dovuti a scarso irraggiamento. La query ritorna i dati sotto forma di una tabella contenenti 4 colonne, il tempo di ricezione, valore della variabile, il codice identificativo della variabile ed il bit di diagnostica.

Le variabili prelevate sono le seguenti:

* **Irradiazione**: L’irradiazione istantanea del pannello solare acquisita ogni 15 minuti
* **Tensione della cella minima**: Tramite le varie celle che compongono il pacco batterie, quella con la tensione minima.
* **SOC (State of Charge)**: Una stima effettuata dal BMS in Watt/ora (Wh) del livello di carica dell’insieme del pacco batterie. Presenta un errore cumulativo sulla stima che si azzera quando la batteria raggiuge lo 0% oppure il 100%.
* **Battery current in, Battery current out:** Le correnti istantanee in ingresso e in uscita alla batteria. Le useremo per calcolare il bilancio delle correnti della batteria.
* **Potenza del Pannello:** Potenza istantanea del pannello solare.

Inoltre, abbiamo prelevato il bit relativo alla diagnostica associato ad ogni campione. Esso viene posto alto dal ricevitore quando il campione atteso non viene ricevuto entro il tempo massimo consentito. Viene rimesso basso quando si riceve un nuovo campione. Permette di accertare la mancanza del campione nel database è dovuta ad un problema di ricezione.

## Prima preelaborazione dei dati (Sincronizzazione)

Per prima cosa andiamo a creare un cell array (di dimensioni 4x1) le cui celle saranno occupate nel seguente modo: nella prima il time-stamp, nella seconda il valore della variabile, nella terza il codice identificativo della variabile e nella quarta il bit di diagnostica. Il tempo viene convertito in formato Excel, un formato che esprime il tempo in numero di giorni a partire dal 01/01/1900.

La struttura del dato è la seguente:

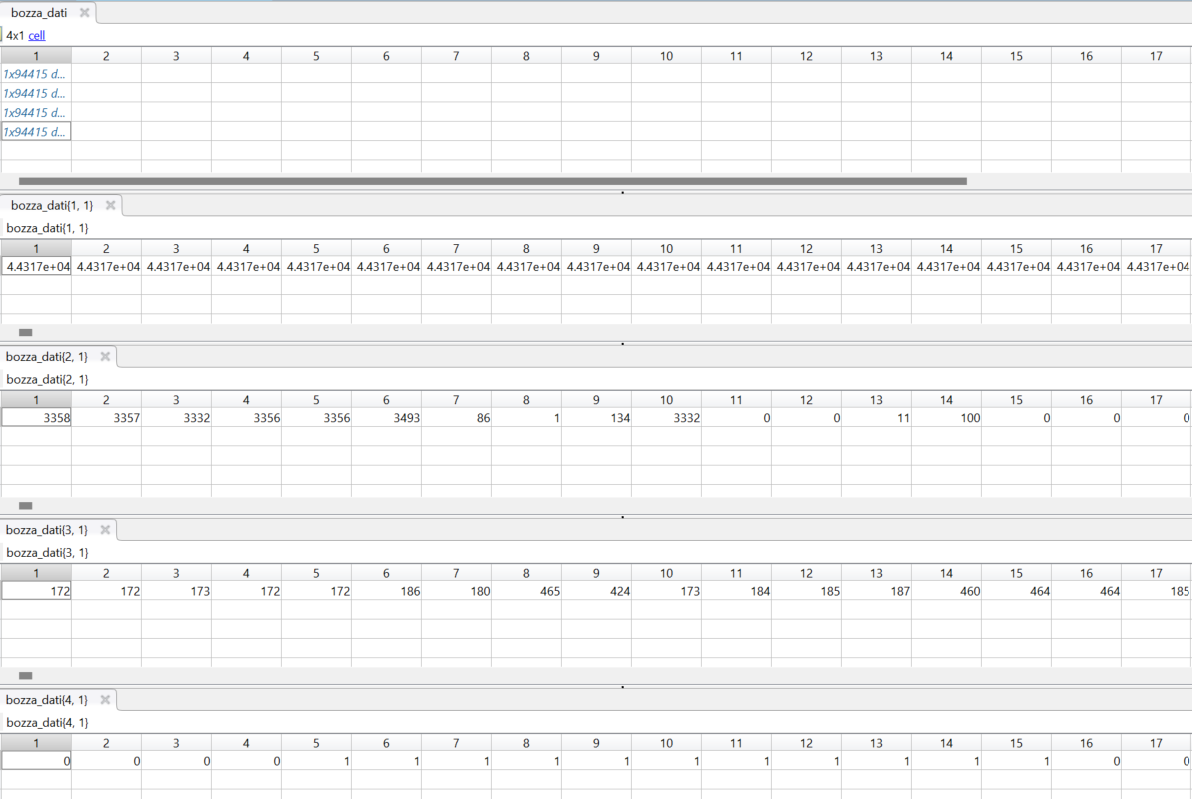


Figura V‑2: Cell array contenente i dati dal database

Come vediamo in **Errore. L'origine riferimento non è stata trovata.** ad ogni istante di tempo viene associato il valore, il codice e la variabile diagnostica. Ogni riga è costituita da una cella.

Per identificare le variabili usiamo una tabella, che chiameremo “ref”, di corrispondenza tra il nome della variabile ed il suo codice:

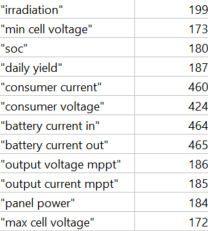


Figura V‑3: Tabella di corrispondenza dei codici delle variabili

Dato che nello stesso istante vengono campionate diverse variabili, colonne consecutive presentano lo stesso istante di tempo. Per ovviare a questo problema andiamo a creare una “struct” per ogni variabile, con i campi “name”, “value” e “diag”, che contiene solo i dati relativi ad essa. In seguito, le struct verranno raggruppate in un unico cell array che chiameremo “coord”:

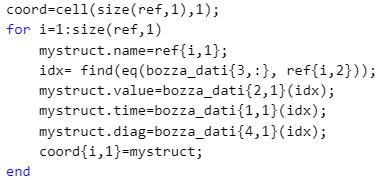


Figura V‑4: Cell array contenente le struct

Una volta creata questa struttura dati, dobbiamo andare a modificare il valore in cui il bit di diagnostica è alto. Come detto in precedenza, quando il ricevitore pone il bit di diagnostica alto, egli riscrive nel database un valore che non ha ricevuto. La nostra scelta di progetto è di modificare il valore dei campioni in diagnostica interpolando linearmente tra il campione precedente e quello successivo, considerando anche l’istante di campionamento.

Vediamo un esempio d’interpolazione:

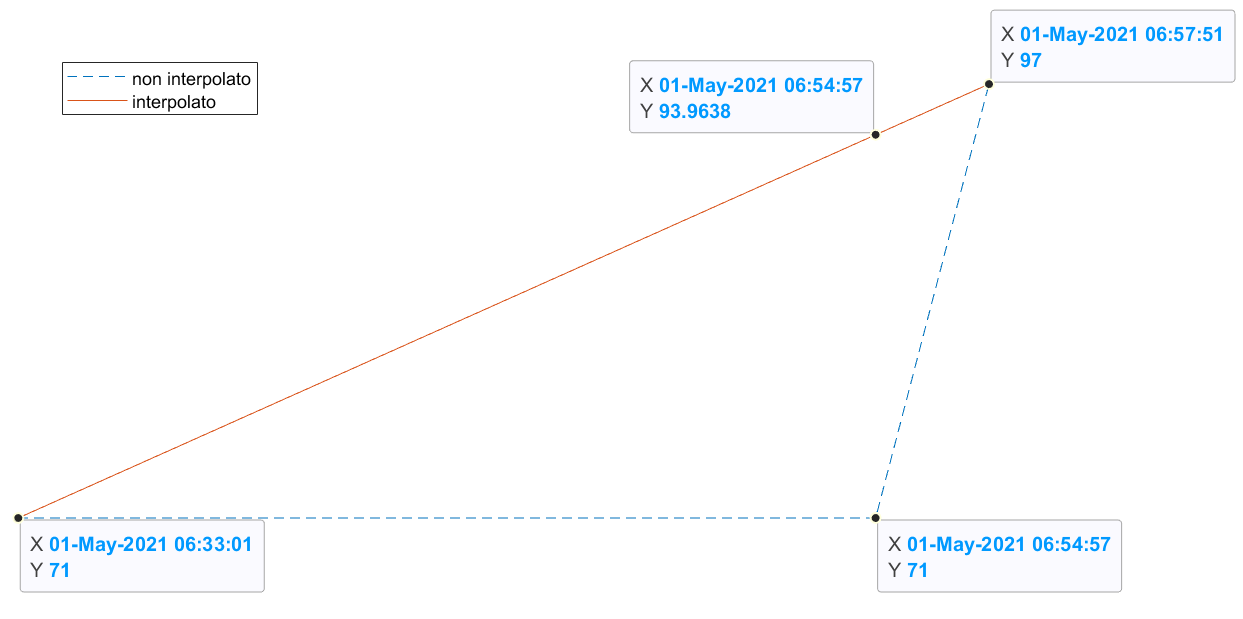


Figura V‑5: Interpolazione

Il campione interpolato è quello intermedio in data “01-05-2021 06:54:57”. La retta blu tratteggiata rappresenta i campioni non interpolati e quella rossa rappresenta i campioni interpolati. In questo modo si vuole approssimare al meglio il buco di dati. Abbiamo scelto l’interpolazione lineare piuttosto che altri modelli più complessi per rimanere il più fedele possibile ai dati a nostra disposizione, senza inserire effetti di addolcimento delle curve che introducessero più errore.

Di seguito il codice utilizzato:

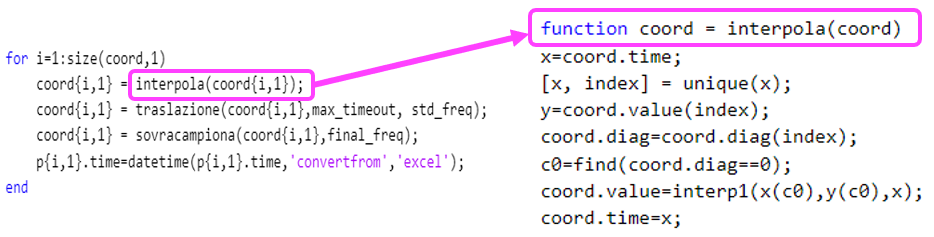


Figura V‑6: Interpolazione dei campioni in diagnostica

Abbiamo usato la funzione “interp1” di Matlab. I primi due argomenti di questa funzione sono gli istanti di tempo e i relativi valori dei campioni non in diagnostica, grazie a cui faremo l’interpolazione, mentre il terzo argomento rappresenta la totalità degli istanti di tempo includendo anche i campioni in diagnostica. In questo modo, i valori degli istanti presenti in “x” ma non in “x(c0)” vengono interpolati usando “y(c0)”.

Avendo ricondotto i valori in diagnostica al probabile valore dal campo, consideriamo di aver risolto tutte le condizioni di diagnostica. Possiamo dunque considerare che tutti i campioni sono validi e quindi, qualora l’intervallo di tempo tra due campioni consecutivi sia maggiore di 20 minuti andiamo ad inserire 15 minuti prima del campione successivo un campione di valore uguale a quello precedente:

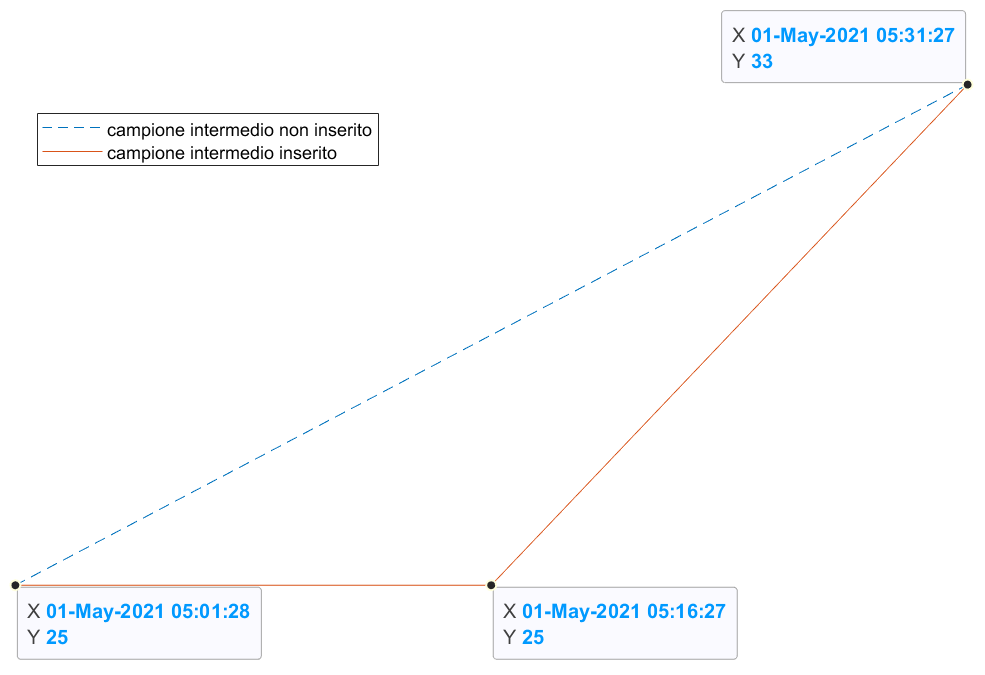


Figura V‑7: Traslazione del dato

In questo caso l’intervallo tra i due punti “01-05-2021 05:01:28” e “01-05-2021 05:31:27” è maggiore di 20 minuti (retta blu tratteggiata), per questo motivo trasliamo il valore del primo punto 15 minuti prima del secondo punto e creiamo un campione intermedio (retta rossa). La retta blu tratteggiata rappresenta i campioni senza l’aggiunta del campione intermedio mentre quella rossa rappresenta i campioni dopo l’aggiunta del campione intermedio.

Di seguito il codice utilizzato:

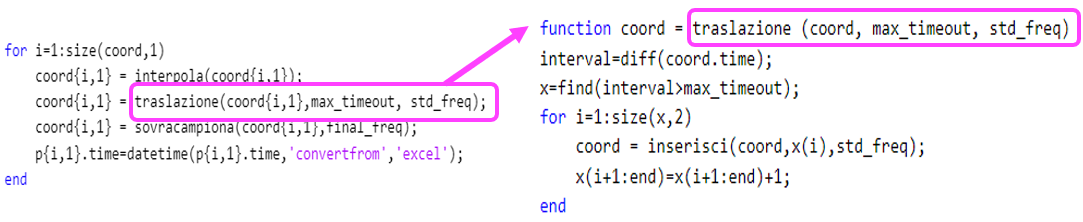


Figura V‑8: Sovrascrittura dei valori invariati

La funzione “traslazione” prende come argomento i seguenti argomenti:

* “max\_timeout”: L’intervallo di tempo in cui va ricevuto il campione. Qualora non venga ricevuto si deduce che il campione non è stato trasmesso perché invariato (20 minuti)
* “std\_freq”: Intervallo standard di campionamento (15 minuti)

L’inserimento del campione intermedio viene fatto tramite la funzione “inserisci”.

In ultimo andiamo a fare un sovra campionamento al secondo di modo da riempire il dataset sovrabbondantemente, per poi estrarre un campione al minuto:



Figura V‑9: Funzione di sovra campionamento

In questo caso usiamo la funzione “interp1” sia per sovra campionare che per sotto campionare. La funzione “sovracampiona” prende come argomento “final\_feq” che è la frequenza di campionamento ad 1 minuto che vogliamo risulti come frequenza di campionamento definitiva. Alla fine sull’intervallo in Figura V‑7 otteniamo la seguente distribuzione:

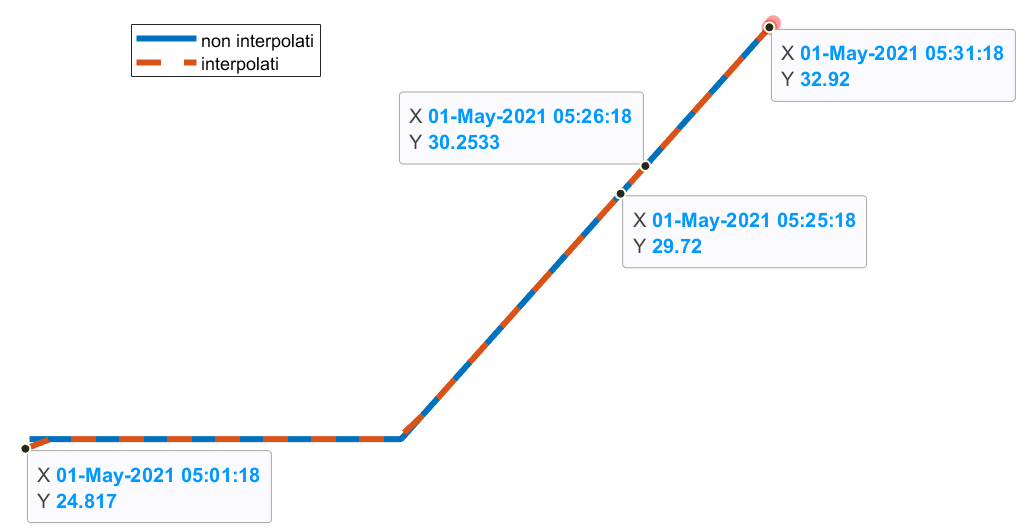


Figura V‑10: Sovra campionamento al minuto

Possiamo osservare che ora abbiamo un campione al minuto. In questo modo ora il dato risulta sincronizzato e con una frequenza di campionamento 15 volte maggiore rispetto a prima. Abbiamo deciso di aumentare la frequenza di campionamento per rilevare tutti i picchi del dato he non sarebbero stati rilevati prendendo l’intervallo a 15 minuti.

Gli step descritti in Figura V‑5, Figura V‑7 e Figura V‑10 vanno ripetuti per tutte le variabili prelevate. Una volta fatta quest’operazione abbiamo un numero di struct pari al numero di variabili, ognuna con un proprio time stamp, nome e valore.

A questo punto dobbiamo ottenere un time stamp. Per fare ciò, usiamo la funzione “allineo” in cui andiamo a trovare, per tutte le variabili, il tempo del primo e dell’ultimo campione. Confrontandoli scegliamo il tempo di inizio massimo e quello di fine minimo, andando a definire così quale sarà l’intervallo su cui valuteremo i dati. Le colonne che non sono comprese nell’intervallo vengono eliminate.

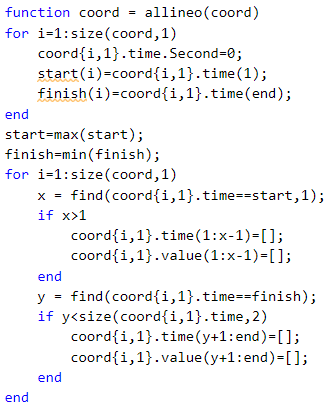


Figura V‑11: allineamento delle sequenze temporali

In questo modo ora possiamo creare un’unica struct con un unico time stamp e un campo per ogni variabile:

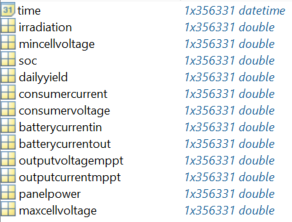


Figura V‑12: Struct finale

A riprova che il time stamp è unico vediamo che le dimensioni di ogni campo è uguale.

Andando a riscrivere il dato quando non è presente andiamo a risolvere un problema che occorreva nel salvataggio dei dati durante il periodo notturno. Infatti, alcune variabili come la potenza del pannello solare hanno valori diversi da zero solo durante il giorno, mentre la notte rimane costantemente a zero. Come si vede in Figura V‑13 accadeva che l’ultimo campione in cui il pannello era attivo (01-Jul-2021 18:07:30) avesse valore nullo, mentre il primo valore campionato la mattina seguente fosse già diverso da zero, e per questo motivo, non avendo campioni intermedi, si registrava come si ci fosse stata una leggera crescita di potenza durante la notte (retta blu tratteggiata). Ciò è stato corretto andando a riscrivere il campione con la procedura di traslazione (retta rossa).

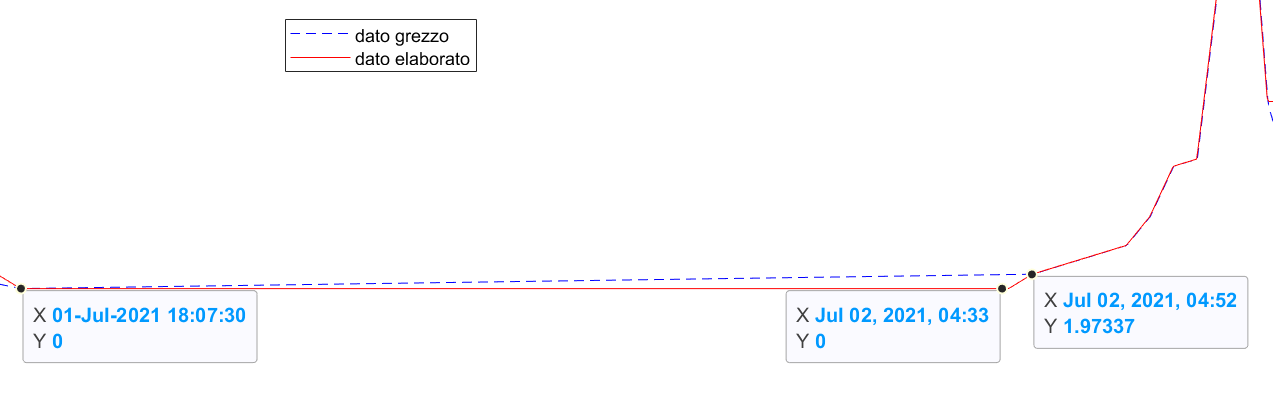


Figura V‑13: Messa a zero della potenza del pannello di notte

Un altro tipo di problema si presentava quando la torre entrava in diagnostica. In alcuni casi il ricevitore scriveva i valori delle variabili come nulli e quindi si verificavano picchi che falsavano completamente l’andamento delle variabili. Ciò si verificava per esempio per le variabili relative alla batteria, come la tensione della cella minima. In Figura V‑14 notiamo l’andamento della tensione della cella minima che sembra andare a zero nel lasso di tempo relativo alla diagnostica (retta blu tratteggiata), ciò viene corretto riscrivendo il campione nullo tramite la procedura di interpolazione (curva rossa):

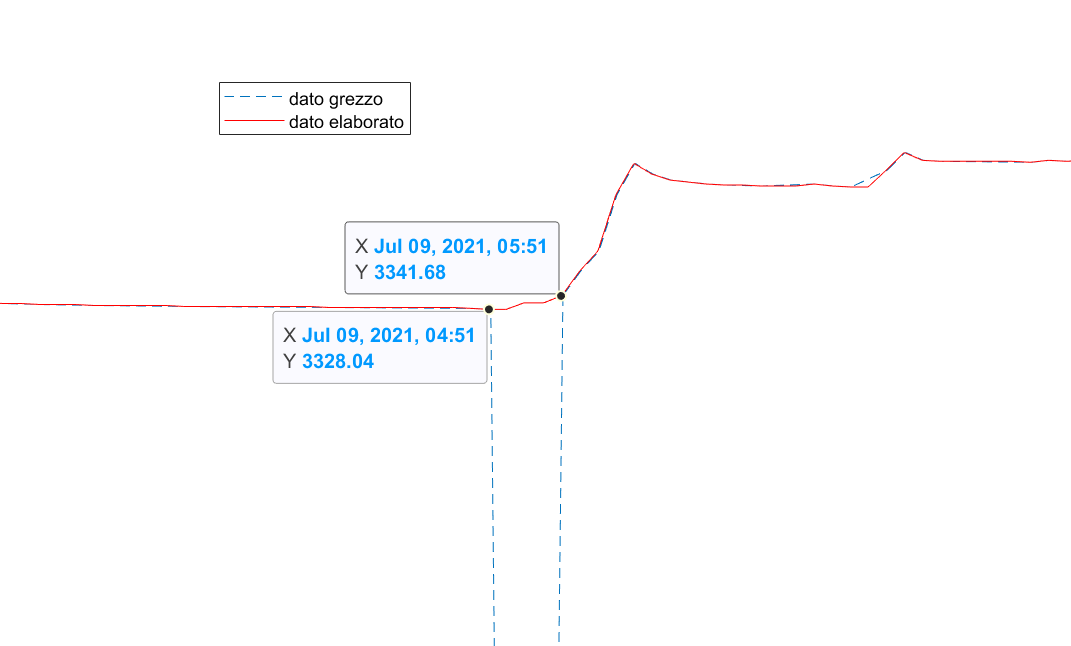


Figura V‑14: Correzione del valore nullo in diagnostica

Andando ad interpolare tra il valore precedente e successivo all’entrata in diagnostica è stato possibile ricostruire l’andamento della tensione della cella minima.

## Estrazione di sequenze lunghe 3 giorni

In questo paragrafo andiamo a descrivere i parametri fondamentali nella costruzione delle sequenze che formeranno il dataset. Li definiamo come variabili globali e li inseriamo all’inizio del main per poter cambiare le impostazioni del progetto in modo semplice ed efficace. Essi non vengono mai scritti successivamente, vengono solamente letti nelle varie routine. Essendo variabili globali essi non devono essere passati come argomento delle funzioni rendendo più snella la dichiarazione delle funzioni rispetto a variabili locali:

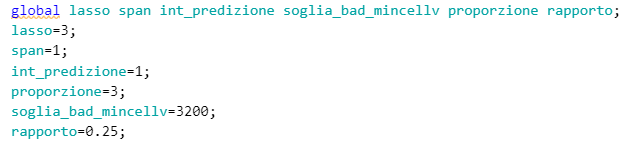


Figura V‑15: Parametri

La lunghezza delle sequenze e lo sfalsamento tra due sequenze consecutive sono rappresentati dalle variabili globali “lasso” e “span” e vengono espressi in giorni. L’implementazione di un’etichettatura delle sequenze per effettuare una classificazione predittiva si fa tramite la variabile “int\_predizione” specificando quanti giorni prima si vuole predire un evento. La variabile “proporzione” specifica il rapporto tra sequenze sane e patologiche nel dataset. La soglia per la selezione delle sequenze patologiche è “soglia\_bad\_mincellv” e si riferisce al valore della tensione della cella minima in mV. La variabile “rapporto” riguarda la ripartizione in percentuale delle sequenze tra test e train e si applica solo nel caso di partizione statica. Maggiori approfondimenti sull’uso di tali variabili verranno fornite nelle sezioni seguenti.

