



UNIVERSITÀ POLITECNICA DELLE MARCHE  
FACOLTÀ DI INGEGNERIA  
Corso di Laurea Triennale in Ingegneria Meccanica

**Studio e sviluppo di tecniche per la disaggregazione  
dell'energia prodotta da pannelli fotovoltaici a partire  
da dati aggregati**

**Study and development of techniques for  
disaggregating the energy produced by photovoltaic  
panels starting from aggregated data**

*Relatore:*

Prof. Emanuele Principi

*Tesi di laurea di:*

Graziosi Noemi

---

Anno accademico 2021/2022

## ***Abstract***

*Il tema della disaggregazione dell'energia solare ha preso piede negli ultimi anni poiché le società di servizi pubblici stanno cercando dei metodi per identificare la generazione di energia elettrica fotovoltaica del consumo netto di un'abitazione residenziale. L'obiettivo è quello di migliorare le loro pratiche di pianificazione e funzionamento e applicare prezzi variabili alla generazione solare distribuita.*

*In questo progetto di tesi proponiamo un algoritmo di apprendimento automatico supervisionato per disaggregare la generazione di energia elettrica dei sistemi fotovoltaici dall'alimentatore principale di un'abitazione. Per sviluppare l'algoritmo ci siamo serviti di finestre scorrevoli a bassa complessità e tecniche convenzionali di apprendimento automatico, applicate a dati di abitazioni di famiglie reali ottenuti dal set di dati del progetto Smart\*.*

*Il nostro studio presenta un'evoluzione rispetto ai lavori precedenti [10][14]: l'utilizzo del Support Vector Regression come modello di regressione.*

# Indice

<b>Capitolo 1, Introduzione:</b>	.....
<b>1.1 Diffusione dell'energia solare</b>	.....
<b>1.2 Disaggregazione del carico</b>	.....
<b>Capitolo 2, Descrizione dell' algoritmo</b>	.....
<b>2.1 Interpretazione datasets</b>	.....
<b>2.2 Pre-processing</b>	.....
<b>2.3 Set di addestramento dati</b>	.....
<b>2.4 Modelli di regressione</b>	.....
<b>2.5 Valutazione modello</b>	.....
<b>Capitolo 3, Sperimentazione</b>	.....
<b>3.1 Descrizione datasets</b>	.....
<b>3.2 Preparazione datasets</b>	.....
<b>3.3 Selezione modelli di regressione</b>	.....
<b>3.4 Risultati</b>	.....
<b>Capitolo 4, Conclusioni e possibili applicazioni dell' algoritmo</b>	.....
<b>Appendice A</b>	.....
<b>Appendice B</b>	.....
<b>Appendice C</b>	.....
<b>Appendice D</b>	.....
<b>Riferimenti</b>	.....



# CAPITOLO 1

## Introduzione

### 1.1 Diffusione dell'energia solare

Secondo lo studio New Energy Outlook 2018 condotto da Bloomberg, nel 2050 lo scenario della generazione di energia elettrica a livello mondiale sarà dominato dalle fonti rinnovabili che, grazie anche allo sviluppo delle batterie, garantiranno il 50% del fabbisogno mondiale [2].