Il main è strutturato nel seguente modo:



Figura V‑16: Main

Per ogni traliccio viene ripetuta la seguente procedura:

* Scegliamo le variabili che vogliamo inserire nelle sequenze tramite il campo “name” della struct “variabili”
* Tramite “estrazione\_sequenze” preleviamo dal dato sincronizzato le sequenze eleggibili per far parte del dataset evitando quegli intervalli temporali in cui non sono arrivati dati dal campo
* In “sospetti” andiamo a discriminare le sequenze sane da quelle patologiche usando “soglia\_bad\_mincellv” e ritorniamo l’indice delle sequenze a “int\_predizione” giorni da esse. Per semplicità chiameremo quest’ultime sequenze predittive.
* La routine “normalizzazione” viene usata per normalizzare il dato
* In “etichette” dividiamo le sequenze tra Train e Test
* Tramite “pulizia” e “salvataggio” salviamo i dati che ci servono

Una volta effettuata questa procedura usiamo la funzione “dataset1” per formare e salvare il dataset che andremmo a passare all’algoritmo di classificazione. In questo dataset vengono amalgamate le sequenze selezionate provenienti dai vari tralicci che abbiamo scelto di analizzare.

### Estrazione delle sequenze

Andiamo a suddividere il dato ottenuto fin qui in sequenze da poter dare in ingresso all’algoritmo di classificazione. In questo esempio andiamo a suddividere il dato in sequenze di 3 giorni sfalsate di 1 giorno rispetto alla precedente. Abbiamo scelto di avere una sovrapposizione parziale delle sequenze per la scarsa casistica di guasto. In questo modo possiamo avere una maggiore consistenza del dataset.

Questa prima fase è parte di una procedura che si trova nel “main” e che andrà reiterata per ogni traliccio grazie ad un ciclo while. I tralicci che scegliamo sono riportati in un file di testo che viene letto durante il ciclo di modo da velocizzare la procedura. Le variabili che selezioneremo e useremo nelle sequenze sono salvate in “variabili.name”. In questo modo possiamo variare a piacere le variabili che andremmo ad usare e scegliere quelle ottimali in base ai risultati forniti dall’algoritmo.

Il codice è il seguente:

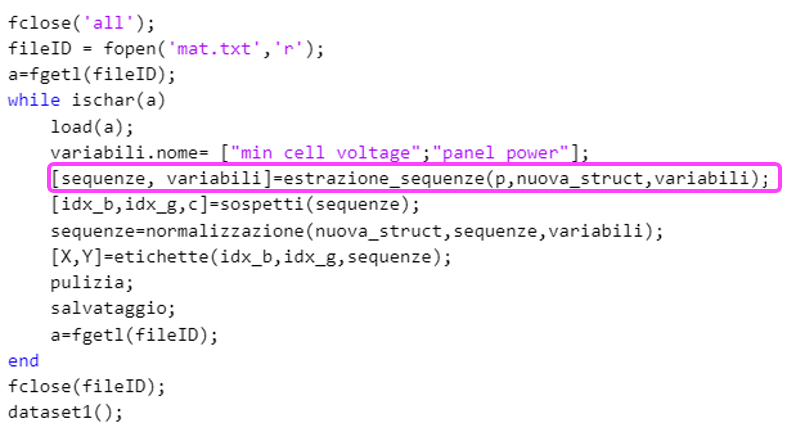


Figura V‑17: Estrazione delle sequenze nel main

Per prima cosa dobbiamo trovare un modo di isolare i singoli giorni per poi assemblare le sequenze di 3 giorni. Per fare ciò, in “estrazione\_sequenze” che prende come argomento il dato non sincronizzato (p), il dato sincronizzato (nuova\_struct), le variabili da selezionare (varabili) e ritorna sia le sequenze che le variabili selezionate. In “estrazione\_sequenze” usiamo la funzione “mezzanotte”. Il suo compito è di trovare ogni giorno la prima occorrenza di un campione nell’intervallo tra le 20.00 e le 05.00. In questo modo riusciamo ad escludere tutti quei lassi di tempo per cui non ci sono dati per 24 ore o più ed escluderli.

A seguire un esempio delle sequenze che vogliamo evitare di selezionare:

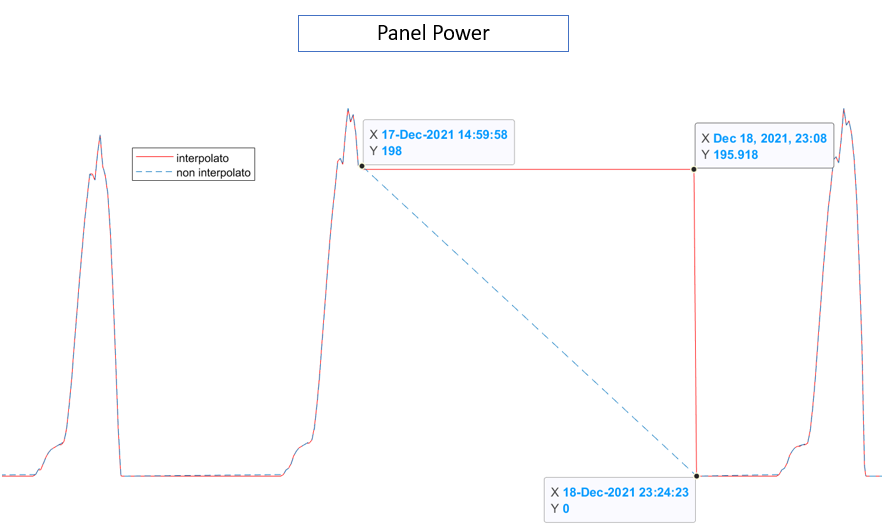


Figura V‑18: Differenza di andamenti tra dato sovra campionato e non, là dove si verifica un buco di dati

In Figura V‑18: Differenza di andamenti tra dato sovra campionato e non, là dove si verifica un buco di datiFigura V‑18 abbiamo un intervallo di giorni in cui troviamo una mancanza di dati nei dati non interpolati dal “17-Dec-2021 14:59:58” al “18-Dec-2021 23:24:23” (curva blu tratteggiata), per questo motivo il 17 dicembre non troviamo nessun campione nell’intervallo tra le 20.00 e le 05.00 e quindi non verrà considerato nelle sequenze da estrarre. Inoltre, non verrà considerato nemmeno il 18 dicembre.

La routine per scandire i giorni (mezzanotte) è la seguente:



Figura V‑19: funzione "mezzanotte" per scandire i giorni

Il ciclo while in Figura V‑19 serve a selezionare solo il primo campione nell’intervallo tra le 20.00 e le 05.00 per ogni giorno (al timestamp vanno aggiunte 2 ore per il fuso orario). La funzione prende come argomento il time stamp del dato non sincronizzato (p) e ritorna l’indice dei campioni che segnano la fine della giornata (dhour) e il corrispettivo istante di tempo in “p” (d0). Una volta che la funzione ritorna questi valori, facciamo la differenza di tempo tra due elementi consecutivi di “d0”:

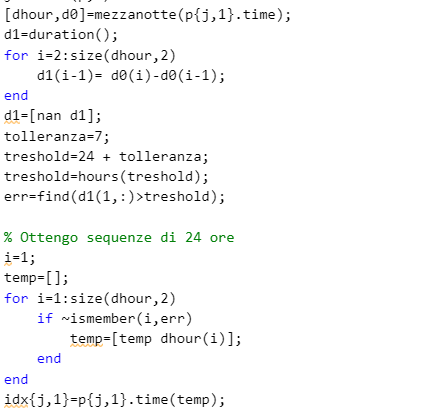


Figura V‑20: Selezione di sequenze da un giorno

Qualora l’intervallo sia maggiore di 31 ore (24 ore + 7 ore di tolleranza), segniamo tale intervallo in “err”. In “idx” andiamo a salvare tutti gli intervalli validi escludendo quelle più lunghe di 31 ore.

A seguire ciò che otteniamo:



Figura V‑21: date di fine giornata

Usiamo ora queste date per selezionare le giornate da concatenare per creare le sequenze:

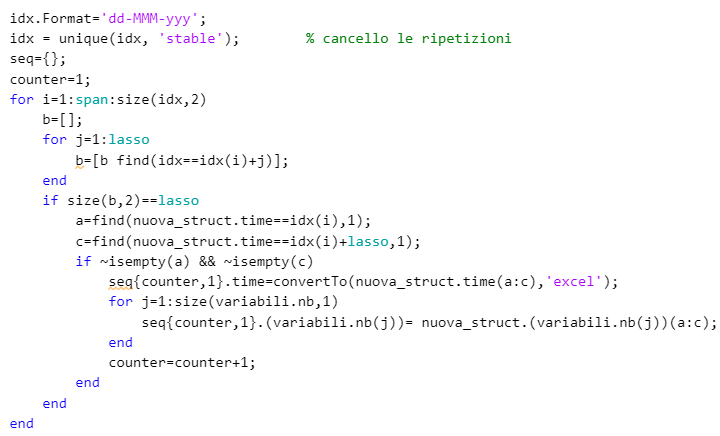


Figura V‑22: Creazione delle sequenze

La lunghezza delle sequenze e lo sfalsamento tra due sequenze consecutive sono parametri rappresentati dalle variabili globali “lasso” e “span”. Usiamo le date contenute in “idx” per estrarre le sequenze da “nuova\_struct”, il dato sincronizzato ottenuto nel capitolo precedente [Prima preelaborazione dei dati (Sincronizzazione)].

In figura vediamo l’andamento della tensione della cella minima sull’intervallo totale comparando le sequenze estratte e la totalità del dato sinconizzato:

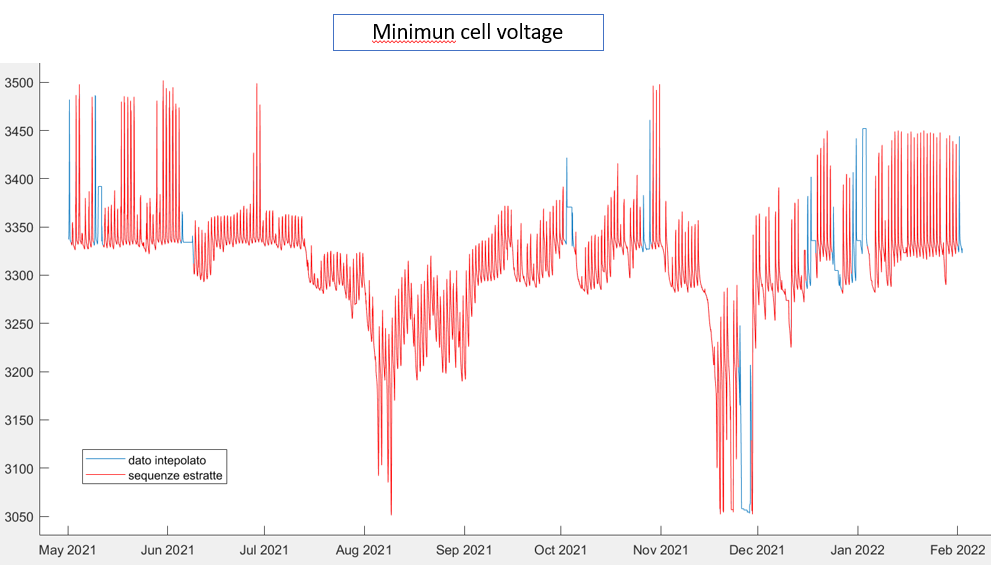


Figura V‑23: Paragone tra le sequenze estratte e la totalità del dato sincronizzato

In figura la curva rossa rappresenta le sequenze estratte per costituire il dataset, mentre in blu si vedono le parti del dato sincronizzato che non sono state selezionate perché contenenti buchi di dati. Se andiamo a zoomare su una porzione, vediamo che le sequenze in blu (quelle che non faranno parte del dataset) presentano lunghi intervalli in cui il dato rimane costante, sintomo del fatto che non è stato ricevuto il dato per un lungo periodo di tempo e quindi è giusto escluderle:

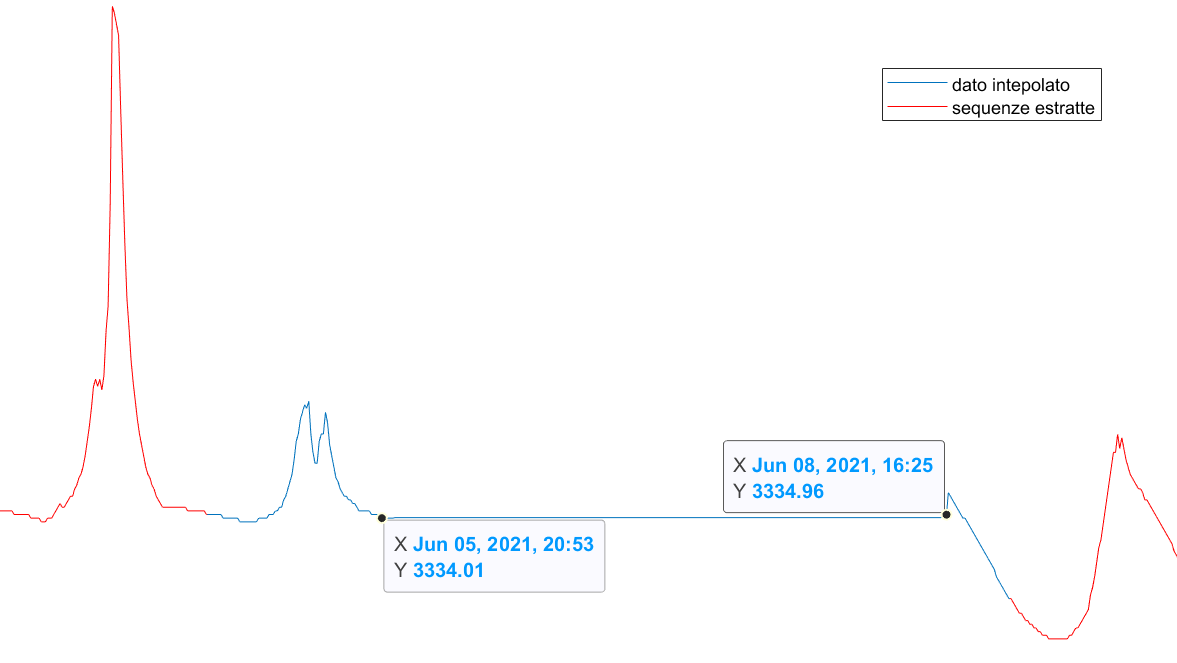


Figura V‑24: Lasso temporale non valido ai fini del dataset

Infatti, tra il “05-Jun-2021 20:53” e il “08-Jun-2021 16:25” abbiamo fatto un’interpolazione ma non ci sono stati dati ricevuti, e infatti la retta rimane piatta per quasi 3 giorni. Per questo motivo quest’intervallo viene escluso dal dataset.

Le sequenze estratte sono delle struct, ognuna con un campo “time” per il tempo, e una per ogni variabile con il proprio nome associato:



Figura V‑25: Esempio di una sequenza

Dato che abbiamo sincronizzato il dato ogni sequenza ha lo tesso numero di campioni. Inoltre, i nomi delle variabili vengono presi ed associati automaticamente. Questo ci permette di richiamare le variabili nelle sequenze direttamente tramite il proprio nome e non tramite il numero di riga, rendendo la gestione delle variabili molto più semplice.

### Identificazione delle sequenze sane e patologiche

Andiamo a differenziare le sequenze sane da quelle patologiche tramite la funzione “sospetti”:

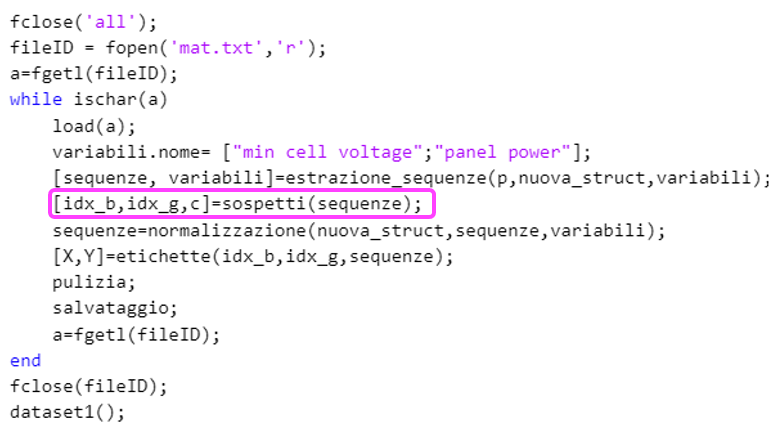


Figura V‑26: Funzione "sospetti" nel main

La funzione prende come argomento le sequenze estratte precedentemente e ritorna gli indici delle sequenze sane e patologiche.

Una sequenza viene detta patologica quando almeno un campione della tensione della cella minima si trova sotto il valore nominale della batteria a 3,2 volt. Il resto delle sequenze viene considerato sano.

L’implementazione è la seguente:

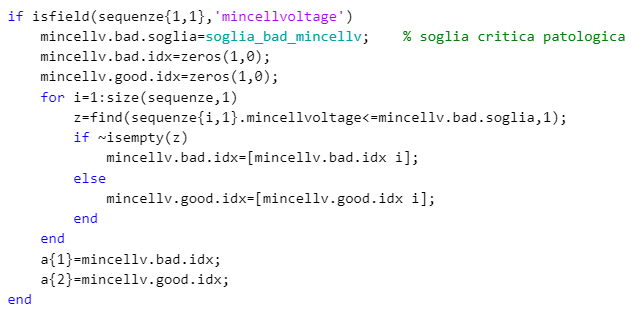


Figura V‑27: Suddivisione delle sequenze tra sane e patologiche

La struct “mincellv” si riferisce alla tensione della cella minima ed è caratterizzata dai campi “bad” e “good l’indice delle sequenze appartenenti al campo (“idx”).

Una volta identificate le sequenze effettivamente patologiche andiamo a cercare le sequenze predittive.

L’implementazione è la seguente:

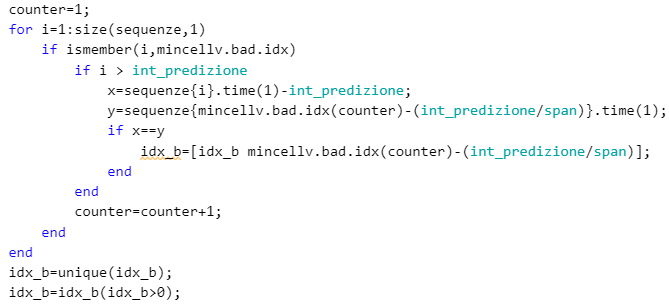


Figura V‑28: Estrazione delle sequenze predittive

In Figura V‑28 possiamo vedere che in “int\_predizione” seleziona l’intervallo di predizione. Vengono salvati gli indici delle sequenze predittive. La stessa procedura vale per tutte le sequenze, sia sane che patologiche.

Poiché vogliamo rispettare una proporzione tra le sequenze sane e quelle patologiche, e tipicamente abbiamo un numero di sequenze sane molto maggiore di quelle patologiche, andiamo a fare la seguente operazione:

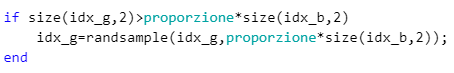


Figura V‑29: Ripetto delle proporzioni tra sequenze sane e patologiche

Tramite la variabile globale “proporzione” decidiamo quante sequenze selezionare, e poi grazie alla funzione “randsample” le estraiamo in modo randomico da quelle che abbiamo. In questo modo le sequenze sane selezionate cambiano ogni volta che eseguiamo la routine.

Una volta eseguite queste operazioni otteniamo una cernita di sequenze classificate tra sane e patologiche.

A seguire un esempio di sequenza patologica e la relativa sequenza predittiva 7 giorni prima:

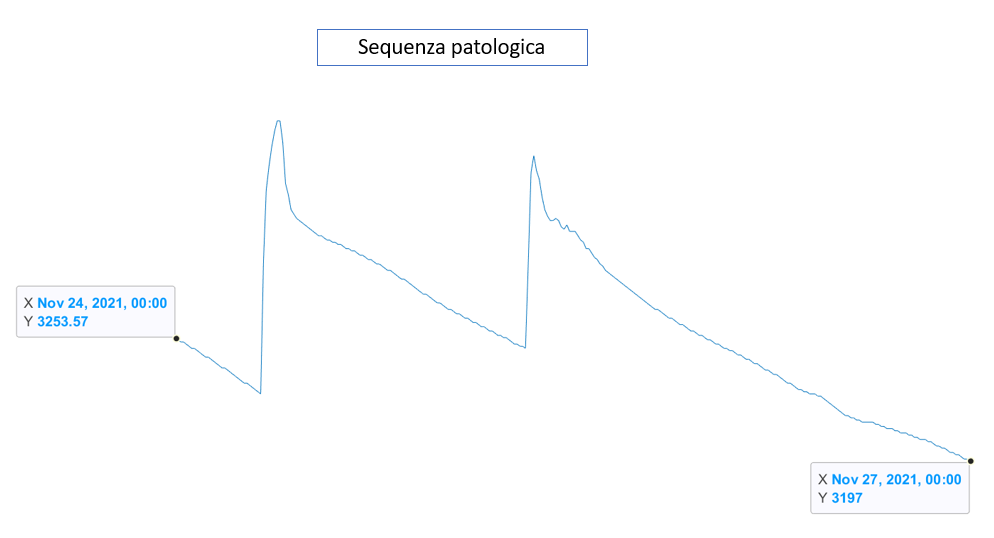


Figura V‑30: sequenza patologica



Figura V‑31: Sequenza predittiva

Vediamo che in Figura V‑30 la sequenza patologica è considerata tale perché nell’ultima parte scende al di sotto d 3200 [mV]. La sequenza predittiva viene selezionata calcolando che l’ultimo campione della sequenza predittiva deve essere 7 giorni precisi prima dell’ultimo campione della sequenza patologica, come vediamo dai marker.

A seguire vediamo come viene prelevata la sequenza predittiva:

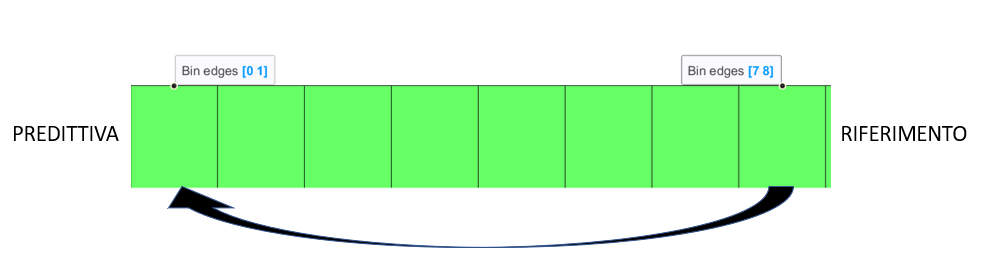


Figura V‑32: Selezione della sequenza predittiva con predzione a 7giorni e sequenza di riferimento sana

In questo caso la previsione è a 7 giorni, quindi la sequenza predittiva è quella 7 sequenze prima. La sequenza predittiva viene selezionata allo stesso modo, sia che si riferisca ad una sequenza sana che patologica.

Vediamo il caso in cui la sequenza di riferimento sia patologica:

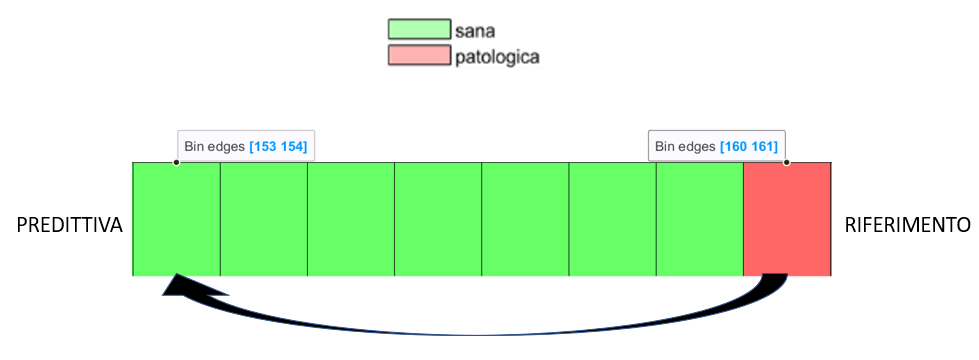


Figura V‑33:Selezione della sequenza predittiva con predzione a 7giorni e sequenza di riferimento patologica

Il meccanismo di selezione è lo stesso tra Figura V‑32 e Figura V‑33. Da notare che la sequenza predittiva acquisisce l’etichetta (sana o patologica) della sequenza di riferimento. Per questo motivo la sequenza predittiva in Figura V‑33 verrà considerata patologica ai fini del dataset per la classificazione predittiva a 7 giorni. Ciò è possibile perché sappiamo per certo che dopo 7 giorni si verificherà un evento patologico.

Nel seguente grafico possiamo vedere le sequenze patologiche di riferimento (rosso) rispetto alle sequenze patologiche predittive (arancioni):

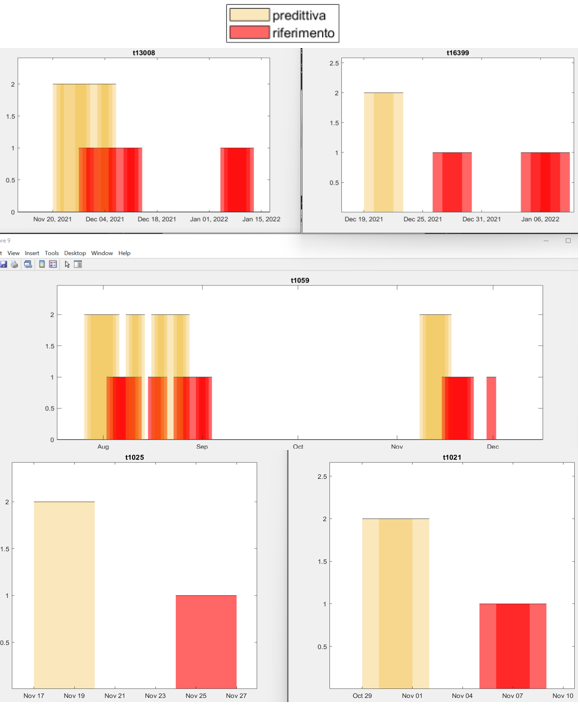


Figura V‑34: Disposizione tra sequenze di riferimento e predittive a 7 giorni per ogni torre

In Figura V‑34 notiamo come le sequenze predittive siano sfalsate a sinistra rispetto a quelle di riferimento, questo perché sono predittive. In teoria dovrebbero essere in numero uguali a quelle di riferimento, in pratica però si possono verificare che la sequenza predittiva cada in un intervallo dove abbiamo un buco di dati. In questo caso non viene generata nessuna sequenza predittiva per una data sequenza di riferimento. Ciò si verifica nei due grafici in alto (t16399 e t13008) in cui rispettivamente le sequenze patologiche nell’intorno del 6 gennaio e nell’intervallo tra il primo e il 15 gennaio ci sono sequenze patologiche di riferimento ma non predittive.