L'energia del mondo entro il 2050

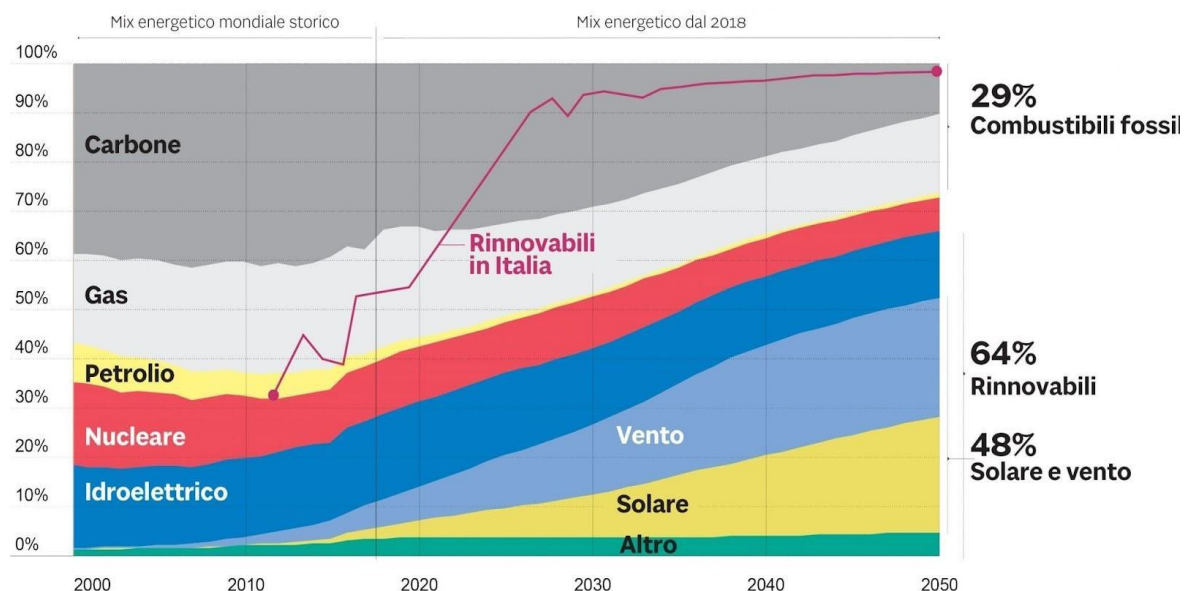


Figura 1.1.2 Scenario generazione elettrica mondiale entro 2050 secondo New Energy Outlook

In Italia questo processo sarà più rapido, infatti si stima che entro il 2030 le fonti di generazione eolica e solare riusciranno a garantire il 90% del fabbisogno, fino ad arrivare al 100% entro il 2050. La sempre maggiore competitività delle fonti rinnovabili secondo questo studio dovrebbe portare all'abbandono del carbone entro il 2035.

Per quanto riguarda gli investimenti, è previsto uno stanziamento globale di 11.500 miliardi, di cui l'86% riservato a tecnologie zero-emissioni.

L'Italia è da sempre all'avanguardia, in Europa e nel mondo, sul fronte delle energie rinnovabili. Le alternative ai combustibili fossili, infatti, rappresentano una quota importante della produzione energetica del paese, con una quota percentuale in costante crescita anno dopo anno.

La generazione di energia a zero emissione in Italia ha molte sfaccettature soprattutto in base alle caratteristiche del territorio e alla distribuzione delle risorse rinnovabili. L'idroelettrico è dominante dove il terreno presenta forti pendenze, come nell'arco alpino e in misura minore lungo la dorsale appenninica.

Il fotovoltaico è più presente al sud, grazie alla minore latitudine e all'insolazione maggiore.

L'energia eolica viene raccolta soprattutto nelle grandi isole, Sicilia e Sardegna, a cui si aggiunge in generale la parte meridionale della dorsale appenninica, a partire da Puglia, Campania e Basilicata.

L'energia geotermica, infine, ha come polo d'eccellenza la Toscana.

Secondo il rapporto Comunità rinnovabili redatto da Legambiente [15], si contano 7.776 comuni nei quali è installato almeno un impianto fotovoltaico, 7.223 con un impianto solare termico, 3.616 con sistemi a bioenergia, 1.489 in cui si sfrutta l'energia idroelettrica, 1.049 con impianti eolici e 594 in cui si approvvigiona anche tramite geotermia.

Circa un dodicesimo dell'energia totale prodotta in Italia, rinnovabile e non, deriva da impianti fotovoltaici. Una crescita impressionante, maturata in pochi anni, dovuta al crollo del costo dell'energia che, negli ultimi 10 anni, è sceso di più dell'80%. Oggi esistono modi diversi di sfruttare i raggi del sole. Oltre ai classici pannelli fotovoltaici, sono diffusi i sistemi solari termici, che sfruttano l'energia dei raggi solari per riscaldare l'acqua o un altro fluido.