Inoltre, possiamo notare come in alcuni casi le sequenze predittive e di riferimento si sovrappongano tra loro. Ciò si verifica per il grafico in altro a sinistra (t13008) perché le sequenze di riferimento sono spalmate in un intervallo maggiore di 7 giorni e quindi, le sequenze predittive delle ultime sequenze di riferimento si sovrappongono con le prime sequenze di riferimento.

A seguire diamo una panoramica dell’andamento della tensione della cella minima per ogni traliccio selezionato. Usiamo l’intervallo di tempo totale (01-05-2021 ad oggi). Le sequenze verdi sono quelle sane e quelle rosse sono quelle patologiche:

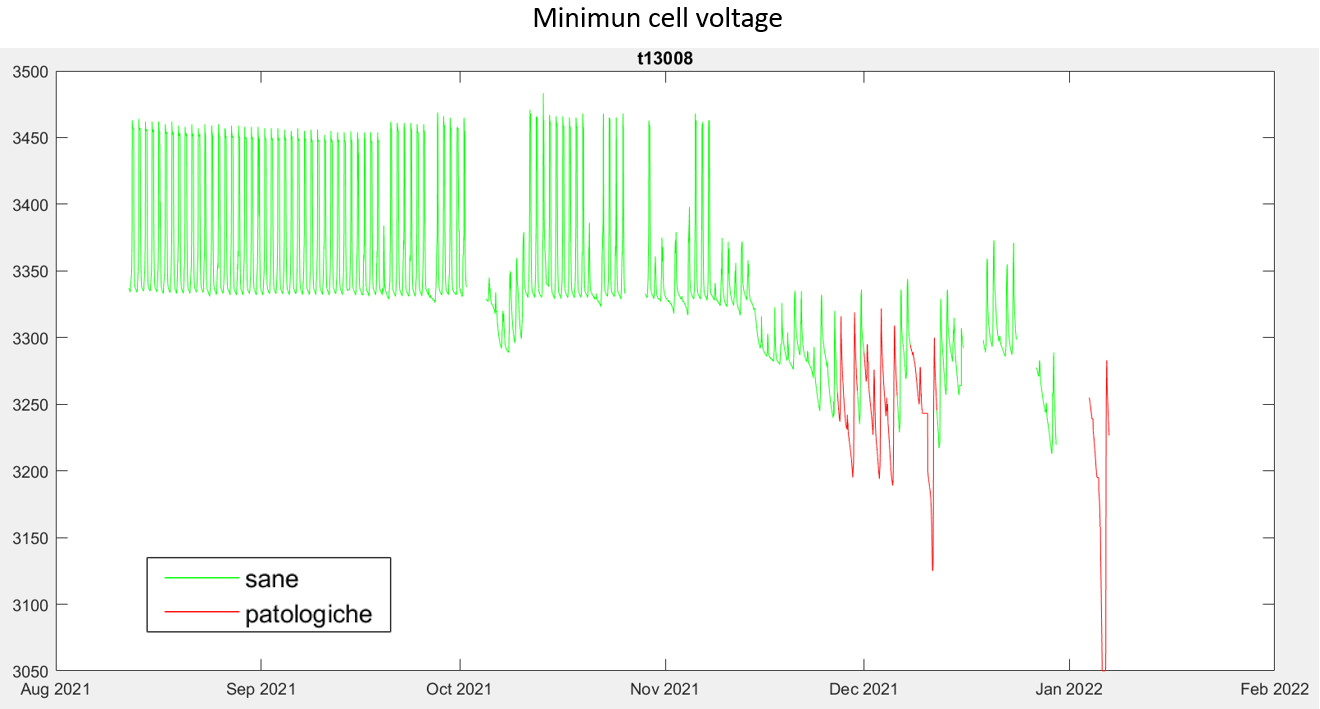


Figura V‑35: Traliccio 13008

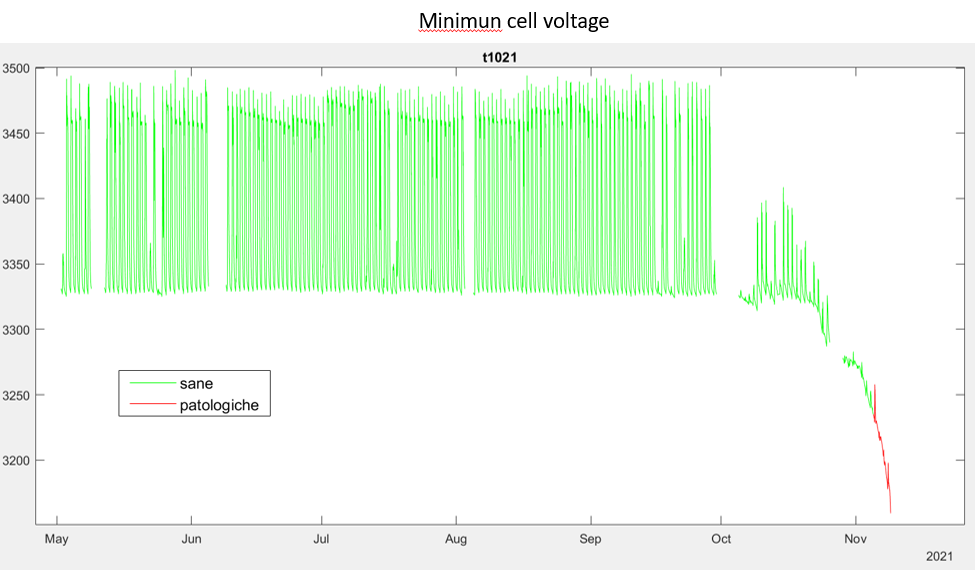


Figura V‑36: Traliccio 1021

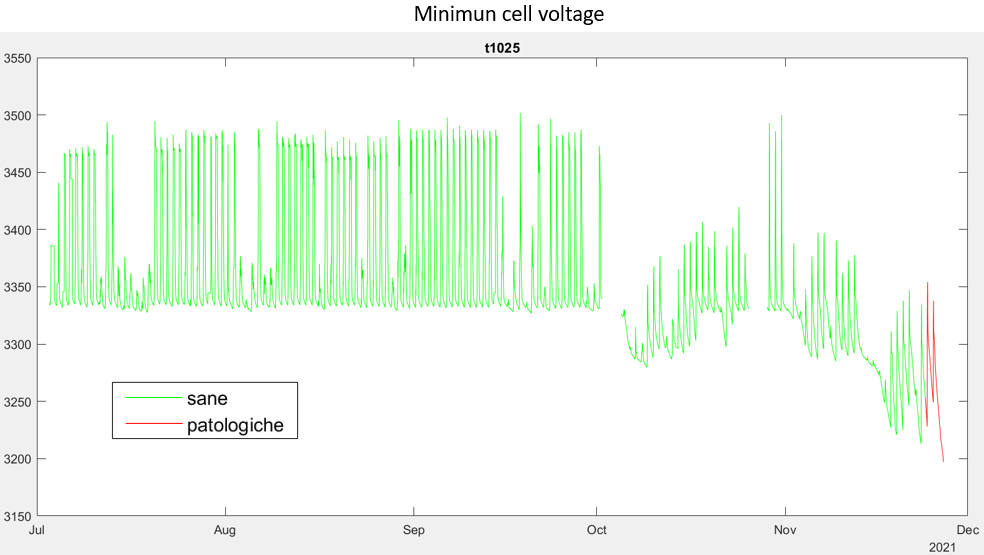


Figura V‑37: Traliccio 1025

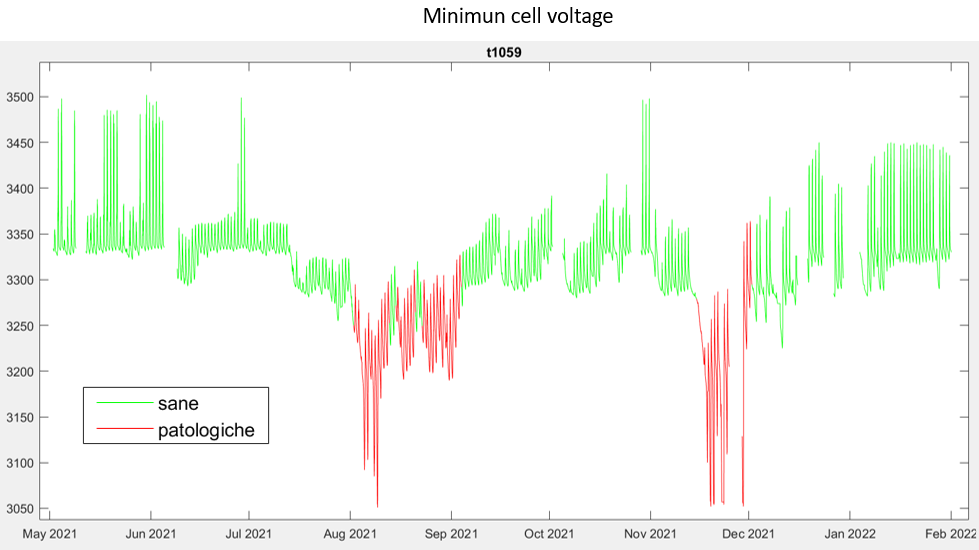


Figura V‑38: Traliccio 1059

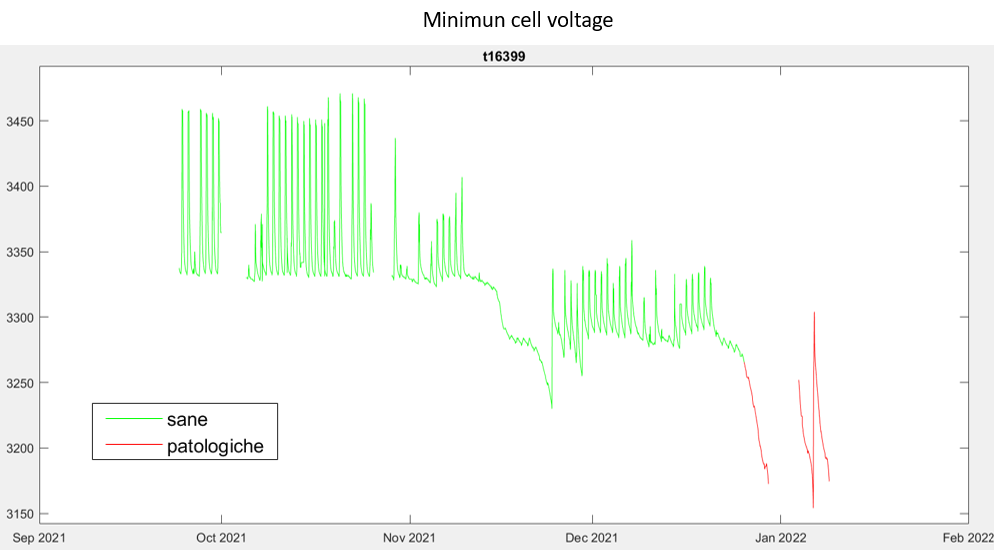


Figura V‑39: Traliccio 16399

Vediamo che la casistica è molto eterogenea: In Figura V‑36 Figura V‑37 e Figura V‑39 le sequenze patologiche sono le ultime sequenze prima di uno spegnimento definitivo. Vediamo invece, che nel caso di Figura V‑35 Figura V‑38 le sequenze patologiche rappresentano un periodo di forte scarica della batteria, che non sfocia sempre in uno spegnimento (tensione della cella minima a 3050 volt), e che poi viene seguito da una ripresa. Questo perché andando a fissare una soglia fissa per le sequenze patologiche non abbiamo la capacità di discriminare tra questo tipo di situazioni.

Alcuni apparati non coprono tutto l’intervallo temporale perché potrebbero essersi spenti definitivamente (Figura V‑36, Figura V‑37), oppure perché la data di implementazione sul campo potrebbe essere successiva al 01-05-2021 (Figura V‑35, Figura V‑39).

Un altro punto rilevante è che dato che le sequenze si sovrappongono parzialmente ogni giornata è presente in 3 sequenze. Per questo motivo una sequenza patologica può contenere giorni contenuti in sequenze sane.

A seguire, vediamo un esempio di tale concetto. Prendiamo in considerazione il traliccio in Figura V‑36 nel periodo a cavallo del periodo in cui l’andamento della tensione della cella minima diventa patologico:



Figura V‑40: Periodo in cui l’andamento della tensione della cella minima diventa patologico

Zoomiamo su tale periodo andando ad estrarre la prima sequenza patologica e le due sequenze precedenti sane che condividono almeno un giorno con quella patologica

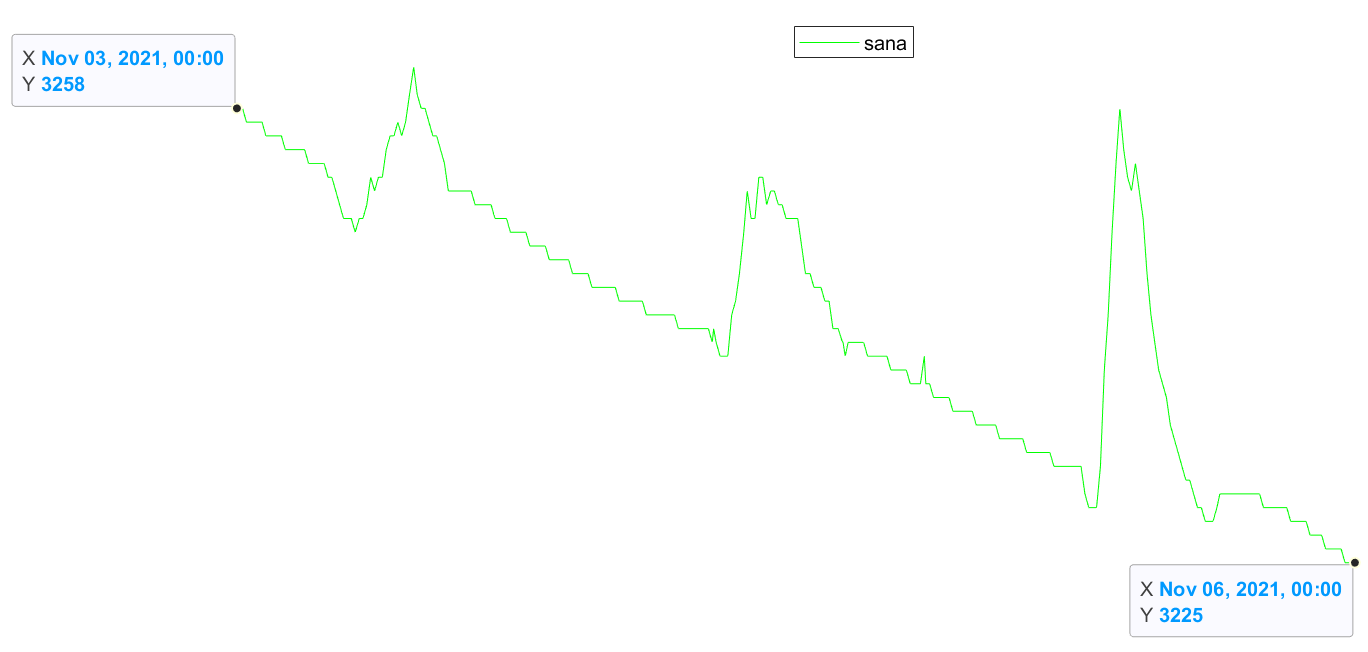


Figura V‑41: Prima sequenza

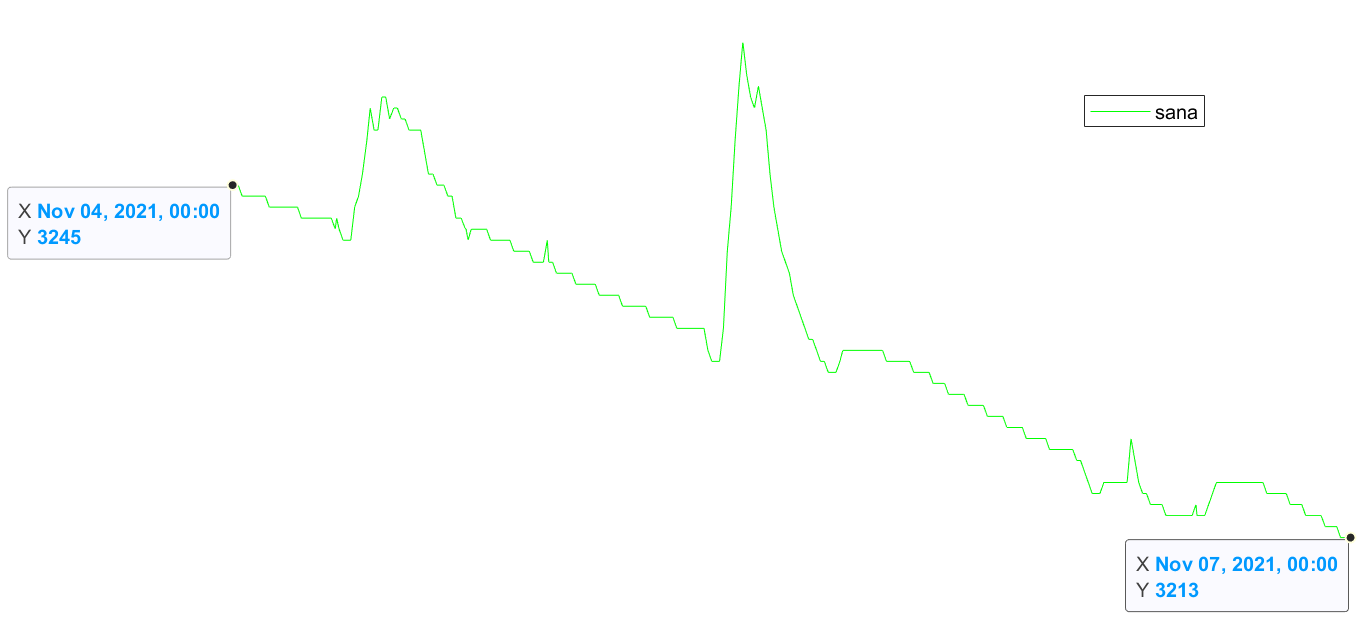


Figura V‑42: Seconda sequenza

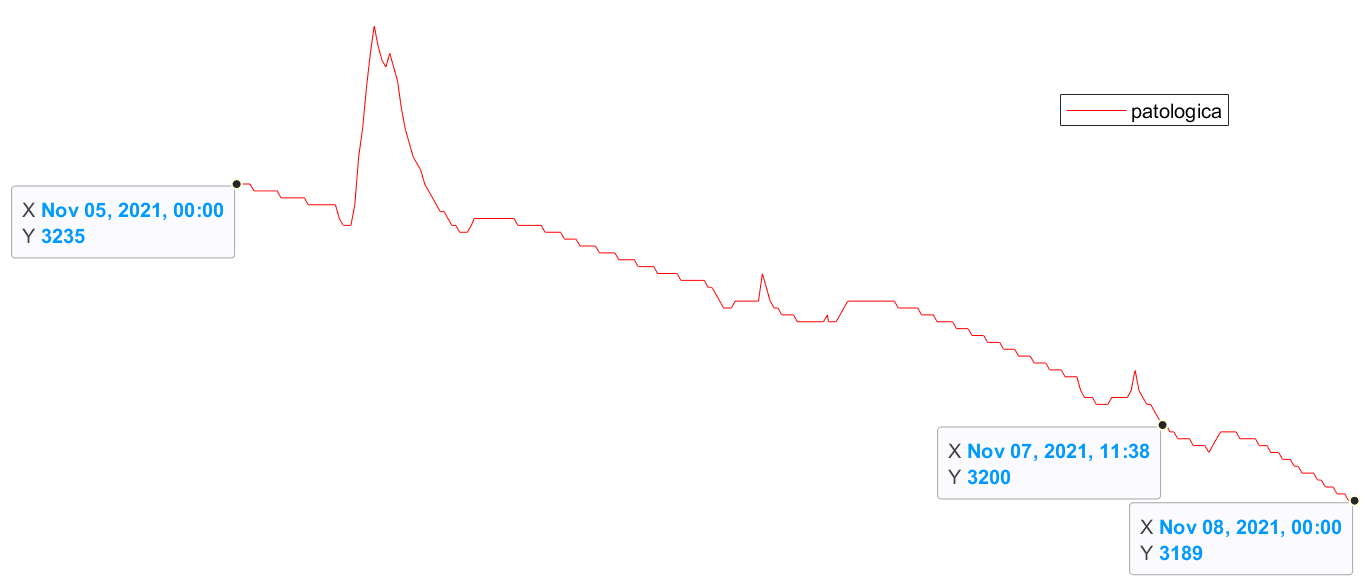


Figura V‑43: Terza sequenza

Come vediamo in Figura V‑43 l’evento patologico avviene il 7 novembre, ma la sequenza contiene anche il 5 e il 6 novembre. Il 5 e il 6 novembre il valore della tensione della cella minima è superiore alla soglia critica e quindi, presi singolarmente, questi giorni potrebbero far parte di una sequenza sana. Ciò si verifica sia in Figura V‑42 dove entrambi i giorni sono compresi, sia in Figura V‑41 ove è compreso solamente il 5 novembre. Al cobtrario, il 7 novembre la tensione scende sotto la soglia e quindi è il giorno effettivo che contiene l’evento critico, motivo per cui apparterrà solo a sequenze patologiche.

In Figura V‑44 sintetizziamo il concetto:

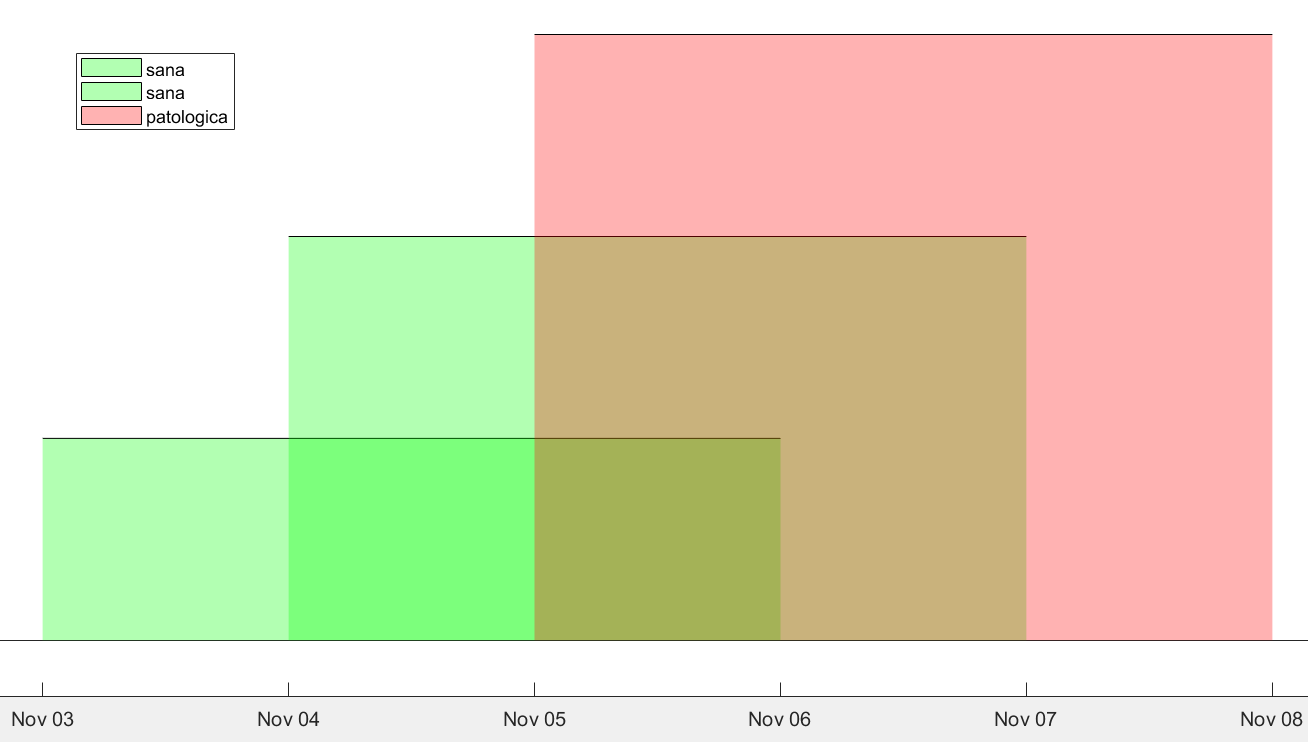


Figura V‑44: Sfalsamento delle sequenze

Dalla figura vediamo come il 5 novembre sia compreso in tutte e tre le sequenze. Ciò è vero per tutti i giorni dell’intervallo salvo quelli agli estremi, dato che le sequenze consecutive sono sfalsate di un giorno

In sintesi, il sistema di identificazione degli eventi di guasto non si base sul giorno dell’evento ma bensì sulla sequenza di appartenenza. Ogni sequenza che contiene un valore inferiore alla soglia viene definita automaticamente come patologica mentre le altre sono sane.