L'energia solare, inoltre, è quella più impiegata anche per alimentare singoli dispositivi e strumenti: dai mezzi di trasporto che funzionano grazie a pannelli fotovoltaici fino a satelliti e veicoli spaziali, le applicazioni sono sempre più numerose e vantaggiose.

Degli oltre 320 TWh del fabbisogno energetico elettrico italiano ogni anno, più di un terzo arriva oggi da fonti rinnovabili. L'ultimo Rapporto sull'efficienza energetica dell'ENEA [18] parla in particolare di 110 TWh, equivalenti a circa una decina di milioni di tonnellate di petrolio.

All'inizio di questo secolo l'idroelettrico aveva il primato assoluto nell'energia elettrica green, coprendo da solo i 4/5 circa del comparto. Con il passare degli anni l'energia prodotta grazie al movimento dell'acqua è rimasta pressoché costante, mentre sono aumentate le quote delle altre fonti pulite. Secondo gli scenari più recenti, l'idroelettrico è al 42% del totale, con una crescita importante di tutte le rinnovabili.

Il secondo gradino del podio è al momento occupato dal fotovoltaico, con una quota parte del 20%. Al terzo posto si trova invece l'eolico, che al momento arriva al 16%.

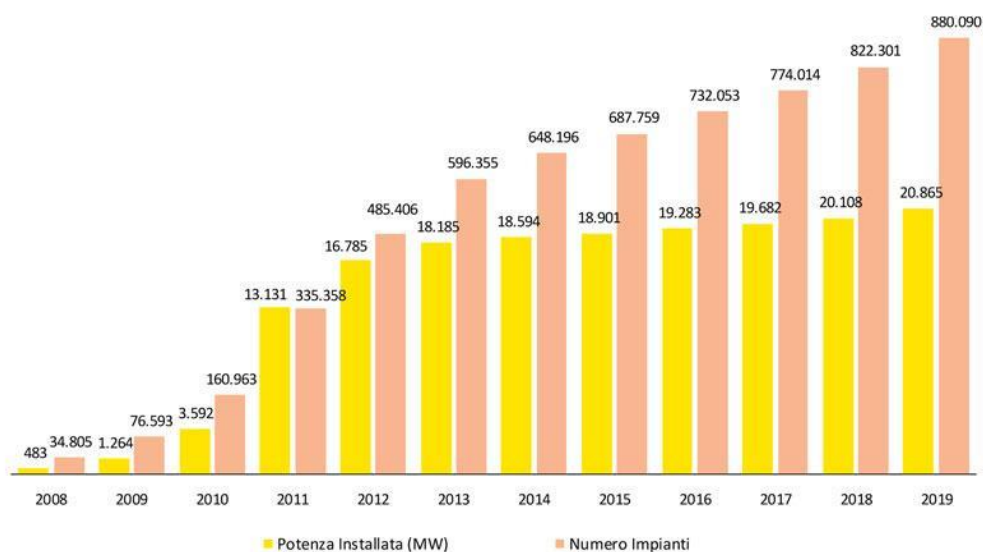
Percentuali sufficienti a fare dell'Italia un paese all'avanguardia nel contesto europeo, il terzo produttore di energia green in Europa, capace di generare da solo poco più del 10% del totale di rinnovabili a livello continentale.

Il GSE (Gestore Servizi Energetici) ha pubblicato da poco il rapporto statistico sul solare fotovoltaico relativo ai dati del 2019 [19].

Dai dati statistici risulta che in Italia, al 31 dicembre 2019, sono stati installati 880.090 impianti fotovoltaici, per una potenza complessiva pari a 20.865 MW, di cui:

- gli impianti di piccola taglia (potenza inferiore o uguale a 20 kW) costituiscono il 92% circa del totale in termini di numero e il 21% in termini di potenza;
- gli impianti di taglia media (potenza pari a 23,7 kW) che costituiscono le restanti percentuali.

Solo durante il 2019 sono stati installati oltre 58.000 nuovi impianti fotovoltaici, che hanno incrementato di 750 MW la potenza installata del Paese [3].



**Figura 1.1.1 Impianti installati in Italia tra il 2008 e il 2019 secondo GSE [3]**

Parlando a livello globale, il solare fotovoltaico (FV) sta attualmente registrando la crescita più rapida delle fonti rinnovabili grazie al calo dei costi dei progetti solari e ai crediti d'imposta governativi. Le fonti di energia rinnovabile, nell'ambito della generazione di elettricità, garantiscono molti vantaggi rispetto ai combustibili fossili, tra cui:

1. Disponibilità infinita: le fonti rinnovabili sono inesauribili e quindi sempre disponibili in natura. Questo significa che non ci sono problemi di scorte o approvvigionamento;
2. Sostenibilità ambientale: l'energia rinnovabile è pulita e non arreca danni all'ambiente né rilascia CO<sub>2</sub> in atmosfera;
3. Bassi costi: produrre energia rinnovabile è sempre più conveniente e in molti casi è finanziata da specifiche agevolazioni e incentivi;
4. Indipendenza energetica: l'aumento del ricorso alle rinnovabili consentirà ai Paesi di raggiungere una maggiore indipendenza energetica e a non dipendere da altri Stati per la fornitura di gas ed energia.



Il passaggio da un'energia basata sui combustibili fossili al fotovoltaico, in ambienti anche residenziali, ha contribuito ad un drastico aumento di reti di distribuzione a bassa tensione (BT).

A causa della scarsa pianificazione della rete e l'integrazione incontrollata, si sono presentate delle problematiche che sono diventate presto oggetto di ricerca; alcuni esempi: violazioni di tensione, sovraccarico, problemi di qualità dell'alimentazione.

Pertanto, i sistemi avanzati di gestione dell'energia sulle reti di distribuzione a bassa tensione sono fondamentali per un'intromissione efficace ed efficiente delle nuove tecnologie a zero emissioni di carbonio.

Le utility stanno rapidamente installando contatori avanzati o "intelligenti" che registrano il flusso di energia a intervalli precisi che vanno da cinque minuti a ogni ora, offrendo l'opportunità di implementare tecniche di monitoraggio del carico per identificare diversi profili di carico delle risorse energetiche distribuite. Si stima che le installazioni di contatori intelligenti raggiungeranno i 70 milioni entro l'inizio del 2017 e i 90 milioni entro il 2020 [17].

Fino ad oggi sono stati proposti diversi metodi per stimare la capacità di generazione di picco e la produzione in tempo reale dei sistemi solari. Ad esempio, le immagini satellitari e aeree sono state utilizzate per identificare i sistemi solari fotovoltaici e stimare le loro caratteristiche fisiche di dispiegamento (dimensioni, inclinazione, orientamento, ecc.).

Questo approccio può fornire solo una stima approssimativa della capacità di picco delle installazioni fotovoltaiche e non può stimare con precisione la generazione solare in tempo reale di produzione in una data area geografica. Questi metodi richiedono, per stimare la produzione fotovoltaica, la conoscenza della capacità totale installata dei sistemi FV in quell'area. Le tecniche di monitoraggio del carico non intrusivo NILM (Non-Intrusive Load Monitoring) possono contribuire alle strategie di gestione della domanda necessaria per il funzionamento in tempo reale delle reti intelligenti, nonché a ridurre gli impatti negativi della generazione distribuita e delle tecnologie a basse emissioni di carbonio nelle reti a bassa tensione [8].

## 1.2 Disaggregazione del carico

A causa della fluttuazione dell'energia fotovoltaica, dipendente dalle condizioni meteorologiche, le società di servizi elettrici hanno bisogno di previsioni accurate della produzione di energia solare per avere il giusto equilibrio tra combustibili fossili e rinnovabili disponibili.

Errori nelle previsioni potrebbero comportare ingenti spese per l'utenza a causa del consumo eccessivo di carburante o di acquisti di emergenza di elettricità da utenze vicine.