### Normalizzazione delle sequenze

La normalizzazione delle sequenze è lo step successivo all’ Identificazione delle sequenze sane e patologiche, come vediamo dalla routine nel main:



Figura V‑45: Normalizzazione

Di seguito vediamo il codice della funzione “normalizzazione”:

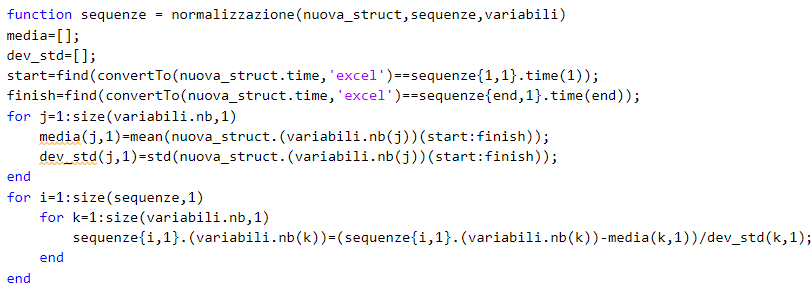


Figura V‑46: Funzione di normalizzazione

La funzione prende come argomento il dato sincronizzato, tutte le sequenze, il nome delle variabili presenti nelle sequenze e ritorna le sequenze normalizzate. La normalizzazione delle sequenze viene effettuata sottraendo ad ogni campione la media della variabile e dividendo per la deviazione standard. Sia la media che la deviazione standard sono effettuate sul dato sincronizzato che contiene la totalità dell’intervallo di tempo.

Di seguito il grafico della tensione della cella minima normalizzata:

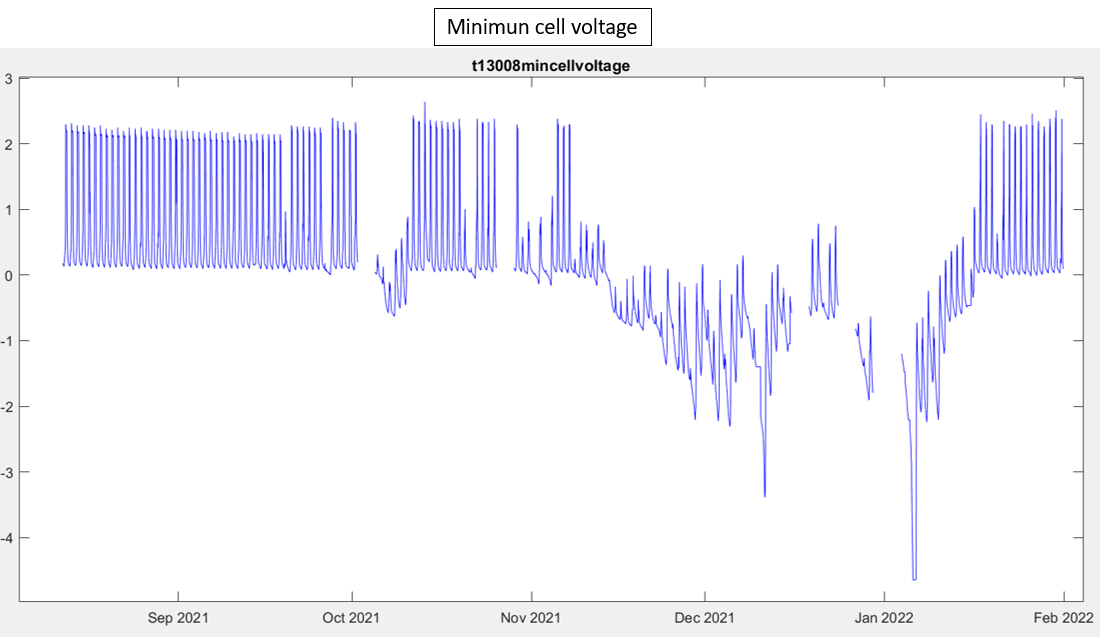


Figura V‑47: Andamento della tensione della cella minima normalizzata

Vediamo che rispetto a Figura V‑35 l’andamento rimane lo stesso, mentre vediamo il valore va da un massimo approssimativo di 2 fino ad un minimo approssimativo si -4,5. Normalizzando il dato lo rendiamo adimensionale, per questo motivo questa operazione viene fatta dopo la selezione delle sequenze tra sane e patologiche. Se l’avessimo fatta prima non avremmo potuto far riferimento alla soglia di 3,2 volt per discriminare tra le sequenze.

### Etichettatura delle sequenze

In questa fase andiamo a creare degli array che formeranno il dataset e quindi ad assegnare le etichette alle sequenze:

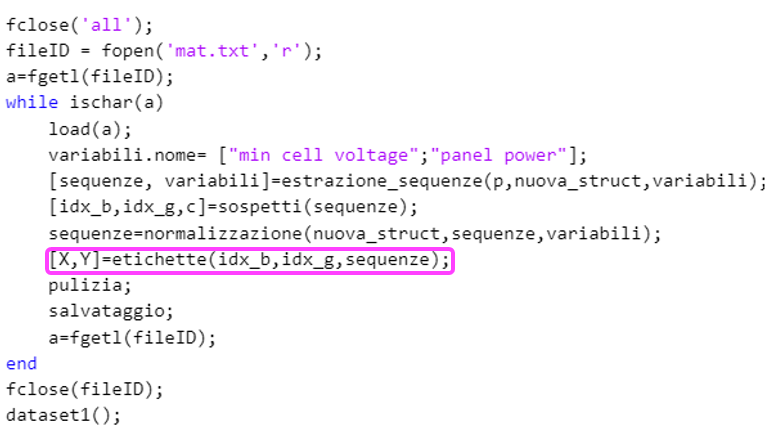


Figura V‑48: Funzione "etichette" nel main

La funzione prende in ingresso le sequenze, gli indici delle sequenze predittive associate a quelle sane e gli indici delle sequenze predittive associate a quelle patologiche. La funzione ritorna il dataset relativo ad un solo traliccio (“X”) e il relativo array categorico contenente le etichette (“Y”).

A seguire il corpo della funzione:

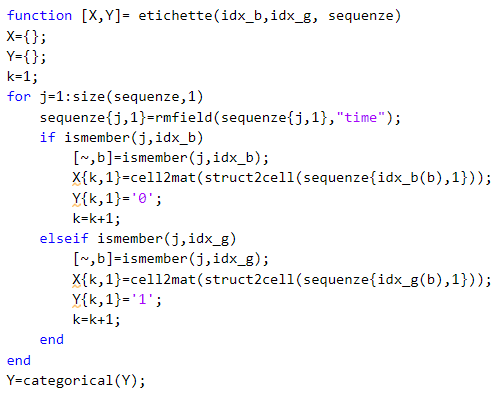


Figura V‑49: Etichettatura del dataset

Per ogni sequenza viene cancellato il campo relativo al tempo dato che non viene preso in ingresso dall’algoritmo di classificazione. Assegniamo le etichette ad ogni sequenza. Ogni sequenza viene salvata non più in una struct ma in un cell array (“X”). L’array delle etichette è un array categorico (“Y”), ossia un array di char.

Le etichette sono:

* ‘1’: la sequenza è sana
* ‘0’: la sequenza è patologica

Una volta che la funzione finisce la propria esecuzione otteniamo un insieme di sequenze relative ad una sola torre. Come detto in precedenza la procedura viene ripetuta per ogni traliccio.

### Salvataggio

Per ogni traliccio le sequenze “XTr”, “XTs”, “YTr” e “YTs” vengono salvate in una cartella dedicata avente il nome del traliccio.

### Creazione del dataset con le sequenze di tutti i tralicci

Le operazioni descritte nei paragrafi precedenti vengono ripetute per ogni traliccio. In questo modo otteniamo per ogni traliccio un dataset. Ora ci troviamo al di fuori del while:

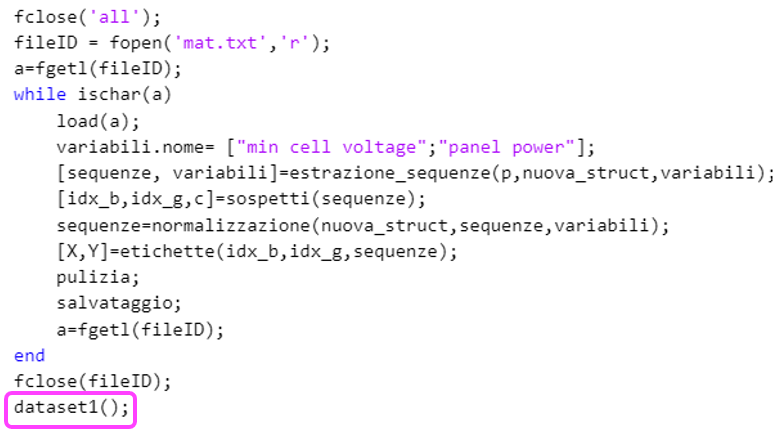


Figura V‑50: Funzione per la creazione del dataset nel main

In questa fase andiamo ad unire tutti i singoli dataset in un unico dataset. Andando a mischiare le sequenze non potremmo più risalire al traliccio d’appartenenza di una data sequenza.

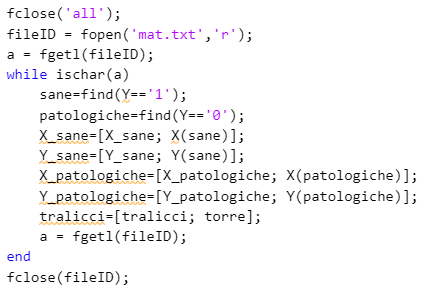


Figura V‑51: Creazione del dataset

Nel file di testo “mat.txt” troviamo le locazioni dei workspace dei singoli tralicci. Facciamo un ciclo while che legge una riga alla volta e apre il workspace corrispondente. Andiamo a trovare gli indici delle sequenze sane e patologiche per ogni traliccio e li usiamo per salvare le sequenze sane in “X\_sane”, “Y\_sane” e “X\_patologiche” e “Y\_patologiche”. Al termine di quest’operazione abbiamo unito le sequenze dei vari tralicci e suddivise tra sane e patologiche.

Andiamo ora a distinguere i casi di partizione statica e dinamica.

#### Partizione statica

Qualora volessimo una partizione statica andiamo a dividere manualmente i dataset tra Training e Test set. La nostra scelta di progetto è indicata dalla variabile globale “rapporto” che indica che il test set è il 25% del dataset, e quindi il training set è il 75%. Tramite la funzione “randsample” e la variabile “rapporto” estraiamo il 25% delle sequenze sane e patologiche e le assegniamo al test set. Il resto delle sequenze invece farà parte del train set.

La procedura è la seguente:

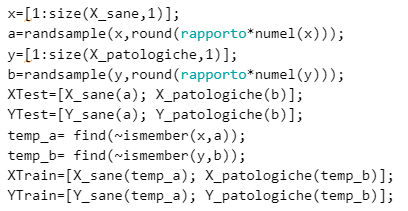


Figura V‑52: Partizione statica

Il dataset statico sarà dunque contenuto in “XTrain”, “YTrain”, ”XTest” e ”YTest”.

Andiamo a graficare le sequenze sane e patologiche comprese nel dataset finale nel caso in cui abbiamo un intervallo di predizione a 7 giorni, una durata delle sequenze di 3 giorni e uno sfalsamento tra sequenze di 1 giorno:

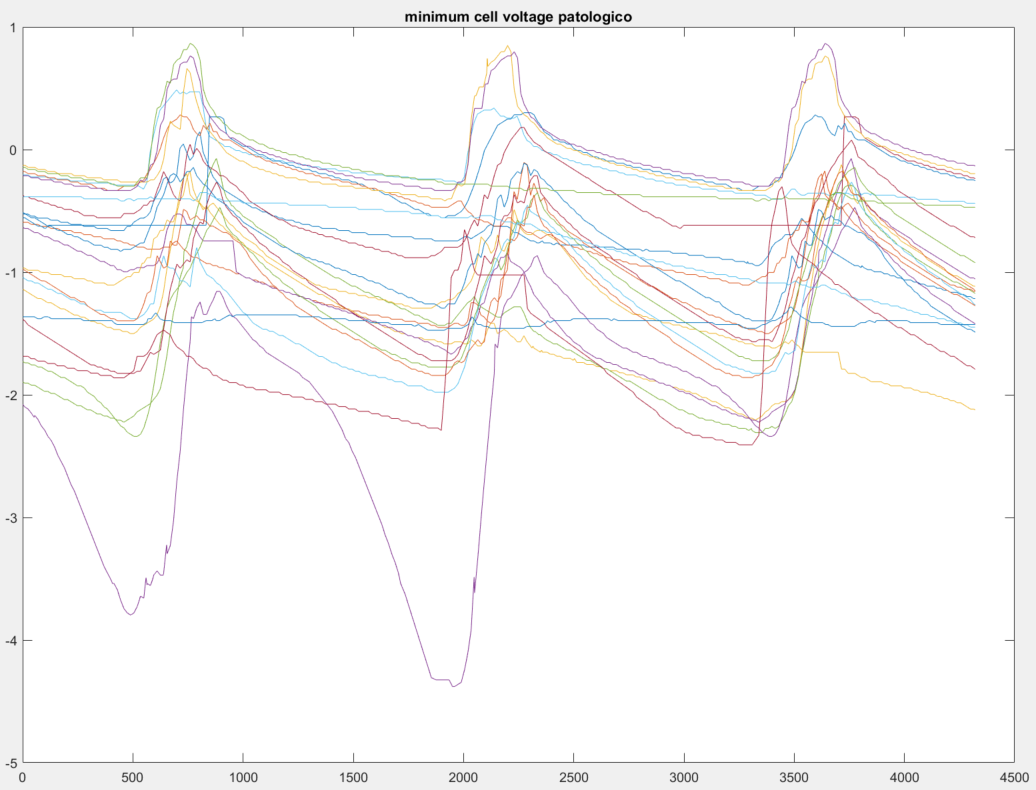


Figura V‑53: Sequenze patologiche della tensione della cella minima

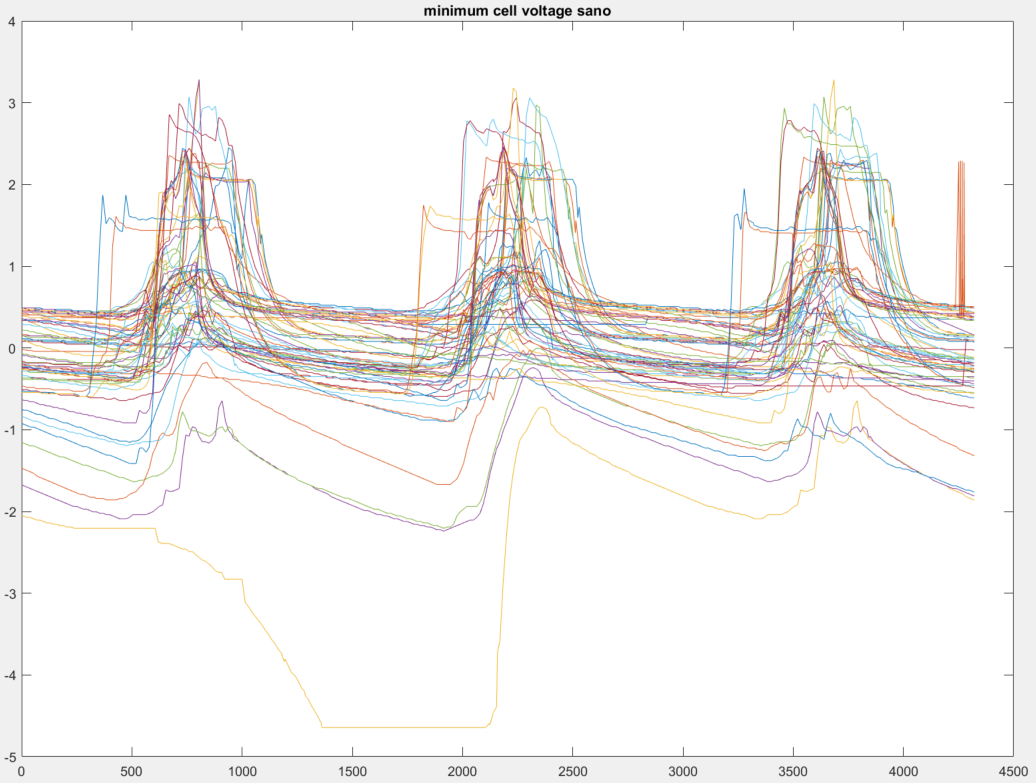


Figura V‑54: Sequenze sane della tensione della cella minima

In Figura V‑53 e Figura V‑54 possiamo osservare l’andamento della tensione della cella minima normalizzata nelle sequenze passate al dataset finale. In Figura V‑53 vediamo le sequenze patologiche, mentre in Figura V‑54 vediamo le sequenze sane. Possiamo notare che le sequenze sane incrociano le ordinate soprattutto nell’intorno dello zero e hanno un andamento ciclico che va da 0 ad un massimo di 3. Dato che le sequenze sono normalizzate, il valor medio globale è in 0, e quindi il fatto che le sequenze buone stiano globalmente sopra lo 0 è coerente con la definizione di sequenza sana. Notiamo anche qualche curva che può arrivare anche oltre -4, questo perché stiamo facendo una previsione a 7 giorni e quindi può accadere che una data sequenza sana sia stata preceduta 7 giorni prima da un evento patologico. Questo fenomeno tende a presentarsi maggiormente con l’aumentare dell’intervallo di previsione. Notiamo comunque che l’andamento della curva in Figura V‑54 ci permette di individuare a occhio i tre giorni contenuti nelle sequenze e quindi ci permette di dedurre che la regolarità delle sequenze sia indice di qualità per le sequenze.

Andando invece a vedere Figura V‑53 vediamo che tale regolarità non si riscontra e che la densità di curve non è concentrata in un punto ma sia mediamente costante tra 0 e -2. Globalmente vediamo che le sequenze sono comprese tra 1 e -4.

Andiamo ora ad osservare le sequenze sia patologiche che sane corrispondenti alla potenza del pannello solare:

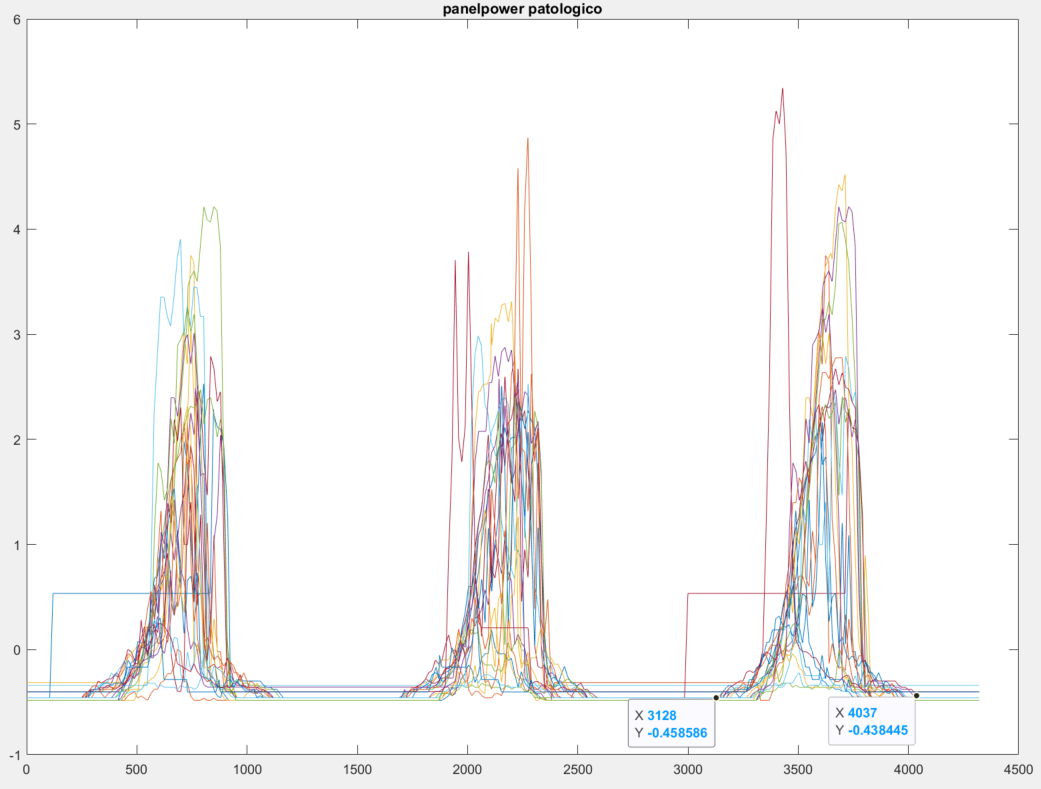


Figura V‑55: Sequenze patologiche della potenza del pannello

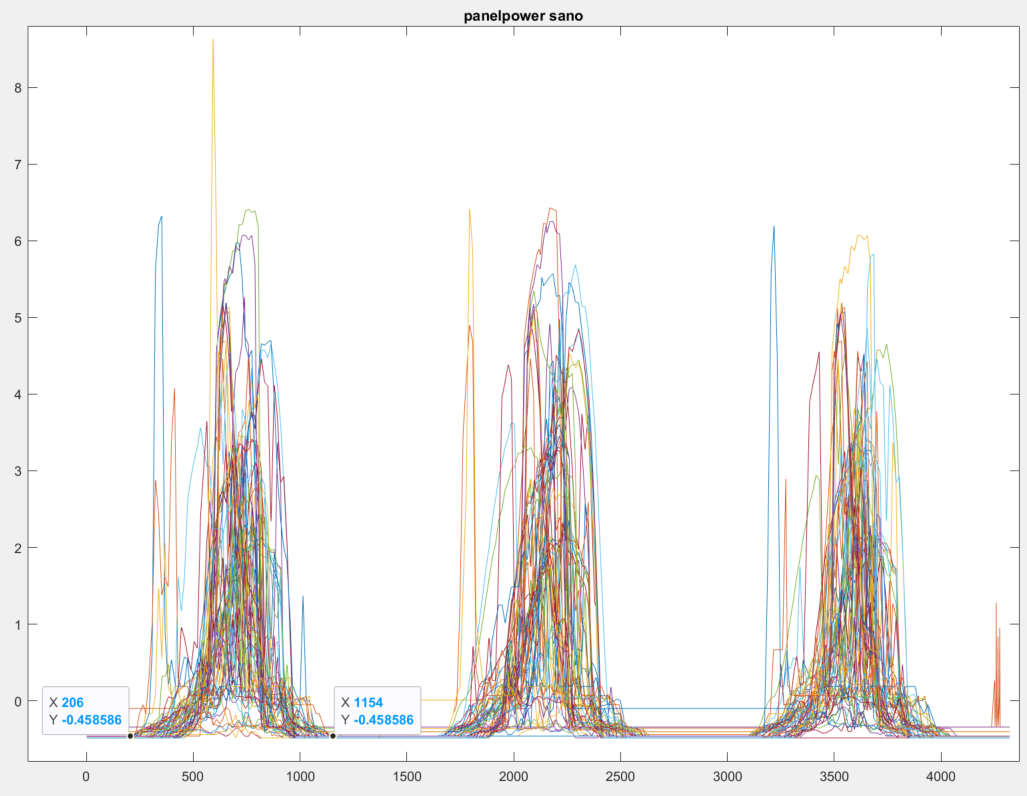


Figura V‑56: Sequenze sane della potenza del pannello

In Figura V‑55 vediamo che il valore del panel power va dall’intorno di 0 fino ad un picco di 7, restando mediamente tra 0 e 4. In Figura V‑56 l’ampiezza va dall’intorno di 0 fino ad un massimo di 10, restando mediamente tra 0 e 6. Vediamo dunque che l’ampiezza è leggermente maggiore in media per le sequenze sane. Ciononostante, la differenza in termine di ampiezza non è molto marcata. Una differenza che si nota maggiormente è la larghezza dei picchi. Per le sequenze patologiche abbiamo una larghezza massima di 600 punti, mentre per quelle sane di 1000 punti. Inoltre, in Figura V‑56 vediamo che il lobo sembra essere più ripido sin da subito, mentre in Figura V‑55 sembra avere una pendenza meno ripida ai margini.

A seguire vediamo l’andamento del SOC:

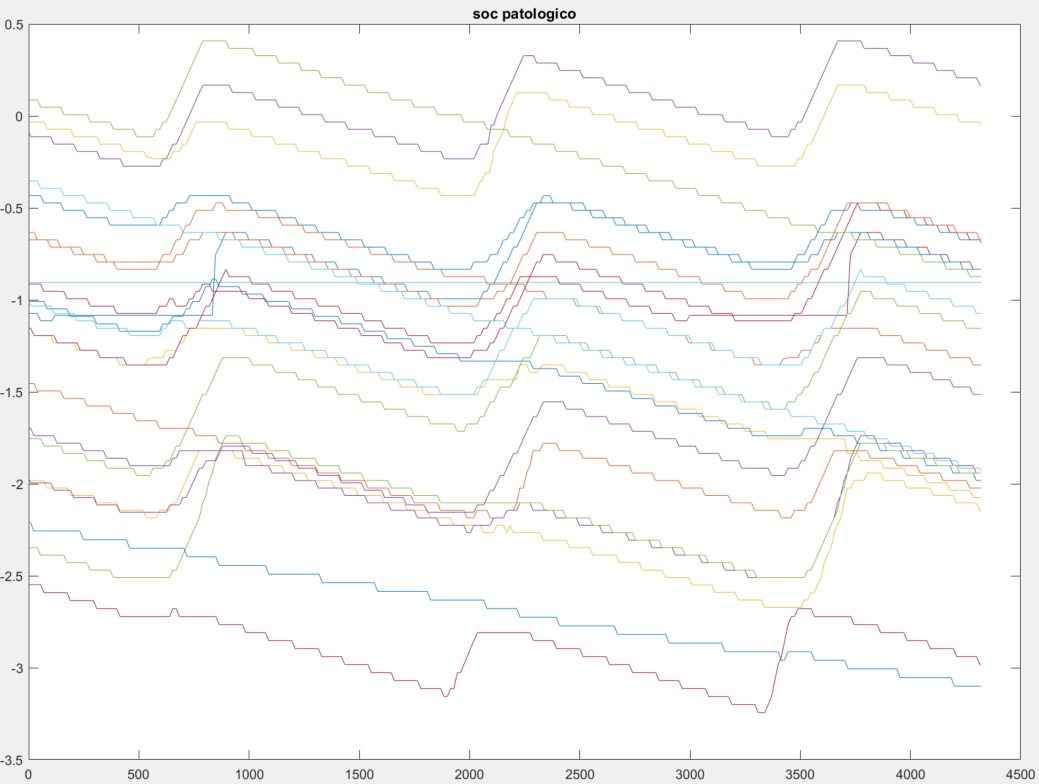


Figura V‑57: Sequenze patologiche del SOC

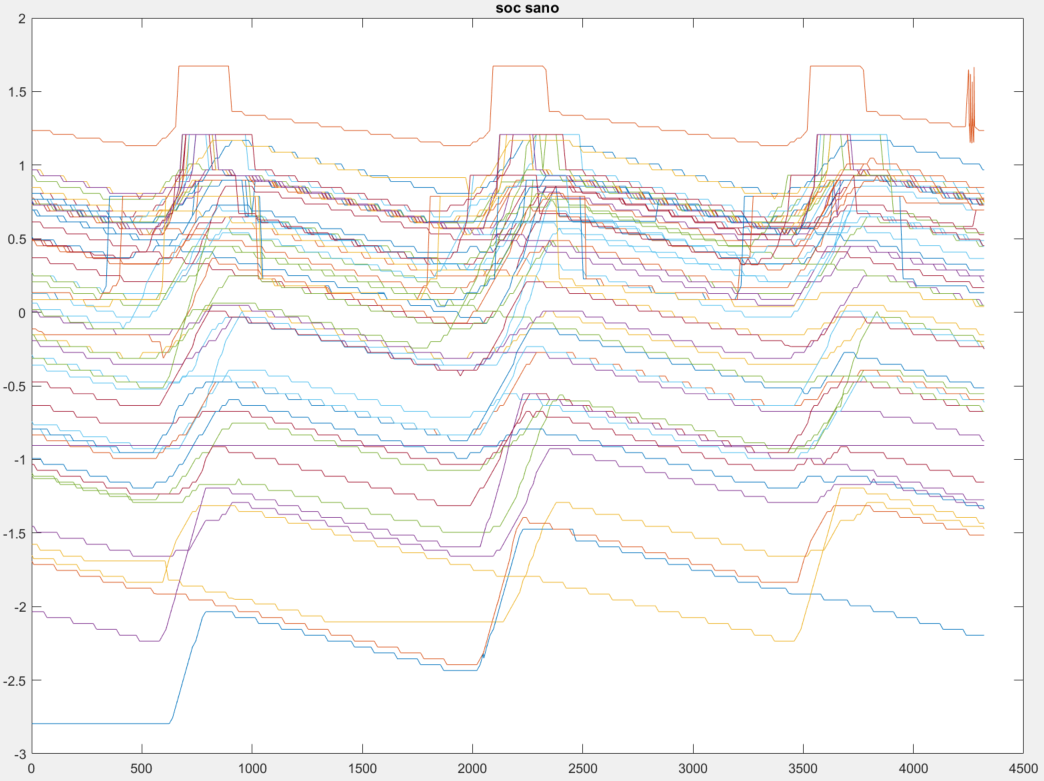


Figura V‑58: Sequenze sane del SOC

Similmente alla tensione della cella minima (Figura V‑53, Figura V‑54) le sequenze sane incrociano le ordinate soprattutto nell’intorno di un punto, in questo caso 0,5, e hanno mediamente un andamento ciclico che va da 0 ad 1. Osserviamo però che rispetto a Figura V‑54 è maggiore il numero di curve fuori da questo range è maggior, con dei minimi a -2 e dei massimi a 1,5. In Figura V‑57 le sequenze patologiche incrociano le ordinate in un intervallo che va 0,5 a -2,5 in modo omogeneo. Notiamo che rispetto a Figura V‑58, si possono ugualmente individuare i 3 giorni nelle sequenze, anche se meno facilmente. Inoltre, osserviamo che il SOC ha un andamento decrescente in Figura V‑57.

#### Partizione dinamica

Dato che vogliamo implementare un K-folding andiamo a salvare un dataset non partizionato nel seguente modo:



Figura V‑59: Partizione dinamica

In questo modo la partizione verrà fatta dinamicamente dalla routine di K-folding. Per questo motivo salviamo “X” e ”Y” che contengono la totalità delle sequenze del dataset.

### Salvataggio del dataset

Per differenziare i dataset generati li salviamo come mostrato in figura:

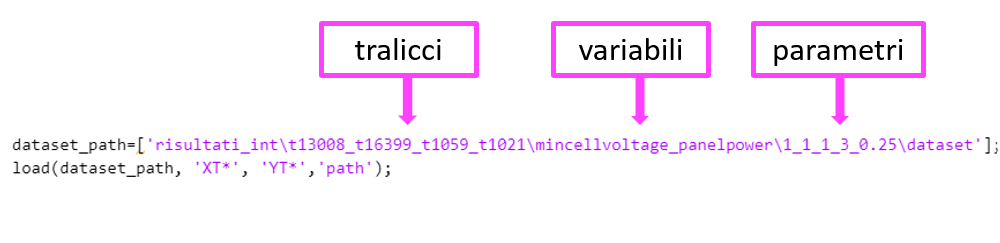


Figura V‑60: Pocedura di selezione del dataset

La prima cartella dentro “risultati\_int” identifica i tralicci usati nel dataset. La seconda sono le variabili usate. La terza rappresenta i parametri impostati nel seguente ordine:

* Durata delle sequenze in giorni
* Sfalsamento tra le sequenze in giorni
* Intervallo di previsione
* Proporzione tra sequenze sane e patologiche nel dataset
* Rapporto tra test e train set

## Estrazione di sequenze lunghe 1 giorno

Andiamo a vedere il caso in cui le sequenze abbiano una durata di 1 solo giorno invece che 3. Questo caso è rilevante poiché essendo lo sfalsamento tra una sequenza e l’altra di 1 giorno non ci sarebbe più sfalsamento tra una sequenza e l’altra (vedi Figura V‑41, Figura V‑42, Figura V‑43 e Figura V‑44). Per di più, dato che una sequenza viene detta patologica anche quando un solo campione della tensione della cella minima è sotto la soglia di 3,2V, prendendo sequenze di lunghezza minore (1 giorno) otteniamo sequenze con maggiore granularità rispetto all’evento. Per implementare questa modifica ci basta cambiare il valore della variabile globale “lasso” da 3 ad 1 giorno come da figura:

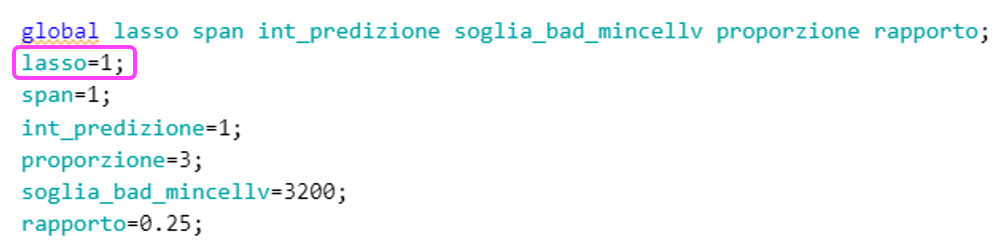


Figura V‑61: Caso di sequenze di durata 1 giorno

A seguire vediamo come le sequenze positive e negative non si sovrappongono più:

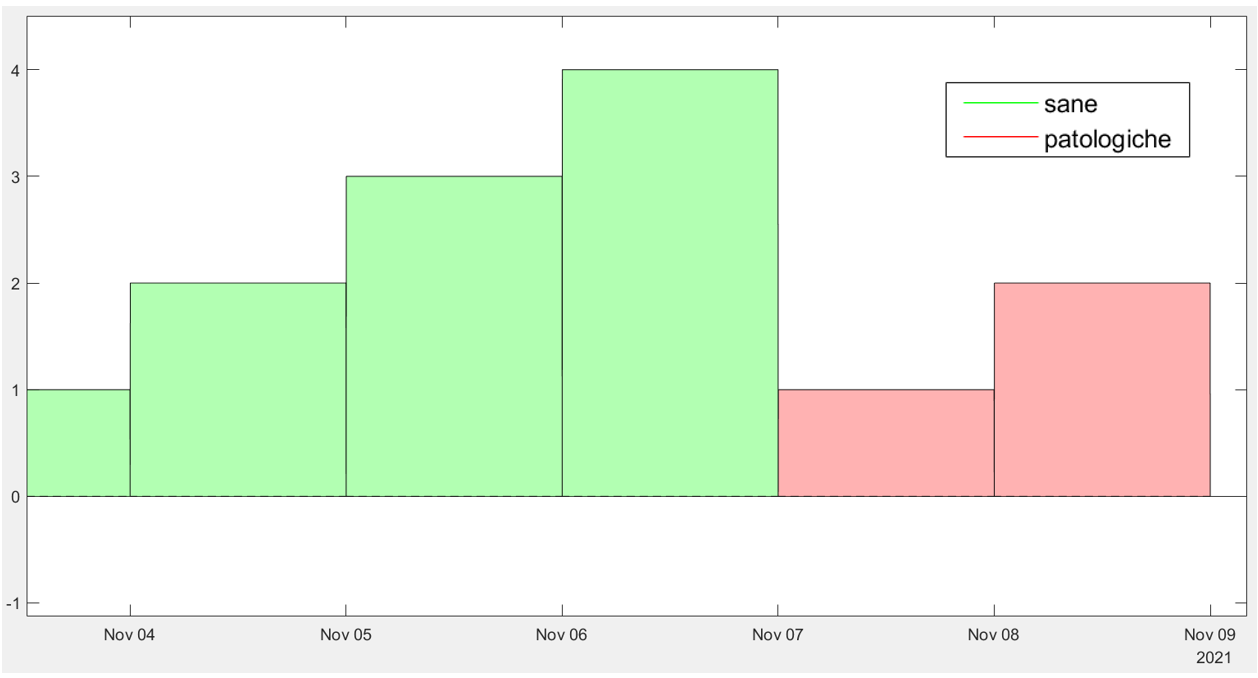


Figura V‑62: Rappresentazione delle sequenze senza sovrapposizione

Nel capitolo riguardante i risultati andremmo a vedere come i risultati possano variare in base alla lunghezza in giorni delle sequenze.

# Rete Neurale

## Layers

Vogliamo creare una rete LSTM bidirezionale:

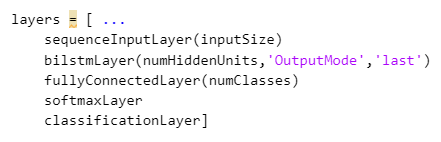


Figura VI‑1: Layers della rete neurale

Dato che abbiamo a disposizione l’interezza delle sequenze usiamo un layer BiLSTM per estrarre le correlazioni sia con gli istanti precedenti che con quelli successivi. Il primo argomento è il numero di hidden layer mentre il secondo riguarda ciò che ritorna la rete. In questo caso in uscita presenta solo l’ultimo istante di tempo della sequenza. Dopo la rete BiLSTM mettiamo un layer fully connected con una funzione d’attivazione non lineare softmax. Dato che la classificazione è binaria in uscita avremmo 1 neurone. L’ultimo layer è un classification layer specifico per i problemi di classificazione che calcola la funzione di cross entropia di perdita grazie alla quale vengono ottimizzati i parametri della rete.

## Parametri

Dopo aver caricato il dataset andiamo ad impostare i parametri dell’algoritmo:

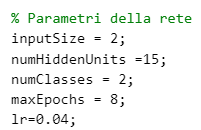


Figura VI‑2: Parametri della rete

Il parametro “inputSize” rappresenta il numero di variabili che passiamo alla rete e “numClasses” è il numero di valori che può avere l’etichetta della sequenza; in questo caso dato che facciamo una classificazione binaria è sempre uguale a 2. Dopo diverse prove scegliamo un numero massimo di epoche uguale ad 8 ed un learning rate iniziale di 0.04. Ponendo un maggiore numero di epoche troviamo che la curva dell’accuratezza assume un andamento periodico e manda in overfitting il modello offrendo dei risultati di accuratezza peggiori. Per quanto riguarda il learning rate, scegliendone uno minore ottenevamo lo stesso risultato ma in più epoche, oppure ottenevamo risultati peggiori. Il numero di hidden unit è stato scelto uguale a 15 poiché all’aumentare otteniamo sempre gli stessi risultati in termini di accuratezza ma con un tempo computazionale maggiore.

## Training Options

Andiamo ora ad impostare le opzioni della rete:

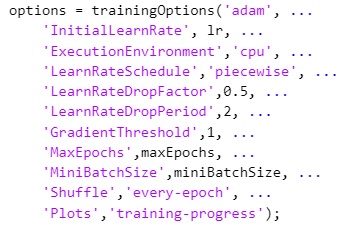


Figura VI‑3: Opzioni della rete

Il solver che andiamo ad utilizzare è ‘Adam’. Scegliamo di usare un fattore di decrescimento del learning rate inziale pari al 50 % ogni 2 epoche di modo che all’avanzare del training l’apprendimento scali per maggiore precisione. Per evitare il l’esplosione del gradiente andiamo a scegliere 1 come valore massimo del gradiente. Le dimensioni del minibatch varieranno in base alle dimensioni del dataset tenendo in conto che vogliamo mantenere il minibatch di una dimensione che sia un divisore delle dimensioni dei set (Training e Test). Solitamente varia tra 10 e 30. Infine, scegliamo di impostare uno shuffle ad ogni epoca poiché le sequenze tra loro non presentano legami di consequenzialità. Abilitando questa opzione vogliamo evitare che il modello resti in un minimo locale non raggiungendo un minimo globale della funzione di perdita. Inoltre, nel caso in cui il minibatch size non sia un divisore della dimensione del set, evitiamo che siano sempre le stesse sequenze ad essere escluse dal set.

Dato che le sequenze hanno tutte la stessa lunghezza non introduciamo né padding né sorting. A seguire il grafico relativo alla lunghezza delle sequenze:

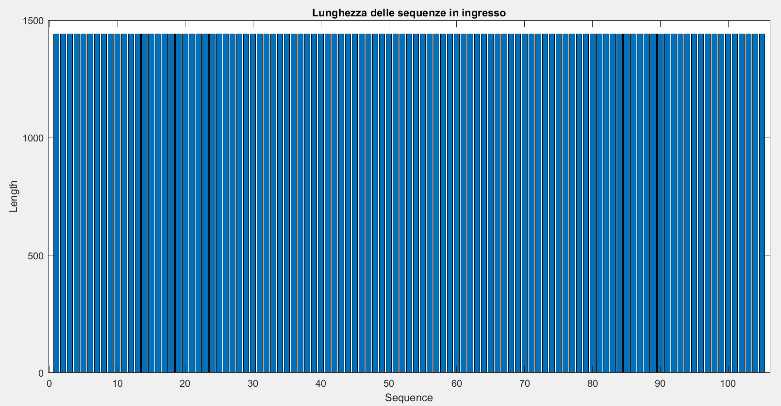


Figura VI‑4: Lunghezza delle sequenze uniforme

## Training e classificazione

Dopo aver impostato i parametri possiamo avviare il training e la classificazione tramite i seguenti comandi:

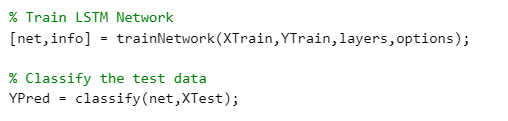


Figura VI‑5: Ttraining e classificazione

In ingresso alla rete di training diamo il training set composto da XTrain (le sequenze) e YTrain (le etichette), i layers e le opzioni impostate (Figura VI‑1, Figura VI‑2, Figura VI‑3). La funzione trainNetwork ritorna il classificatore che poi useremo per classificare il test set. Dopo la classificazione la funzione ‘classify’ ritorna le etichette predette dal classificatore che in seguito andremo a confrontare con quelle di riferimento per calcolare l’accuratezza.

A seguire un andamento dell’accuratezza e della funzione di perdita con le suddette opzioni della rete:

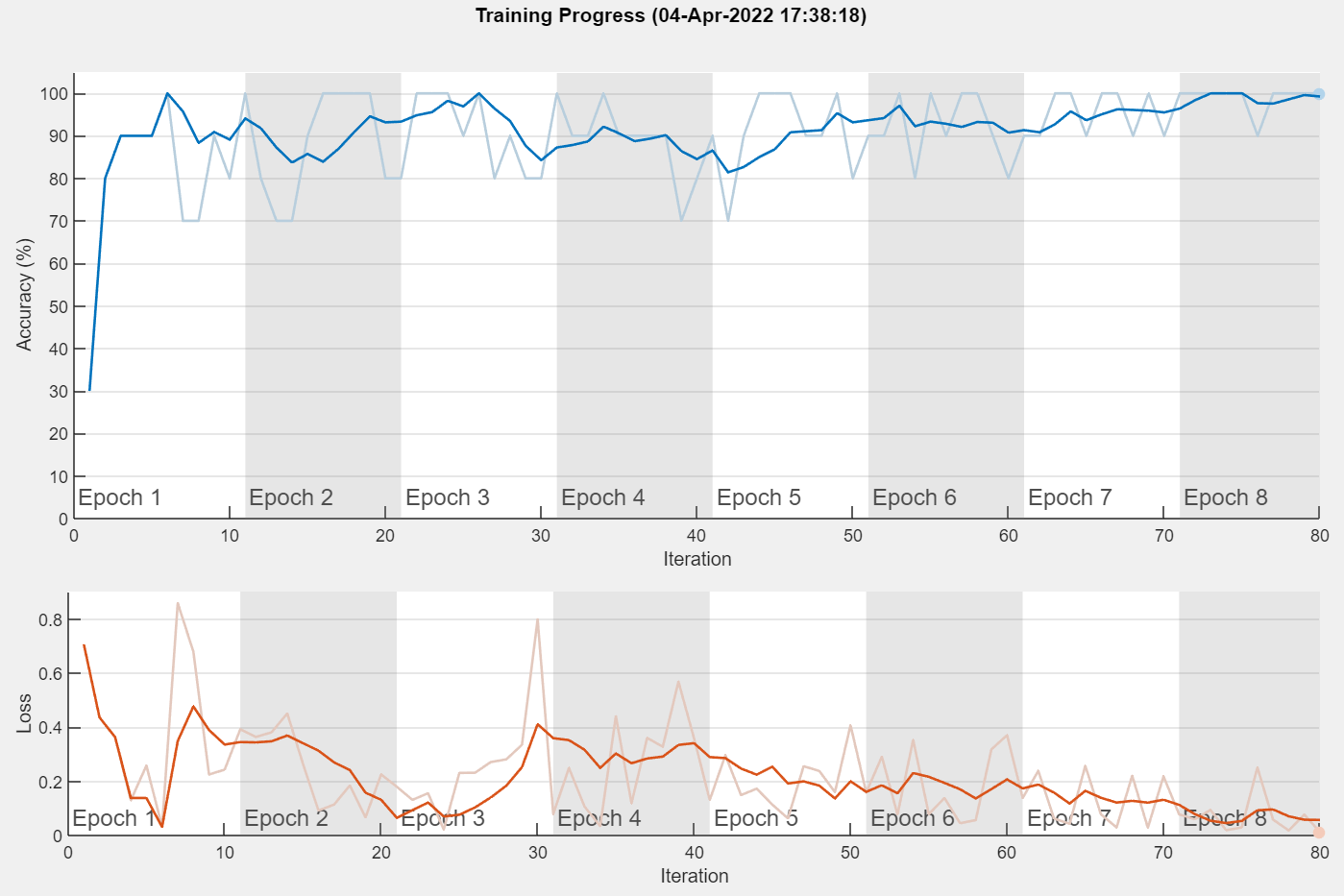


Figura VI‑6: Andamento tipico dell'accuratezza e della funzione di perdita con le suddette opzioni della rete

Come vediamo in Figura VI‑6 dopo 5/6 epoche l’andamento dell’accuratezza (curva blu) tende a stabilizzarsi sopra il 90% e la loss (curva rossa) rimane sotto lo 0.2.

Una volta effettuata la classificazione calcoliamo l’accuratezza e grafichiamo la confusion chart con il seguente codice:

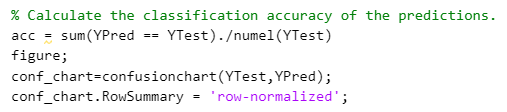


Figura VI‑7: Calcola dell'accuratezza e confusion chart

Come vediamo l’accuratezza è il numero di elementi di ‘YPred’ che coincidono con ‘YTest’ fratto il numero di elementi di ‘YTest’.

La confusion chart avrà la seguente raffigurazione:

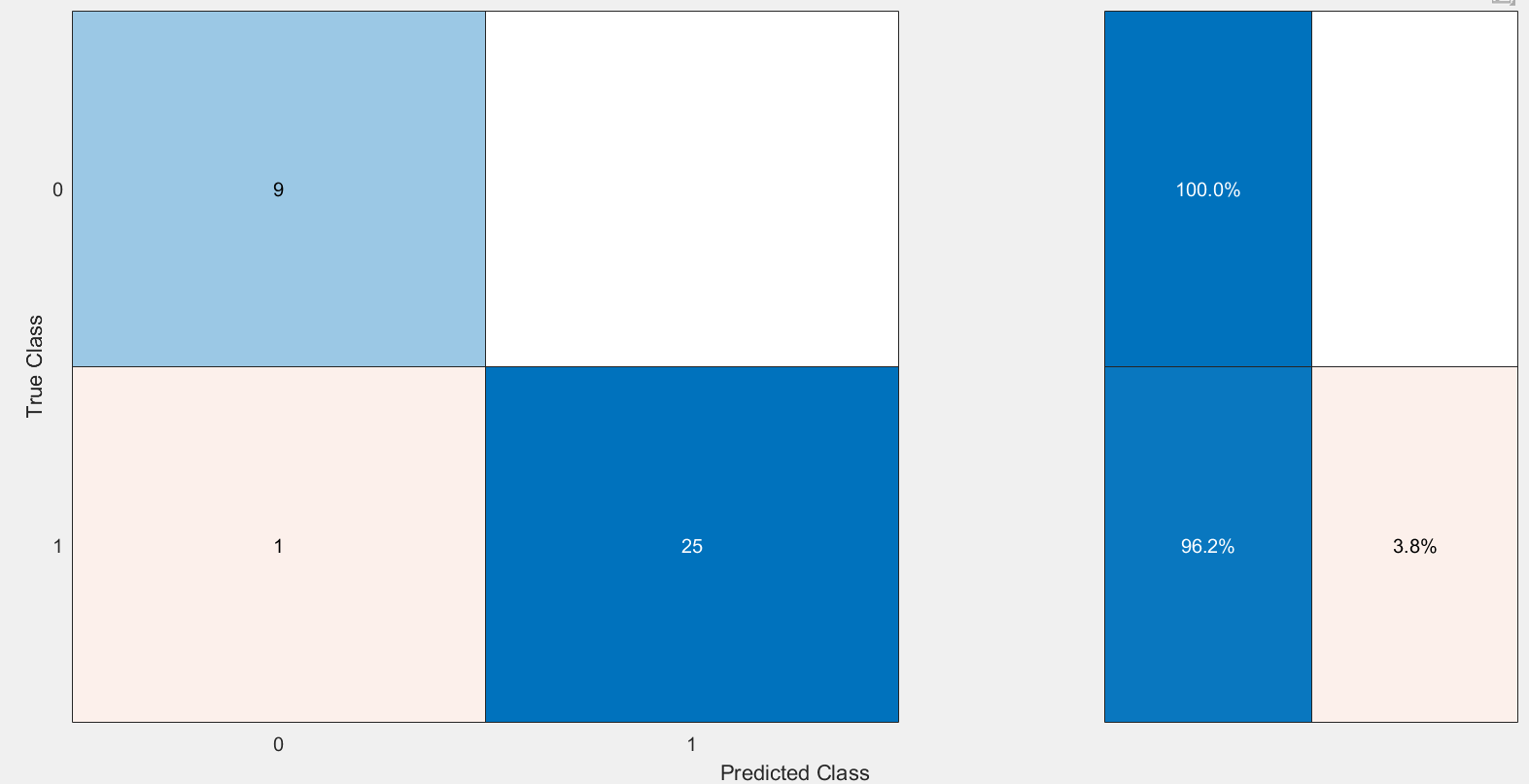


Figura VI‑8: Confusion chart

La confusion chart rappresenta sulla diagonale principale gli elementi correttamente classificati mentre in (1,0) troviamo i falsi negativi (gli elementi classificati come patologici dal modello ma che sono sani) e in (0,1) i falsi positivi (viceversa). Nel riquadro di destra invece troviamo per ogni classe le percentuali di elementi classificati correttamente e la percentuale di quelli classificati in modo errato rispetto alla classe.

## K-Folding

Al fine di aumentare l’affidabilità dei risultati che otteniamo scegliamo di implementare un k-folding. In particolare, nel nostro caso scegliamo di dividere il set in 4 fold e di ripetere la procedura 10 volte. In questo modo speriamo di ottenere un risultato di accuratezza che rispecchi il più possibile le capacità di apprendimento del modello con i dati a disposizione.

L’algoritmo di implementazione del k-folding è il seguente:

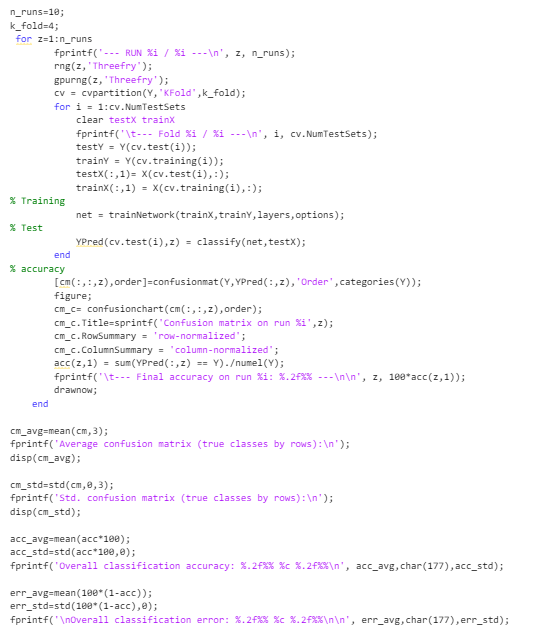


Figura VI‑9: Algoritmo d K-Folding

Tramite la funzione ‘cvpartition’ creiamo 4 fold che verranno usati a rotazione per formare il training set e il test set. In particolare, 3 fold formeranno il training set e 1 il test set. Da notare che la funzione mantiene automaticamente la proporzione presente tra le etichette dal dataset. Facciamo un primo ciclo for sul numero di volte che vogliamo ripetere la partizione del dataset. Per evitare che le partizioni estratte da ‘cvpartition’ siano le stesse impostiamo, ad ogni iterazione del ciclo for, il seed del generatore pseudo random uguale al numero dell’iterazione in corso. Dopodiché impostiamo un ciclo for annidato sul numero di fold di modo che ogni fold sia una sola volta il test set. Per ogni permutazione alleniamo la rete e facciamo la classificazione come nel capitolo precedente. Una volta creato un modello per ogni permutazione del dataset e fatta la classificazione, facciamo la confusion chart rispetto alla globalità del dataset e calcoliamo l’accuratezza come media delle accuratezze ottenute per le 4 permutazioni del dataset. Rispetto alla partizione statica abbiamo il vantaggio che sia la confusion chart che l’accuratezza siano rispetto a tutto il dataset e quindi siano più consistenti.

Alla fine del procedimento otteniamo 10 accuratezze di cui facciamo la media e la deviazione standard. Facciamo la stessa cosa per la confusion matrix e l’errore (1-acc).

# Risultati

Andiamo a fare multiple combinazioni di variabili per trovare quella che potrebbe portare ai migliori risultati. Durante tutte le prove manterremo la soglia critica tra sequenze sane e patologiche a 3200 [mV] e per la partizione statica un rapporto di sequenze di test uguale al 25% del dataset totale e 75% per il training. Non abbiamo implementato il validation set.

## Tensione della cella minima e potenza del pannello

Usiamo la combinazione di tensione della cella minima e potenza del pannello.

### Predizione ad 1 giorno con sequenze lunghe 3 giorni

Poniamoci nel caso in cui le sequenze durino 3 giorni, siano sfalsate di 1 giorno, e ci sia una proporzione di sequenze tale che per una sequenza patologica ce ne siano 3 sane. Vogliamo prevedere l’evento 1 giorno prima. La dimensione dei mini-batch è di 12 elementi con dei fold ognuno da 48 elementi, per un totale di 192 sequenze nel dataset.

L’accuratezza che otteniamo è la seguente:

Otteniamo un’accuratezza del 94% con 0,99% di deviazione standard. L’errore di classificazione è approssimativamente del 6%.

Per quanto riguarda la confusion matrix, le righe rappresentano le classi e sulle colonne troviamo le classi assegnate alle sequenze dal modello predittivo. La prima riga rappresenta la classe patologica e la seconda la classe sana:

Tabella 2: Confusion Matrix per predizioni con anticpo di 1 giorno e sequenze lunghe 3 giorni

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 43.4 ± 2.2 | 4.6 ± 2.2 |
| 1 | 6.8 ± 2.2 | 137.2 ± 2.2 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 2**Errore. L'origine riferimento non è stata trovata.** evidenzia che sulla totalità del dataset mediamente 6,8 elementi vengono classificati erroneamente come patologici e 4,6 elementi vengono classificati erroneamente come sani. Inoltre, la deviazione standard della confusion matrix ci dice la differenza di ogni elemento della confusion matrix tra una run e l’altra e quindi la variazione di prestazione in base alla scelta dei fold.

### Predizione ad 1 giorno con sequenze lunghe 1 giorno

Poniamoci nelle stesse condizioni del caso precedente ma con sequenze di durata 1 giorno. La dimensione dei mini-batch è di 7 elementi con dei fold ognuno da 35 elementi, per un totale di 140 sequenze nel dataset.

L’accuratezza che otteniamo è la seguente:

Otteniamo un’accuratezza del 97% con 1,36% di deviazione standard. L’errore di classificazione è approssimativamente del 3%.

La media e la deviazione standard delle confusion matrix sono le seguenti:

Tabella 3: Confusion matrix per predizioni con anticipo di 1 giorno e sequenze lunghe 1 giorno

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 34.6 ± 0.5 | 0.4 ± 0.5 |
| 1 | 4.0 ± 1.6 | 101.0 ± 1.6 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 3 evidenzia che sulla totalità del dataset mediamente 4 elementi vengono classificati erroneamente come patologici e 0.4 elementi vengono classificati erroneamente come sani. Inoltre, la deviazione standard è maggiore per la classificazione degli elementi sani.

Mediamente vediamo che con questa combinazione di variabili le sequenze lunghe 1 giorno sembrano risultare in migliori prestazioni per previsioni 1 giorno prima dell’evento.

### Predizione a 3 giorni con sequenze lunghe 3 giorni

In questo caso manteniamo gli stessi parametri ma impostiamo sequenze lunghe 3 giorni e intervallo di predizione uguale a 3 giorni.

La dimensione dei mini-batch è di 23 elementi con dei fold ognuno da 46 elementi, per un totale di 184 sequenze nel dataset.

L’accuratezza che otteniamo è la seguente:

Otteniamo un’accuratezza approssimativa del 91% con 0,99% di deviazione standard. L’errore di classificazione è approssimativamente del 9%.

La media e la deviazione standard delle confusion matrix sono le seguenti:

Tabella 4: Confusion matrix per predizioni con anticipo di 3 giorni e sequenze lunghe 3 giorni

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 39.8 ± 1.6 | 6.2 ± 1.6 |
| 1 | 9.8 ± 1.4 | 128.2 ± 1.4 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 4 evidenzia che sulla totalità del dataset mediamente 9,8 elementi vengono classificati erroneamente come patologici e 6.2 elementi vengono classificati erroneamente come sani. Inoltre, la deviazione standard è maggiore per la classificazione degli elementi patologici.

### Predizione a 3 giorni con sequenze lunghe 1 giorno

In questo caso manteniamo gli stessi parametri ma impostiamo sequenze lunghe 1 giorno.

La dimensione dei mini-batch è di 16 elementi con dei fold ognuno da 32 elementi, per un totale di 128 sequenze nel dataset. Da notare che il numero di sequenze è minore rispetto al caso precedente.

L’accuratezza che otteniamo è la seguente:

Otteniamo un’accuratezza approssimativa del 91% con 1,4% di deviazione standard. L’errore di classificazione è approssimativamente del 9%.

La media e la deviazione standard delle confusion matrix sono le seguenti:

Tabella 5: Confusion matrix per predizioni con anticipo di 3 giorni e sequenze lunghe 1 giorno

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 25.7 ± 2.1 | 6.3 ± 2.1 |
| 1 | 4.8 ± 1.8 | 91.2 ± 1.8 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 5: Confusion matrix per predizioni con anticipo di 3 giorni e sequenze lunghe 1 giornoTabella 5 evidenzia che sulla totalità del dataset mediamente 4,8 elementi vengono classificati erroneamente come patologici e 6,3 elementi vengono classificati erroneamente come sani. Inoltre, la deviazione standard è maggiore per la classificazione degli elementi patologici.

Mediamente vediamo che con questa combinazione di variabili le sequenze lunghe 1 giorno hanno una deviazione standard della confusion matrix e dell’accuratezza maggiore rispetto al caso di sequenze lunghe 3 giorni. I risultati medi invece sono approssimativamente uguali.

### Predizione a 7 giorni con sequenze lunghe 3 giorni

In questo caso manteniamo gli stessi parametri e impostiamo l’intervallo di predizione uguale a 7 giorni,

La dimensione dei mini-batch è di 23 elementi con dei fold ognuno da 46 elementi, per un totale di 184 sequenze nel dataset.

L’accuratezza che otteniamo è la seguente:

Otteniamo un’accuratezza approssimativa del 82% con 1,39% di deviazione standard. L’errore di classificazione è approssimativamente del 18%.

La media e la deviazione standard delle confusion matrix sono le seguenti:

Tabella 6: Confusion matrix per predizioni con anticipo di 7 giorni e sequenze lunghe 3 giorni

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 30.8 ± 1.7 | 15.2 ± 1.7 |
| 1 | 17.9 ± 2.9 | 120.1 ± 2.9 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 6 evidenzia che sulla totalità del dataset mediamente 17,9 elementi vengono classificati erroneamente come patologici e 15,2 elementi vengono classificati erroneamente come sani. Inoltre, la deviazione standard è maggiore per la classificazione degli elementi sani.

### Predizione a 7 giorni con sequenze lunghe 1 giorno

In questo caso manteniamo gli stessi parametri del paragrafo precedente impostando però la lunghezza delle sequenze di 1 giorno.

La dimensione dei mini-batch è di 10 elementi con dei fold ognuno da 31 elementi, per un totale di 124 sequenze nel dataset.

L’accuratezza che otteniamo è la seguente:

Otteniamo un’accuratezza approssimativa del 84% con 1,14% di deviazione standard. L’errore di classificazione è approssimativamente del 16%.

La media e la deviazione standard delle confusion matrix sono le seguenti:

Tabella 7: Confusion matrixper predizioni con anticipo di 7 giorni e sequenze lunghe 1 giorno

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 18.6 ± 1.1 | 12.4 ± 1.1 |
| 1 | 7.3 ± 1.3 | 85.7 ± 1.3 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 7 evidenzia che sulla totalità del dataset mediamente 7,3 elementi vengono classificati erroneamente come patologici e 12,4 elementi vengono classificati erroneamente come sani. Inoltre, la deviazione standard è maggiore per la classificazione degli elementi sani.

### Considerazioni

A seguire le accuratezze ottenute con la combinazione di tensione della cella minima e potenza del pannello:

Tabella 8: Tabella delle accuratezze ottenute con la tensione della cella minima e la potenza del pannello

|  |  |  |
| --- | --- | --- |
|  | Sequenze da 3 giorni | Sequenze da 1 giorno |
| Previsione ad 1 giorno | 94.06% ± 0.66% | 96.86% ± 1.36% |
| Previsione a 3 giorni | 91.30% ± 0.99% | 91.33% ± 1.44% |
| Previsione a 7 giorni | 82.01% ± 1.39% | 84.11% ± 1.14% |

A seguire un diagramma a barre che rappresenta le accuratezze in Tabella 8:

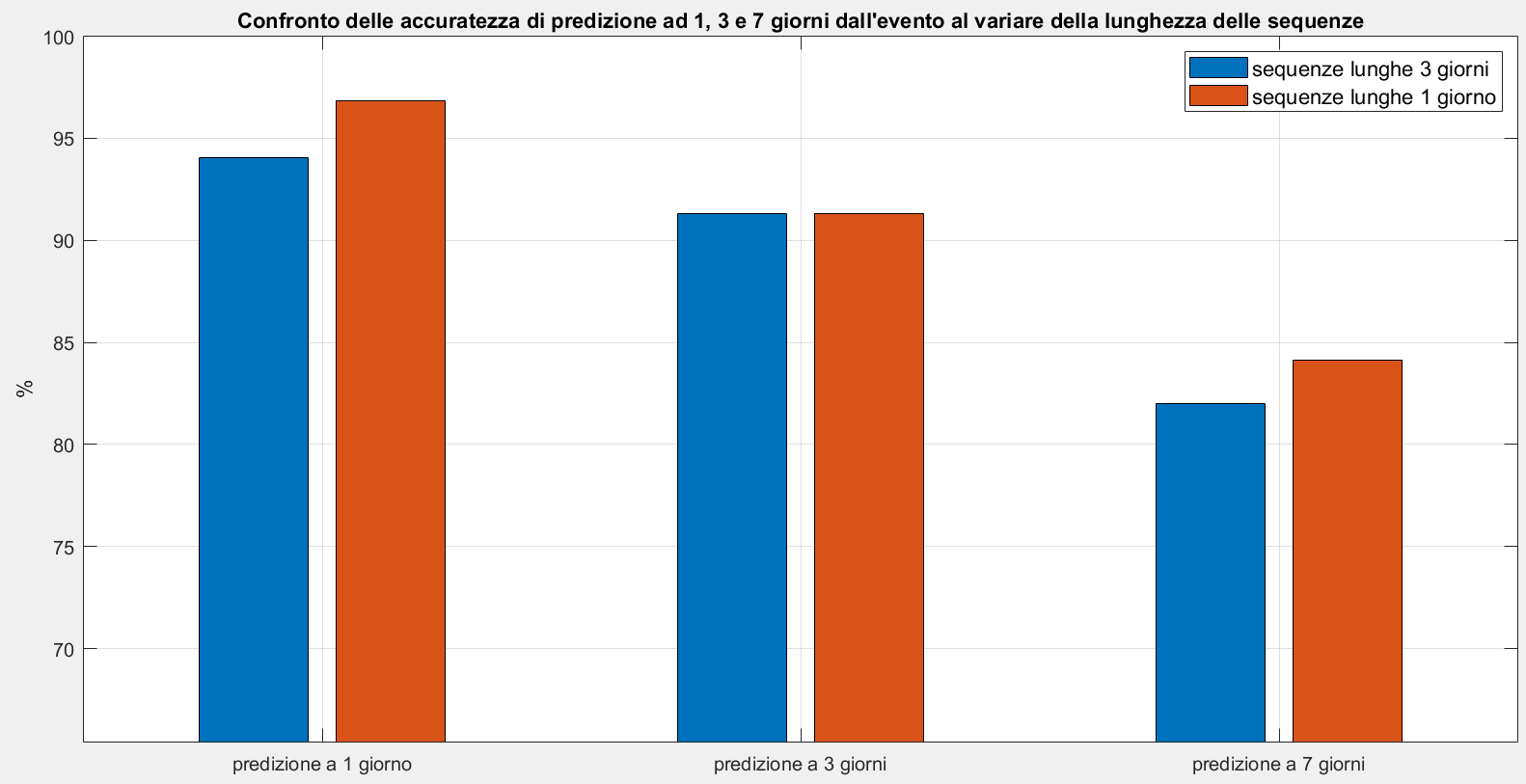


Figura VII‑1: Confronto delle accuratezza di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze

Osserviamo come l’accuratezza decresca a mano a mano che il numero di giorni dall’evento aumenta. Inoltre, notiamo che a 3 giorni dall’evento le accuratezze per il caso di sequenze lunghe 3 giorni ed 1 giorno si equivalgono, mentre negli altri casi l’accuratezza migliore si ottiene con delle sequenze lunghe 1 giorno. In questo caso, l’accuratezza migliore in assoluto è uguale al 96,86% ± 1,36% e viene ottenuta nel caso di predizione ad 1 giorno con sequenze lunghe 1 giorno. Il caso peggiore invece corrisponde a sequenze lunghe 3 giorni a distanza di 7 giorni dall’evento ed è 82,01% ± 1,39%.

Di seguito andiamo a graficare l’andamento dei falsi positivi (Figura VII‑2) e dei falsi negativi (Figura VII‑3) al variare dell’intervallo di predizione:

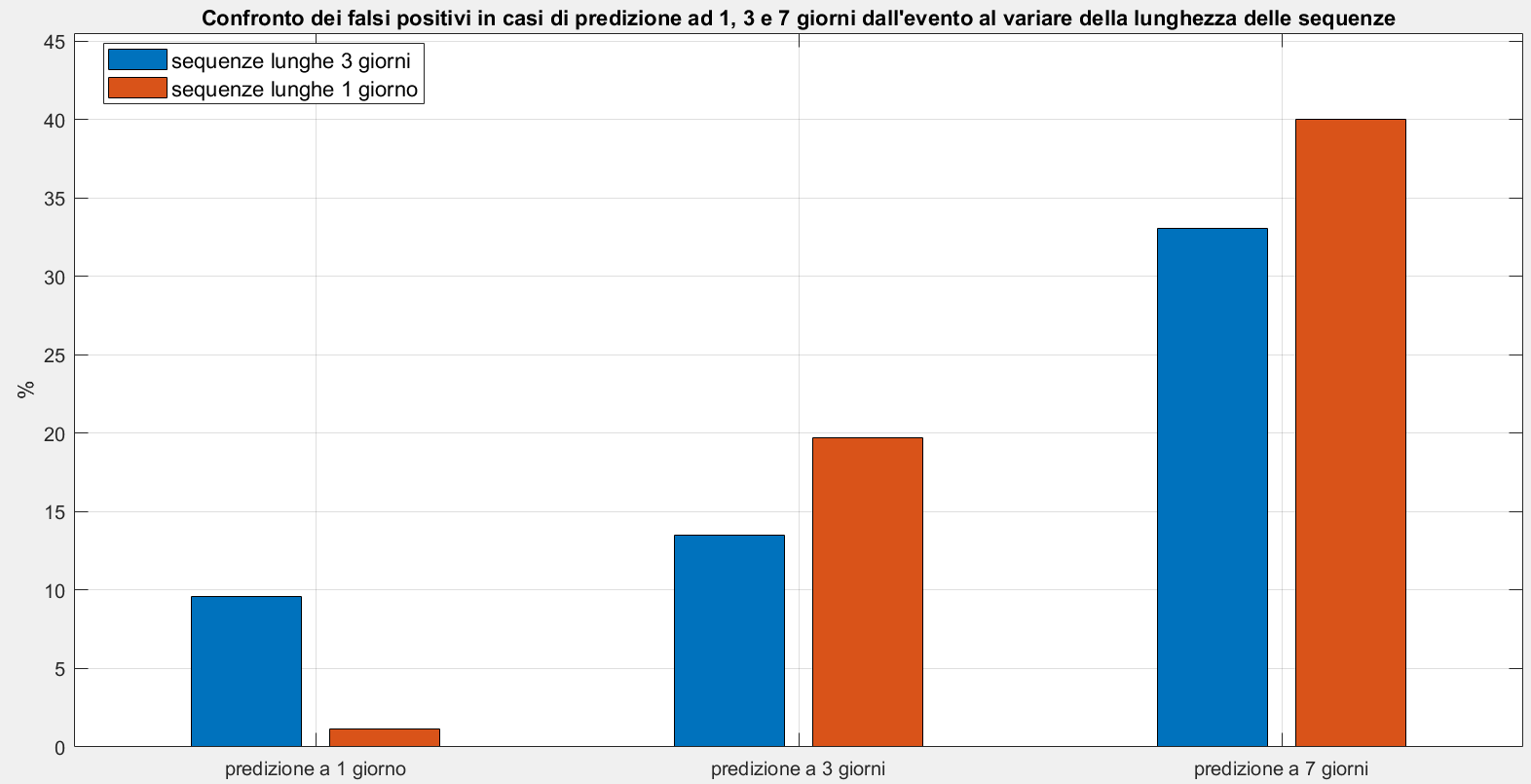


Figura VII‑2: Confronto dei falsi positivi in casi di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze

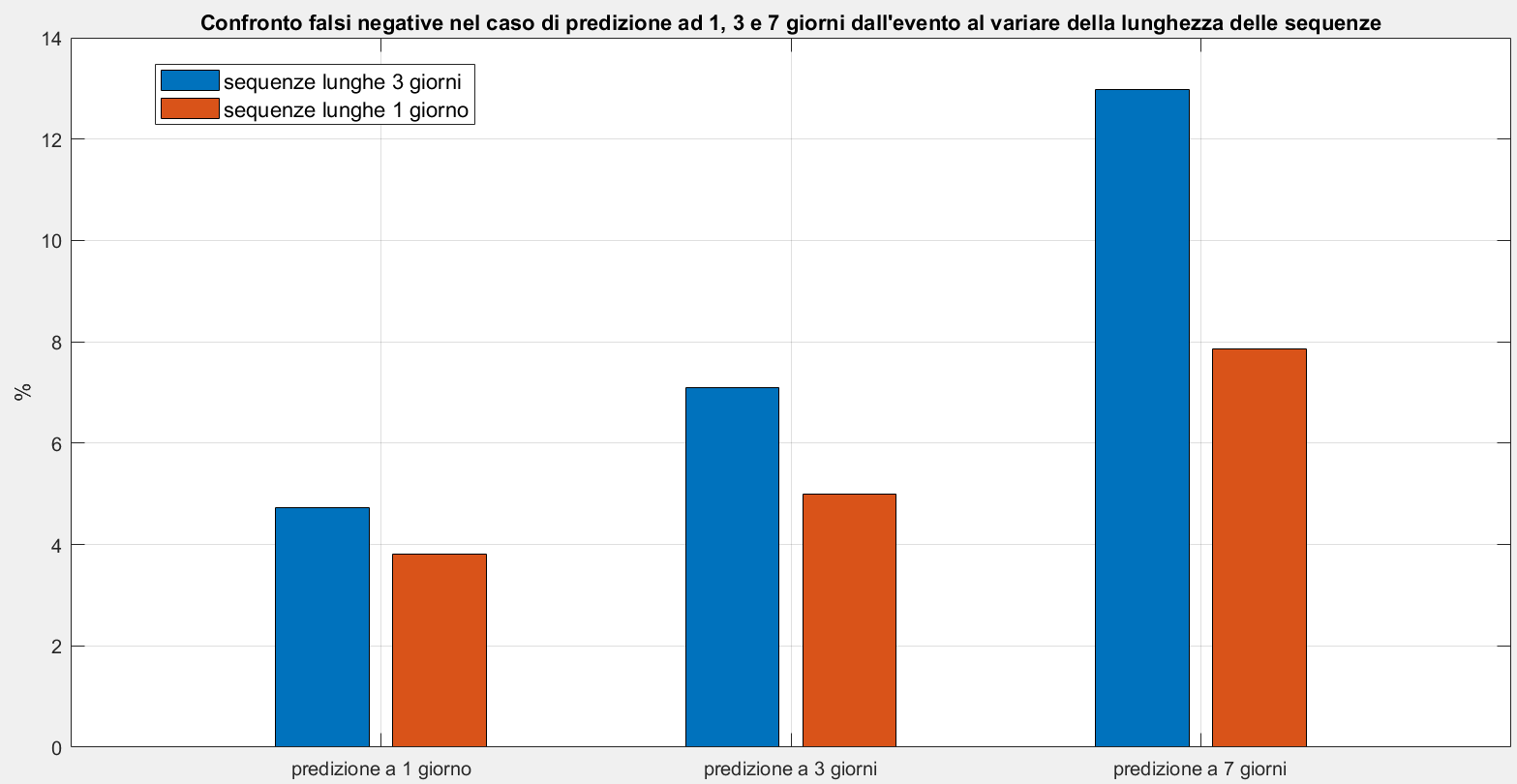


Figura VII‑3: Confronto dei falsi negativi nel caso di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze

Osservando entrambi i grafici vediamo che sia i falsi positivi che i falsi negativi crescono all’aumentare dell’intervallo di predizione. In Figura VII‑2 i falsi positivi per sequenze lunghe 1 giorno con intervallo di predizione di 1 giorno sono all’incirca il 2%, per predizioni a 3 giorni sono all’incirca il 20% e per predizioni a 7 giorni sono all’incirca il 40%. Per le sequenze lunghe 3 giorni che passano all’incirca dal 10%, al 14% al 33%. Constatiamo un maggiore aumento dei falsi positivi per le sequenze lunghe 3 giorno rispetto alle sequenze lunghe 3 giorni.

In Figura VII‑3 i falsi negativi delle sequenze lunghe 3 giorni passano all’incirca dal 5%, al 7%, al 13% rispetto alle sequenze lunghe 1 giorno che vanno dal 4%, al 5% fino al 8%. Constatiamo un maggiore aumento dei falsi negativi per le sequenze lunghe 3 giorni rispetto alle sequenze lunghe 1 giorno.

Osservando sia Figura VII‑2 che Figura VII‑3 vediamo che nel caso di predizione ad 1 giorno dall’evento con sequenze lunghe 1 giorno sia i falsi positivi che negativi sono inferiori in percentuale rispetto al caso di sequenze lunghe 3 giorni. Ciò non si verifica con gli altri intervalli di predizione.

## Tensione della cella minima, potenza del pannello, SOC e irradiazione

Usiamo la combinazione di tensione della cella minima, potenza del pannello, lo stato di carica della batteria (SOC) e l’irradiazione del pannello.

### Predizione ad 1 giorno con sequenze lunghe 3 giorni

Poniamoci nel caso in cui le sequenze durino 3 giorni, siano sfalsate di 1 giorno, e ci sia una proporzione di sequenze tale che per una sequenza patologica ce ne siano 3 sane.

La dimensione dei mini-batch è di 12 elementi con dei fold ognuno da 47 elementi, per un totale di 188 sequenze nel dataset.

L’accuratezza che otteniamo è la seguente:

Otteniamo un’accuratezza approssimativa del 95% con 1,12% di deviazione standard. L’errore di classificazione è approssimativamente del 5%.

Per quanto riguarda la confusion matrix, le righe rappresentano le classi e sulle colonne troviamo le classi assegnate alle sequenze dal modello predittivo. La prima riga rappresenta la classe patologica e la seconda la classe sana:

Tabella 9: Confusion matrix per predizioni con anticipo di 1 giorno e sequenze lunghe 3 giorni

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 42.2 ± 2.2 | 4.8 ± 2.2 |
| 1 | 5.2 ± 0.9 | 135.8 ± 0.9 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 9 evidenzia che sulla totalità del dataset mediamente 5,2 elementi vengono classificati erroneamente come patologici e 4,8 elementi vengono classificati erroneamente come sani. Inoltre, la deviazione standard è maggiore per la classificazione degli elementi patologici.

### Predizione ad 1 giorno con sequenze lunghe 1 giorno

Poniamoci nelle stesse condizioni del caso precedente ma con sequenze di durata 1 giorno.

La dimensione dei mini-batch è di 17 elementi con dei fold ognuno da 34 elementi, per un totale di 136 sequenze nel dataset.

L’accuratezza che otteniamo è la seguente:

Otteniamo un’accuratezza del 97% con 1,59% di deviazione standard. L’errore di classificazione è approssimativamente del 3%.

La media e la deviazione standard delle confusion matrix sono le seguenti:

Tabella 10: Confusion matrix per predizioni con anticpo di 1 giorno e sequenze lunghe 1 giorno

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 33.1 ± 1.0 | 0.9 ± 1.0 |
| 1 | 3.1 ± 1.5 | 98.9 ± 1.5 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 10 evidenzia che sulla totalità del dataset mediamente 3.1 elementi vengono classificati erroneamente come patologici e 0.9 elementi vengono classificati erroneamente come sani. Inoltre, la deviazione standard è maggiore per la classificazione degli elementi sani.

Osserviamo che in questo caso le sequenze lunghe 1 giorno hanno un’accuratezza migliore del 2 % rispetto alle sequenze di 3 giorni.

### Predizione a 3 giorni con sequenze lunghe 3 giorni

In questo caso manteniamo gli stessi parametri ma impostiamo sequenze lunghe 3 giorni e intervallo di predizione uguale a 3 giorni.

La dimensione dei mini-batch è di 15 elementi con dei fold ognuno da 45 elementi, per un totale di 180 sequenze nel dataset.

L’accuratezza che otteniamo è la seguente:

Otteniamo un’accuratezza approssimativa del 91% con 1,49% di deviazione standard. L’errore di classificazione è approssimativamente del 9%.

La media e la deviazione standard delle confusion matrix sono le seguenti:

Tabella 11: Confusion matrix per predizioni con anticipo di 3 giorni e sequenze lunghe 3 giorni

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 37.5 ± 1.9 | 7.5 ± 1.9 |
| 1 | 8.1 ± 1.4 | 126.9 ± 1.4 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 11 evidenzia che sulla totalità del dataset mediamente 8,1 elementi vengono classificati erroneamente come patologici e 7,5 elementi vengono classificati erroneamente come sani. Inoltre, la deviazione standard è maggiore per la classificazione degli elementi patologici.

### Predizione a 3 giorni con sequenze lunghe 1 giorno

In questo caso manteniamo gli stessi parametri ma impostiamo sequenze lunghe 1 giorno.

La dimensione dei mini-batch è di 10 elementi con dei fold ognuno da 31 elementi, per un totale di 124 sequenze nel dataset. Da notare che il numero di sequenze è minore rispetto al caso precedente.

L’accuratezza che otteniamo è la seguente:

Otteniamo un’accuratezza approssimativa del 91% con 1,32% di deviazione standard. L’errore di classificazione è approssimativamente del 9%.

La media e la deviazione standard delle confusion matrix sono le seguenti:

Tabella 12: Confusion matrix per predizioni con anticipo di 3 giorni e sequenze lunghe 1 giorno

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 26.9 ± 1.2 | 4.1 ± 1.2 |
| 1 | 7.6 ± 0.8 | 85.4 ± 0.8 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 12 evidenzia che sulla totalità del dataset mediamente 7,6 elementi vengono classificati erroneamente come patologici e 4,1 elementi vengono classificati erroneamente come sani. Inoltre, la deviazione standard è maggiore per la classificazione degli elementi patologici.

### Predizione a 7 giorni con sequenze lunghe 3 giorni

In questo caso manteniamo gli stessi parametri e impostiamo l’intervallo di predizione uguale a 7 giorni,

La dimensione dei mini-batch è di 15 elementi con dei fold ognuno da 46 elementi, per un totale di 184 sequenze nel dataset.

L’accuratezza che otteniamo è la seguente:

Otteniamo un’accuratezza approssimativa del 82% con 0,96% di deviazione standard. L’errore di classificazione è approssimativamente del 18%.

La media e la deviazione standard delle confusion matrix sono le seguenti:

Tabella 13: Confusion matrix per predizioni con anticipo di 7 giorni e sequenze lunghe 3 giorni

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 31.2 ± 1.6 | 13.8 ± 1.6 |
| 1 | 18.3 ± 1.5 | 116.7 ± 1.5 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 13 evidenzia che sulla totalità del dataset mediamente 18,3 elementi vengono classificati erroneamente come patologici e 13,8 elementi vengono classificati erroneamente come sani. Inoltre, la deviazione standard è maggiore per la classificazione degli elementi sani.

### Predizione a 7 giorni con sequenze lunghe 1 giorno

In questo caso manteniamo gli stessi parametri del paragrafo precedente impostando però la lunghezza delle sequenze di 1 giorno.

La dimensione dei mini-batch è di 10 elementi con dei fold ognuno da 30 elementi, per un totale di 120 sequenze nel dataset.

L’accuratezza che otteniamo è la seguente:

Otteniamo un’accuratezza approssimativa del 82% con 1,51% di deviazione standard. L’errore di classificazione è approssimativamente del 18%.

La media e la deviazione standard delle confusion matrix sono le seguenti:

Tabella 14: Confusion matrix per predizioni con anticipo di 7 giorni e sequenze lunghe 1 giorno

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 18.5 ± 1.3 | 11.5 ± 1.3 |
| 1 | 10.3 ± 1.5 | 79.7 ± 1.5 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 14 evidenzia che sulla totalità del dataset mediamente 10,3 elementi vengono classificati erroneamente come patologici e 11,5 elementi vengono classificati erroneamente come sani. Inoltre, la deviazione standard è maggiore per la classificazione degli elementi sani.

### Considerazioni

A seguire le accuratezze ottenute con la combinazione di tensione della cella minima e potenza del pannello:

Tabella 15: Tabella delle accuratezze ottenute con la tensione della cella minima, la potenza del pannello, il SOC e l’irradiazione

|  |  |  |
| --- | --- | --- |
|  | Sequenze da 3 giorni | Sequenze da 1 giorno |
| Previsione ad 1 giorno | 94.68% ± 1.12% | 97.06% ± 1.59% |
| Previsione a 3 giorni | 91.33% ± 1.49% | 90.56% ± 1.32% |
| Previsione a 7 giorni | 82.17% ± 0.96% | 81.83% ± 1.51% |

A seguire un diagramma a barre che rappresenta le accuratezze in Tabella 15:

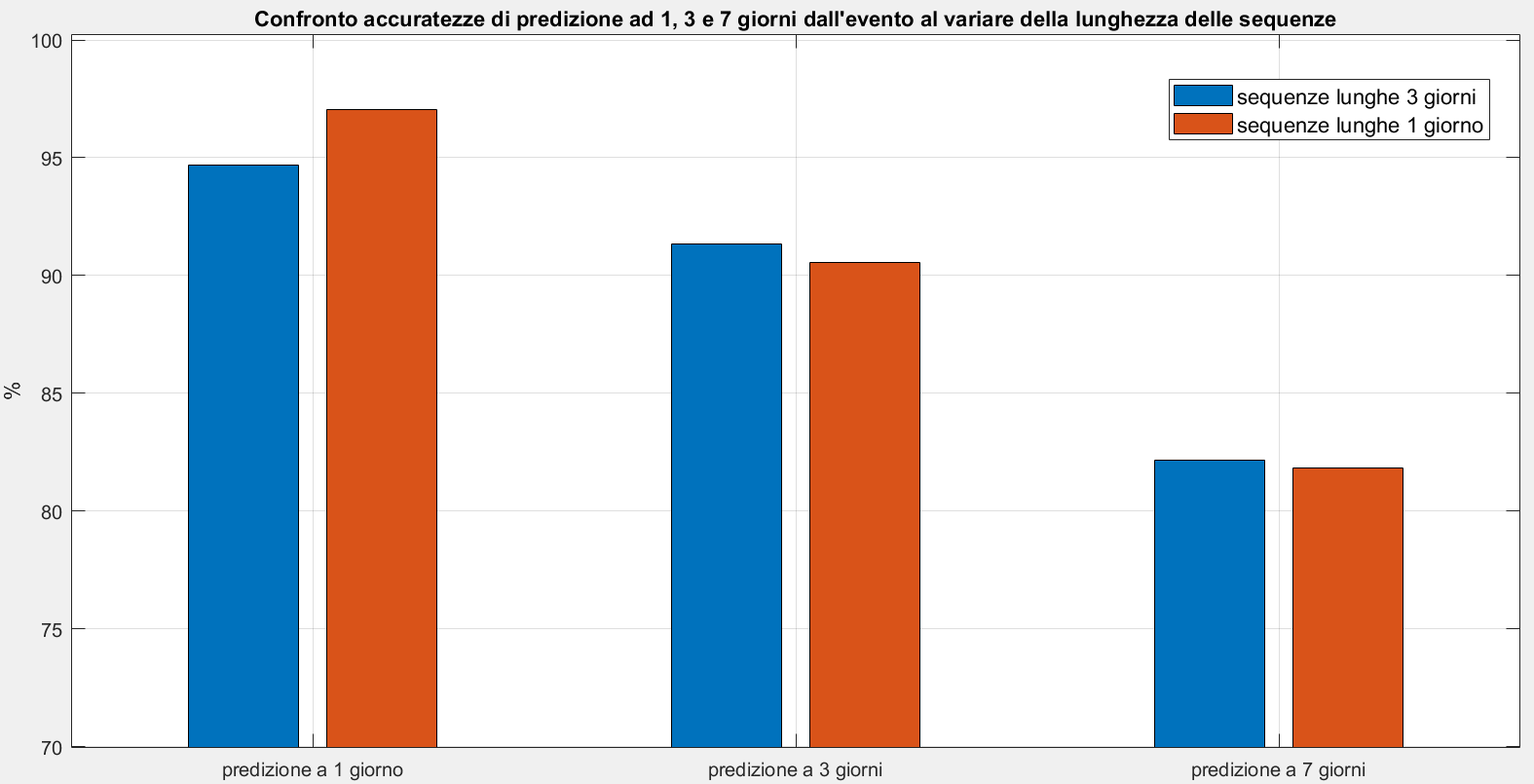


Figura VII‑4: Confronto delle accuratezza di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze

Osserviamo come l’accuratezza decresca a mano a mano che il numero di giorni dall’evento aumenta. L’accuratezza migliore in assoluto è uguale al 97,06% ± 1,59% e viene ottenuta nel caso di predizione ad 1 giorno con sequenze lunghe 1 giorno. Il caso peggiore invece corrisponde a sequenze lunghe 1 giorno a distanza di 7 giorni dall’evento ed è 81,83% ± 1,51%. Inoltre, notiamo che a 3 giorni e a 7 giorni dall’evento le accuratezze per il caso di sequenze lunghe 3 giorni ed 1 giorno sono molto simili (differenza inferiore al 1%), mentre nel caso di sequenze lunghe 1 giorno la differenza di accuratezza è maggiore (2,38 %).

Andiamo ora a raffigurare l’andamento dei falsi positivi (Figura VII‑5) e dei falsi negativi (Figura VII‑6):

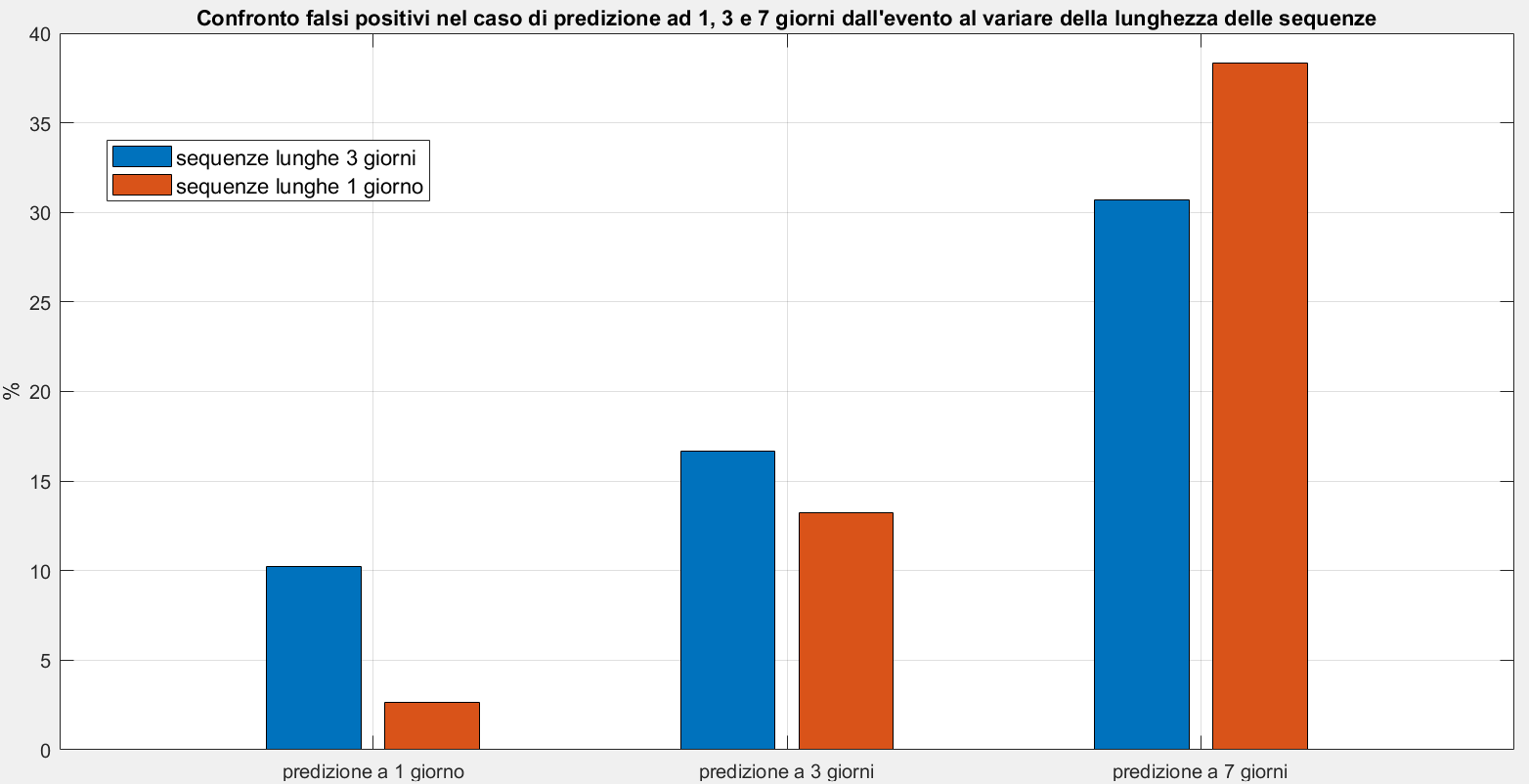


Figura VII‑5:: Confronto dei falsi positivi in casi di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze

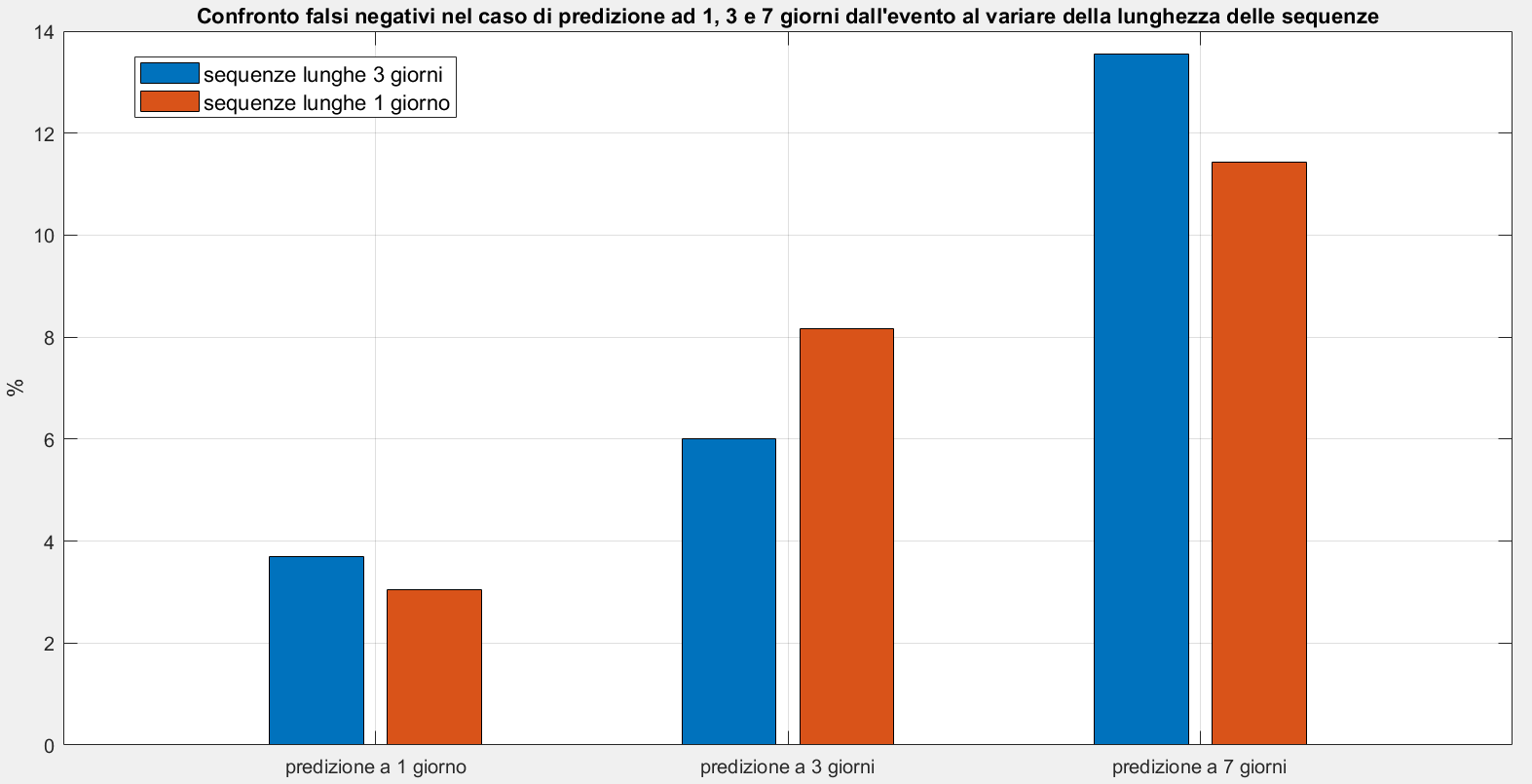


Figura VII‑6: : Confronto dei falsi negativi in casi di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze

In Figura VII‑5 i falsi positivi per sequenze lunghe 1 giorno con intervallo di predizione di 1 giorno sono all’incirca il 3%, per predizioni a 3 giorni sono all’incirca il 13% e per predizioni a 7 giorni sono all’incirca il 38%. Per le sequenze lunghe 3 giorni passiamo invece dal 10%, al 17% al 30%. Constatiamo un maggiore aumento dei falsi positivi per le sequenze lunghe 1 giorno rispetto alle sequenze lunghe 3 giorni.

In Figura VII‑6 i falsi negativi delle sequenze lunghe 3 giorni passano all’incirca dal 4%, al 6%, al 14% rispetto alle sequenze lunghe 1 giorno che vanno dal 3%, al 8% fino al 11%. Constatiamo una crescita simile sia per i falsi negativi di sequenze lunghe 1 giorno, sia di quelli di sequenze lunghe 3 giorni.

Osservando sia Figura VII‑5 che Figura VII‑6 vediamo che nel caso di predizione ad 1 giorno dall’evento con sequenze lunghe 1 giorno sia i falsi positivi che negativi sono inferiori in percentuale rispetto al caso di sequenze lunghe 3 giorni. Ciò non si verifica con gli altri intervalli di predizione.

## Tensione della cella minima, potenza del pannello, SOC, irradiazione e corrente della batteria

Usiamo la combinazione di tensione della cella minima, potenza del pannello, lo stato di carica della batteria (SOC), l’irradiazione del pannello e la corrente totale della batteria (in entrata e in uscita).

### Predizione ad 1 giorno con sequenze lunghe 3 giorni

Poniamoci nel caso in cui le sequenze durino 3 giorni, siano sfalsate di 1 giorno, e ci sia una proporzione di sequenze tale che per una sequenza patologica ce ne siano 3 sane.

La dimensione dei mini-batch è di 12 elementi con dei fold ognuno da 47 elementi, per un totale di 188 sequenze nel dataset.

L’accuratezza che otteniamo è la seguente:

Otteniamo un’accuratezza approssimativa del 93% con 0,76% di deviazione standard. L’errore di classificazione è approssimativamente del 7%.

Per quanto riguarda la matrice di confusione:

Tabella 16: Confusion matrix per predizioni con anticipo di 1 giorno e sequenze lunghe 3 giorni

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 40.2 ± 1.6 | 6.8 ± 1.6 |
| 1 | 5.6 ± 1.5 | 135.4 ± 1.5 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 16 evidenzia che sulla totalità del dataset mediamente 5,6 elementi vengono classificati erroneamente come patologici e 6,8 elementi vengono classificati erroneamente come sani. Inoltre, la deviazione standard per la classificazione degli elementi patologici e sani è approssimativamente uguale.

### Predizione ad 1 giorno con sequenze lunghe 1 giorno

Poniamoci nelle stesse condizioni del caso precedente ma con sequenze di durata 1 giorno.

La dimensione dei mini-batch è di 17 elementi con dei fold ognuno da 34 elementi, per un totale di 136 sequenze nel dataset.

L’accuratezza che otteniamo è la seguente:

Otteniamo un’accuratezza del 98% con 0,83% di deviazione standard. L’errore di classificazione è approssimativamente del 2%.

La media e la deviazione standard delle confusion matrix sono le seguenti:

Tabella 17: Confusion matrix per predizioni con anticipo di 1 giorno e sequenze lunghe 1 giorno

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 32.6 ± 1.4 | 1.4 ± 1.4 |
| 1 | 1.8 ± 1.2 | 100.2 ± 1.2 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 17 evidenzia che sulla totalità del dataset mediamente 1.8 elementi vengono classificati erroneamente come patologici e 1.4 elementi vengono classificati erroneamente come sani. Inoltre, la deviazione standard è maggiore per la classificazione degli elementi patologici.

Osserviamo che in questo caso le sequenze lunghe 1 giorno hanno un’accuratezza migliore del 2 % rispetto alle sequenze di 3 giorni.

### Predizione a 3 giorni con sequenze lunghe 3 giorni

In questo caso manteniamo gli stessi parametri ma impostiamo sequenze lunghe 3 giorni e intervallo di predizione uguale a 3 giorni.

La dimensione dei mini-batch è di 15 elementi con dei fold ognuno da 45 elementi, per un totale di 180 sequenze nel dataset.

L’accuratezza che otteniamo è la seguente:

Otteniamo un’accuratezza approssimativa del 90% con 1,29% di deviazione standard. L’errore di classificazione è approssimativamente del 10%.

La media e la deviazione standard delle confusion matrix sono le seguenti:

Tabella 18: Confusion matrix per predizioni con anticipo di 3 giorni e sequenze lunghe 3 giorni

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 37.9 ± 1.4 | 7.1 ± 1.4 |
| 1 | 10.4 ± 2.0 | 124.6 ± 2.0 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 18 evidenzia che sulla totalità del dataset mediamente 10,4 elementi vengono classificati erroneamente come patologici e 7,1 elementi vengono classificati erroneamente come sani. Inoltre, la deviazione standard è maggiore per la classificazione degli elementi sani.

### Predizione a 3 giorni con sequenze lunghe 1 giorno

In questo caso manteniamo gli stessi parametri ma impostiamo sequenze lunghe 1 giorno.

La dimensione dei mini-batch è di 10 elementi con dei fold ognuno da 31 elementi, per un totale di 124 sequenze nel dataset. Da notare che il numero di sequenze è minore rispetto al caso precedente.

L’accuratezza che otteniamo è la seguente:

Otteniamo un’accuratezza approssimativa del 89% con 1,14% di deviazione standard. L’errore di classificazione è approssimativamente del 11%.

La media e la deviazione standard delle confusion matrix sono le seguenti:

Tabella 19: Confusion matrix per predizioni con anticpo di 3 giorni e sequenze lunghe 1 giorno

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 27.2 ± 1.5 | 3.8 ± 1.5 |
| 1 | 9.9 ± 1.5 | 83.1 ± 1.5 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 19 evidenzia che sulla totalità del dataset mediamente 9,9 elementi vengono classificati erroneamente come patologici e 3,8 elementi vengono classificati erroneamente come sani. Inoltre, le deviazioni standard per la classificazione degli elementi patologici e la classificazione degli elementi sani sono approssimativamente uguali.

### Predizione a 7 giorni con sequenze lunghe 3 giorni

In questo caso manteniamo gli stessi parametri e impostiamo l’intervallo di predizione uguale a 7 giorni.

La dimensione dei mini-batch è di 15 elementi con dei fold ognuno da 45 elementi, per un totale di 180 sequenze nel dataset.

L’accuratezza che otteniamo è la seguente:

Otteniamo un’accuratezza approssimativa del 85% con 1,02% di deviazione standard. L’errore di classificazione è approssimativamente del 15%.

La media e la deviazione standard delle confusion matrix sono le seguenti:

Tabella 20: Confusion matrix per predizioni con anticipo di 7 giorni e sequenze lunghe 3 giorni

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 32.4 ± 1.4 | 12.6 ± 1.4 |
| 1 | 13.8 ± 1.3 | 121.2 ± 1.3 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 20 evidenzia che sulla totalità del dataset mediamente 13,8 elementi vengono classificati erroneamente come patologici e 12,6 elementi vengono classificati erroneamente come sani. Inoltre, la deviazione standard è maggiore per la classificazione degli elementi patologici.

### Predizione a 7 giorni con sequenze lunghe 1 giorno

In questo caso manteniamo gli stessi parametri del paragrafo precedente impostando però la lunghezza delle sequenze di 1 giorno.

La dimensione dei mini-batch è di 10 elementi con dei fold ognuno da 30 elementi, per un totale di 120 sequenze nel dataset.

I risultati che otteniamo sono i seguenti:

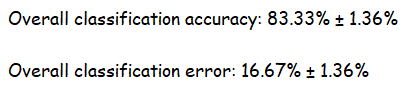


Figura VII‑7: Accuratezza ed errore medi

Otteniamo un’accuratezza approssimativa del 83% con 1,36% di deviazione standard. L’errore di classificazione è approssimativamente del 17%.

La media e la deviazione standard delle confusion matrix sono le seguenti:

Tabella 21: Confusion matrix per predizioni con anticipo di 7 giorni e sequenze lunghe 1 giorno

|  |  |  |  |
| --- | --- | --- | --- |
| Etichette di riferimento | 0 | 18.1 ± 1.1 | 11.9 ± 1.1 |
| 1 | 8.1 ± 1.3 | 81.9 ± 1.3 |
|  | | 0 | 1 |
| Etichette predette | |

La confusion matrix media in Tabella 21 evidenzia che sulla totalità del dataset mediamente 8,1 elementi vengono classificati erroneamente come patologici e 11,9 elementi vengono classificati erroneamente come sani. Inoltre, la deviazione standard è maggiore per la classificazione degli elementi sani.

### Considerazioni

A seguire le accuratezze ottenute con la combinazione di tensione della cella minima e potenza del pannello:

Tabella 22: Tabella delle accuratezze ottenute con la tensione della cella minima, la potenza del pannello, il SOC, l’irradiazione e la corrente della batteria

|  |  |  |
| --- | --- | --- |
|  | Sequenze da 3 giorni | Sequenze da 1 giorno |
| Previsione ad 1 giorno | 93.40% ± 0.76% | 97.65% ± 0.83% |
| Previsione a 3 giorni | 90.28% ± 1.29% | 88.95% ± 1.14% |
| Previsione a 7 giorni | 85.33% ± 1.02% | 83.33% ± 1.36% |

A seguire un diagramma a barre che riporta le accuratezze in Tabella 22:Tabella 8: Tabella delle accuratezze ottenute con la tensione della cella minima e la potenza del pannello

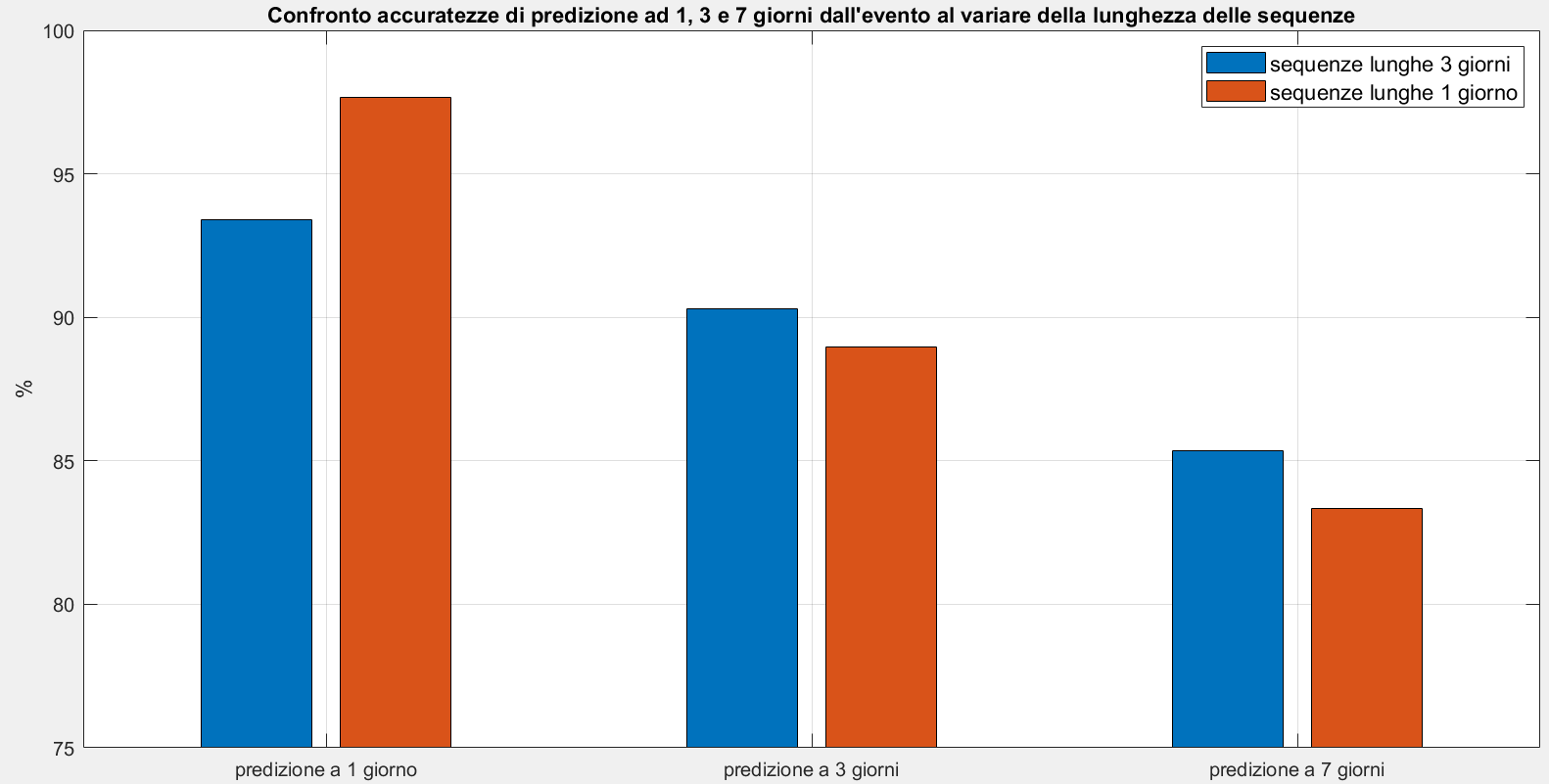


Figura VII‑8: Confronto delle accuratezza di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze

Osserviamo come l’accuratezza decresca a mano a mano che il numero di giorni dall’evento aumenta. L’accuratezza migliore in assoluto è uguale al 97,65% ± 0,83 % e viene ottenuta nel caso di predizione ad 1 giorno con sequenze lunghe 1 giorno. Il caso peggiore invece corrisponde a sequenze lunghe 1 giorno a distanza di 7 giorni dall’evento ed è 83,33% ± 1,36%. Inoltre, notiamo che a 3 giorni e a 7 giorni dall’evento le accuratezze nel caso di sequenze lunghe 3 giorni sono maggiori di quelle ad 1 giorno rispettivamente di 1,3% e 2%. Nel caso di previsione ad 1 giorno dell’evento l’accuratezza per sequenze lunghe 1 giorno è maggiore di quelle lunghe 3 giorni di 4,24%.

Andiamo ora a raffigurare l’andamento dei falsi positivi (Figura VII‑9) e dei falsi negativi (Figura VII‑10):

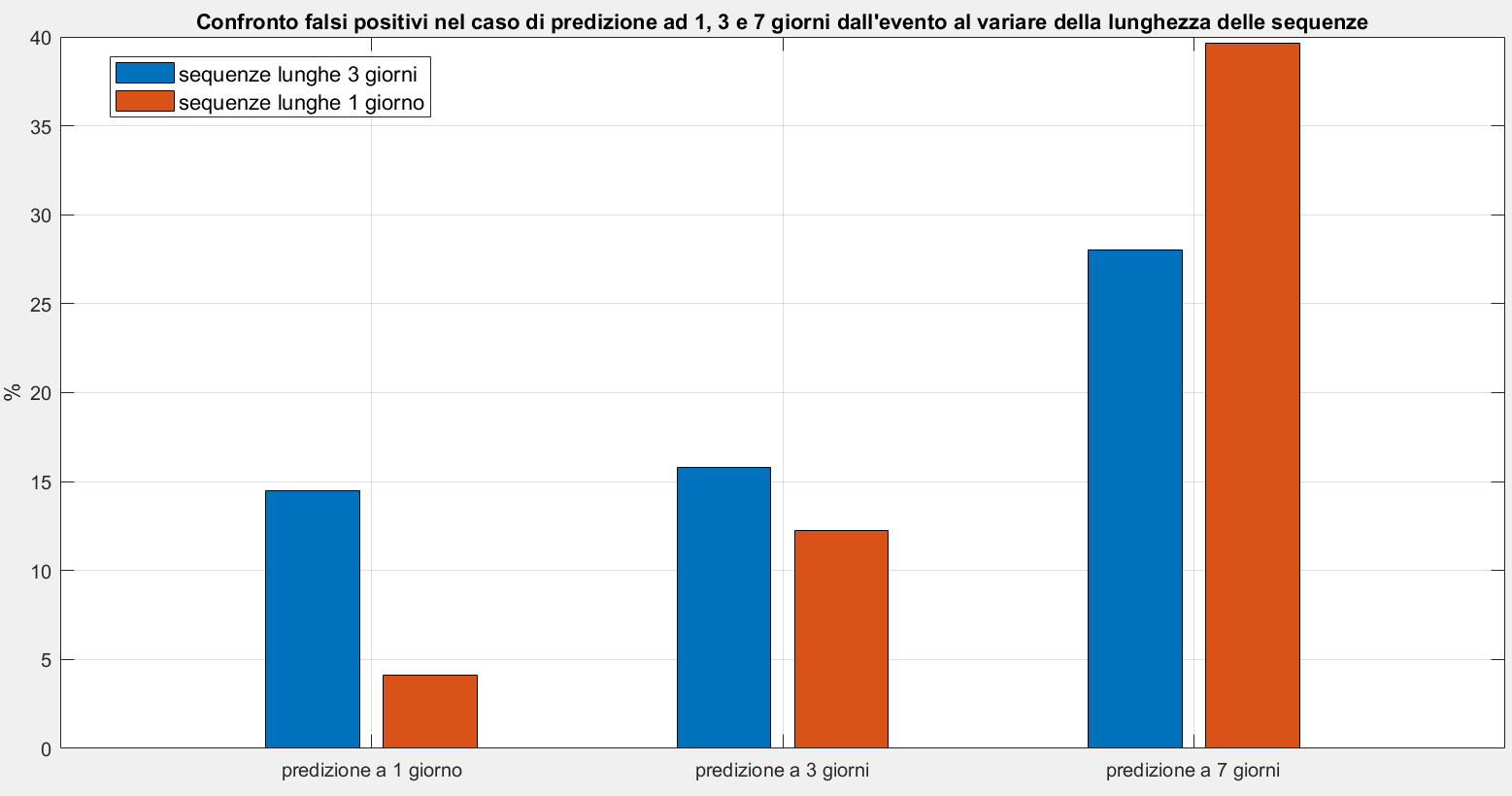


Figura VII‑9: Confronto dei falsi positivi in casi di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze

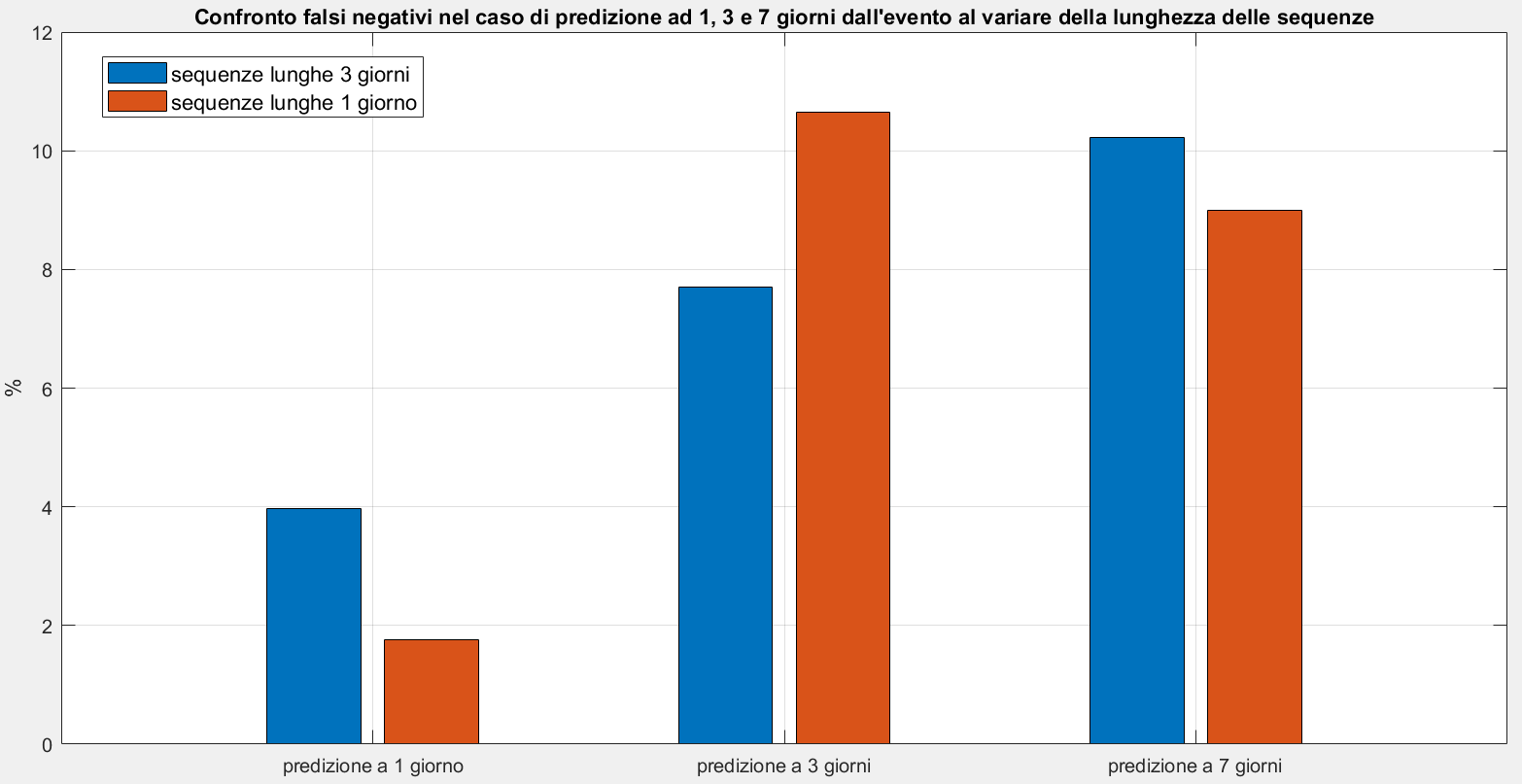


Figura VII‑10: Confronto dei falsi negativi in casi di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze

In Figura VII‑9 i falsi positivi per sequenze lunghe 1 giorno con intervallo di predizione di 1 giorno sono all’incirca il 4%, per predizioni a 3 giorni sono all’incirca il 12% e per predizioni a 7 giorni sono all’incirca il 40%. Per sequenze lunghe 3 giorni passiamo invece all’incirca dal 15%, al 16% al 28. Constatiamo un maggiore aumento dei falsi positivi per le sequenze lunghe 1 giorno rispetto alle sequenze lunghe 3 giorni.

Riguardo i falsi negativi in Figura VII‑10 le sequenze lunghe 3 giorni passano all’incirca dal 4%, al 6%, al 10% rispetto alle sequenze lunghe 1 giorno che vanno dal 2%, al 11% per poi diminuire fino al 9%. Constatiamo che per sequenze lunghe 3 giorni l’aumento è lineare rispetto all’intervallo di predizione, mentre nel caso di sequenze lunghe 1 giorno abbiamo un forte aumento tra 1 e 3 giorni mentre poi si inverte la pendenza diventando un decremento del 2 % a 7 giorni.

Osservando sia Figura VII‑9 che Figura VII‑10 vediamo che per predizione ad 1 giorno dall’evento con sequenze lunghe 1 giorno, sia i falsi positivi che negativi sono inferiori rispetto al caso di sequenze lunghe 3 giorni. Ciò non si verifica con gli altri intervalli di predizione.

# Conclusione

# Appendice

Immagine che contiene testo

Descrizione generata automaticamente Figura IX‑1: Query generata in Matlab

# Indice delle figure

[Figura II‑1: Neurone artificiale 7](#_Toc102647963)

[Figura II‑2: Rete feedforward 8](#_Toc102647964)

[Figura II‑3: Recurrent Neural Network 9](#_Toc102647965)

[Figura II‑4: Schema RNN srotolato 10](#_Toc102647966)

[Figura II‑5: Rete RNN bidirezionale 12](#_Toc102647967)

[Figura II‑6: Predizione delle parole in una frase in base al resto della frase 12](#_Toc102647968)

[Figura II‑7: Rete LSTM 13](#_Toc102647969)

[Figura II‑8: Cella LSTM 13](#_Toc102647970)

[Figura III‑1: Diversi tipi di manutenzione 16](#_Toc102647971)

[Figura III‑2: Schema di manutenzione predittiva 17](#_Toc102647972)

[Figura III‑3: Costi dell'implementazione di un sistema di manutenzione predittiva 18](#_Toc102647973)

[Figura IV‑1: Schema a blocchi di un dispositivo di tipo A 22](#_Toc102647974)

[Figura IV‑2: Implementazione hardware di un dispositivo di tipo B 23](#_Toc102647975)

[Figura IV‑3: Connessione bidirezionale tra i dispositivi in campo e il databse 25](#_Toc102647976)

[Figura V‑1: Struttura del dato ritornato dalla query 28](#_Toc102647977)

[Figura V‑2: Cell array contenente i dati dal database 29](#_Toc102647978)

[Figura V‑3: Tabella di corrispondenza dei codici delle variabili 30](#_Toc102647979)

[Figura V‑4: Cell array contenente le struct 30](#_Toc102647980)

[Figura V‑5: Interpolazione 31](#_Toc102647981)

[Figura V‑6: Interpolazione dei campioni in diagnostica 31](#_Toc102647982)

[Figura V‑7: Traslazione del dato 32](#_Toc102647983)

[Figura V‑8: Sovrascrittura dei valori invariati 33](#_Toc102647984)

[Figura V‑9: Funzione di sovra campionamento 33](#_Toc102647985)

[Figura V‑10: Sovra campionamento al minuto 34](#_Toc102647986)

[Figura V‑11: allineamento delle sequenze temporali 35](#_Toc102647987)

[Figura V‑12: Struct finale 35](#_Toc102647988)

[Figura V‑13: Messa a zero della potenza del pannello di notte 36](#_Toc102647989)

[Figura V‑14: Correzione del valore nullo in diagnostica 37](#_Toc102647990)

[Figura V‑15: Parametri 37](#_Toc102647991)

[Figura V‑16: Main 38](#_Toc102647992)

[Figura V‑17: Estrazione delle sequenze nel main 40](#_Toc102647993)

[Figura V‑18: Differenza di andamenti tra dato sovra campionato e non, là dove si verifica un buco di dati 41](#_Toc102647994)

[Figura V‑19: funzione "mezzanotte" per scandire i giorni 42](#_Toc102647995)

[Figura V‑20: Selezione di sequenze da un giorno 43](#_Toc102647996)

[Figura V‑21: date di fine giornata 43](#_Toc102647997)

[Figura V‑22: Creazione delle sequenze 44](#_Toc102647998)

[Figura V‑23: Paragone tra le sequenze estratte e la totalità del dato sincronizzato 45](#_Toc102647999)

[Figura V‑24: Lasso temporale non valido ai fini del dataset 46](#_Toc102648000)

[Figura V‑25: Esempio di una sequenza 46](#_Toc102648001)

[Figura V‑26: Funzione "sospetti" nel main 47](#_Toc102648002)

[Figura V‑27: Suddivisione delle sequenze tra sane e patologiche 47](#_Toc102648003)

[Figura V‑28: Estrazione delle sequenze predittive 48](#_Toc102648004)

[Figura V‑29: Ripetto delle proporzioni tra sequenze sane e patologiche 48](#_Toc102648005)

[Figura V‑30: sequenza patologica 49](#_Toc102648006)

[Figura V‑31: Sequenza predittiva 49](#_Toc102648007)

[Figura V‑32: Selezione della sequenza predittiva con predzione a 7giorni e sequenza di riferimento sana 50](#_Toc102648008)

[Figura V‑33:Selezione della sequenza predittiva con predzione a 7giorni e sequenza di riferimento patologica 50](#_Toc102648009)

[Figura V‑34: Disposizione tra sequenze di riferimento e predittive a 7 giorni per ogni torre 51](#_Toc102648010)

[Figura V‑35: Traliccio 13008 52](#_Toc102648011)

[Figura V‑36: Traliccio 1021 53](#_Toc102648012)

[Figura V‑37: Traliccio 1025 53](#_Toc102648013)

[Figura V‑38: Traliccio 1059 54](#_Toc102648014)

[Figura V‑39: Traliccio 16399 54](#_Toc102648015)

[Figura V‑40: Periodo in cui l’andamento della tensione della cella minima diventa patologico 55](#_Toc102648016)

[Figura V‑41: Prima sequenza 56](#_Toc102648017)

[Figura V‑42: Seconda sequenza 56](#_Toc102648018)

[Figura V‑43: Terza sequenza 57](#_Toc102648019)

[Figura V‑44: Sfalsamento delle sequenze 58](#_Toc102648020)

[Figura V‑45: Normalizzazione 59](#_Toc102648021)

[Figura V‑46: Funzione di normalizzazione 59](#_Toc102648022)

[Figura V‑47: Andamento della tensione della cella minima normalizzata 60](#_Toc102648023)

[Figura V‑48: Funzione "etichette" nel main 61](#_Toc102648024)

[Figura V‑49: Etichettatura del dataset 61](#_Toc102648025)

[Figura V‑50: Funzione per la creazione del dataset nel main 62](#_Toc102648026)

[Figura V‑51: Creazione del dataset 63](#_Toc102648027)

[Figura V‑52: Partizione statica 64](#_Toc102648028)

[Figura V‑53: Sequenze patologiche della tensione della cella minima 65](#_Toc102648029)

[Figura V‑54: Sequenze sane della tensione della cella minima 65](#_Toc102648030)

[Figura V‑55: Sequenze patologiche della potenza del pannello 67](#_Toc102648031)

[Figura V‑56: Sequenze sane della potenza del pannello 68](#_Toc102648032)

[Figura V‑57: Sequenze patologiche del SOC 69](#_Toc102648033)

[Figura V‑58: Sequenze sane del SOC 69](#_Toc102648034)

[Figura V‑59: Partizione dinamica 70](#_Toc102648035)

[Figura V‑60: Pocedura di selezione del dataset 70](#_Toc102648036)

[Figura V‑61: Caso di sequenze di durata 1 giorno 71](#_Toc102648037)

[Figura V‑62: Rappresentazione delle sequenze senza sovrapposizione 72](#_Toc102648038)

[Figura VI‑1: Layers della rete neurale 72](#_Toc102648039)

[Figura VI‑2: Parametri della rete 73](#_Toc102648040)

[Figura VI‑3: Opzioni della rete 74](#_Toc102648041)

[Figura VI‑4: Lunghezza delle sequenze uniforme 75](#_Toc102648042)

[Figura VI‑5: Ttraining e classificazione 75](#_Toc102648043)

[Figura VI‑6: Andamento tipico dell'accuratezza e della funzione di perdita con le suddette opzioni della rete 76](#_Toc102648044)

[Figura VI‑7: Calcola dell'accuratezza e confusion chart 76](#_Toc102648045)

[Figura VI‑8: Confusion chart 77](#_Toc102648046)

[Figura VI‑9: Algoritmo d K-Folding 78](#_Toc102648047)

[Figura VII‑1: Confronto delle accuratezza di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze 84](#_Toc102648048)

[Figura VII‑2: Confronto dei falsi positivi in casi di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze 86](#_Toc102648049)

[Figura VII‑3: Confronto dei falsi negativi nel caso di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze 86](#_Toc102648050)

[Figura VII‑4: Confronto delle accuratezza di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze 92](#_Toc102648051)

[Figura VII‑5:: Confronto dei falsi positivi in casi di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze 93](#_Toc102648052)

[Figura VII‑6: : Confronto dei falsi negativi in casi di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze 93](#_Toc102648053)

[Figura VII‑7: Accuratezza ed errore medi 98](#_Toc102648054)

[Figura VII‑8: Confronto delle accuratezza di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze 99](#_Toc102648055)

[Figura VII‑9: Confronto dei falsi positivi in casi di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze 101](#_Toc102648056)

[Figura VII‑10: Confronto dei falsi negativi in casi di predizione ad 1, 3 e 7 giorni dall'evento al variare della lunghezza delle sequenze 101](#_Toc102648057)

[Immagine che contiene testo

Descrizione generata automaticamente Figura IX‑1: Query generata in Matlab 103](#_Toc102648058)

# Bibliografia

1. Using Deep Learning for Predictive Maintenance, Texas Instruments <https://training.ti.com/sites/default/files/docs/using-deep-learning-for-predictive-maintenance-slides.pdf>
2. Data-Driven Methods for Predictive Maintenance of Industrial Equipment: A Survey. Weiting Zhang , Dong Yang , Member, IEEE, and Hongchao Wang , Member, IEEE
3. Machine Learning for Predictive Maintenance: A Multiple Classifier Approach”. Gian Antonio Susto, Andrea Schirru, Simone Pampuri, Seán McLoone, Senior Member, IEEE, and Alessandro Beghi, Member, IEEE
4. “Prognostics and Health Management: A Review on Data Driven Approaches”. Kwok L. Tsui, Nan Chen, Qiang Zhou, Yizhen Hai, andWenbinWang
5. L. Layton, M. Glod, and L. H. Sun, Probe Finds Metro Control. “Anomalies”, The Washingthon Post, Washington, DC, USA, 2009.
6. J. Lyons, “Brazil blackout sparks infrastructure concerns,” The Washington Post, 2009.
7. J. Finch, “Toyota sudden acceleration: a case study of the national highway traffic safety administration-recalls forchange,” Loyola Consumer Law Review, vol. 22, p. 472, 2009.
8. <https://smartme.io/blog/conosciamo-il-protocollo-di-comunicazione-lora/>
9. https://www.vodafone.it/portal/Privati/Supporto/Glossario/lte
10. “Battery Management System (BMS) for Lithium-Ion Batteries”, José Miguel Branco Marques, September 2014
11. “Reti neurali e vita artificiale“, Domenico Parisi - Enciclopedia della Scienza e della Tecnica (2007). <https://www.treccani.it/enciclopedia/reti-neurali-e-vita-artificiale_%28Enciclopedia-della-Scienza-e-della-Tecnica%29/>
12. “Soft Computing & Computational Intelligence”, Circuiti e Algoritmi per il Calcolo Distribuito – Prof. Massimo
13. “Deep Learning and Deep Neural Networks”, Circuiti e Algoritmi per il Calcolo Distribuito – Prof. Massimo
14. “On the difficulty of training recurrent neural networks”, Razvan Pascanu, Tomas Mikolov, Yoshua Bengio
15. “[Bidirectional recurrent neural networks](https://www.researchgate.net/profile/Mike_Schuster/publication/3316656_Bidirectional_recurrent_neural_networks/links/56861d4008ae19758395f85c.pdf)”, M. Schuster, K.K. Paliwal
16. “Long short-term memory”, S Hochreiter, J Schmidhuber – 1997
17. “Illustrated Guide to LSTM’s and GRU’s: A step by step explanation”, Michael Phi - 2018. <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>”
18. <https://d2l.ai/chapter_recurrent-modern/bi-rnn.html>
19. UNE-EN 13306. 2018. Maintenance. Maintenance terminology. Standard. Asociación Española de Normalización, Génova, Madrid.
20. “Smart IoT Monitoring System for Agriculture with Predictive Analysis”, Alaa Adel Araby, Mai Mohamed Abd Elhameed, Nada Mohamed Magdy, Loa’a Ahmed Said, Nada Abdelaal, Yomna Tarek Abd Allah, M. Saeed Darweesh, Mohamed Ali Fahim, Hassan Mostafa - 2019
21. “Data-driven predictive maintenance scheduling policies for railways”, Pedro Cesar Lopes Gerum, Ayca Altay, Melike Baykal-Gürsoy
22. Srikanth Namuduri et al 2020 J. Electrochem. Soc. 167 037552
23. “Fault diagnosis and remaining useful life estimation of aero engine using LSTM network”, Mei Yuan, Wu and Li Lin
24. “Early Fault Detection of Machine Tools Based on Deep Learning and Dynamic Identification”, Bo Luo , Haoting Wang , Hongqi Liu, Bin Li, and Fangyu Peng

Immagine che contiene testo

Descrizione generata automaticamente