Attualmente, la maggior parte delle utenze domestiche è in grado di ottenere solo letture di potenza netta corrispondente alla produzione dei pannelli solari al netto del consumo elettrico, perciò i dati del contatore netto sono l'unico tipo di dati comunemente disponibile.

Tuttavia, l'impossibilità di differenziare la produzione dal consumo porta all'incertezza sulla quantità di elettricità da immettere in rete e sui costi inutili per l'utenza.

La disaggregazione solare è il problema della stima della generazione solare a partire dalle misurazioni del carico netto che possono essere ottenute a diversi livelli di aggregazione e consiste nel separare il consumo di energia misurato da un contatore intelligente in domanda domestica (o aziendale) e generazione solare.

Gli “*smart meter*” sono contatori o misuratori intelligenti che permettono di ricavare dati puntuali di consumo relativamente all'energia elettrica, al gas e all'acqua corrente. Nel caso dell'elettricità questi dispositivi monitorano i flussi di energia in entrata e in uscita di un'utenza dotata di impianto di produzione di energia rinnovabile [5].

Lo “*smart metering*” è il principio che sta alla base dei contatori intelligenti e che consente di attuare strategie di efficienza energetica; la misurazione e il monitoraggio intelligente dei dati di consumo sono indispensabili non solo alle società di servizi che distribuiscono energia e gas, ma anche ai consumatori al fine di renderli più consapevoli e di conseguenza più attivi nel migliorare la propria efficienza.

Una “*smart grid*” è l'insieme di una rete di informazione e di una rete di distribuzione elettrica, tale da consentire di gestire la rete elettrica in maniera "intelligente" sotto vari aspetti o funzionalità, ovvero gestendo in maniera efficiente per la distribuzione di energia elettrica e per un uso più razionale dell'energia, minimizzando, al contempo, eventuali sovraccarichi e variazioni della tensione elettrica intorno al valore nominale [4].

I vantaggi di questi sistemi sono numerosi per tutte le parti coinvolte:

- Riduzione dei costi di gestione delle letture e del contratto, operazioni che ora diventano praticabili da remoto;
- Frequenza superiore di lettura;
- Monitoraggio della rete e ottimizzazione della manutenzione in caso di perdite;
- Possibilità di concorrenza libera;
- Consapevolezza dell'utente in merito a consumi e sprechi, data dalla misurazione in tempo reale dei consumi;
- Miglioramento delle abitudini energetiche e aumento del risparmio energetico;
- Riduzione dei costi dell'energia per l'utente.

A titolo di esempio vengono riportati 3 grafici, estratti dall'articolo "*Solar Energy using Whole-House Consumption Signal*" [8], che descrivono i dati di potenza oraria di una stessa abitazione in cui non è stata applicata la disaggregazione, in 3 giorni diversi:

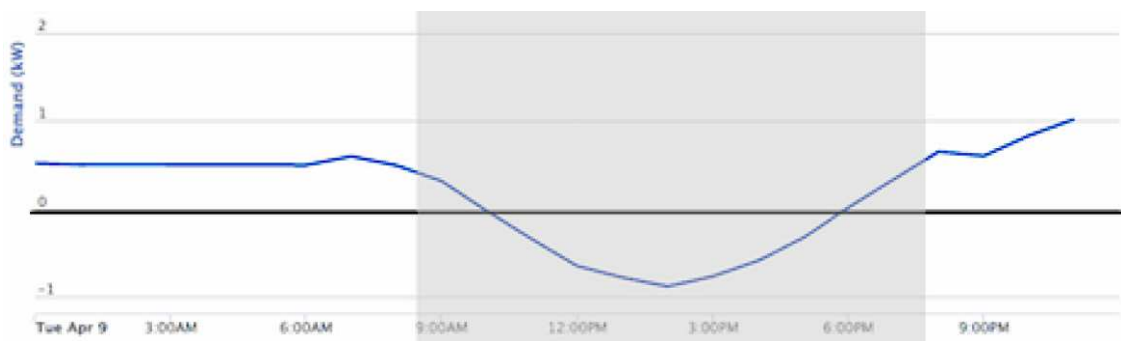
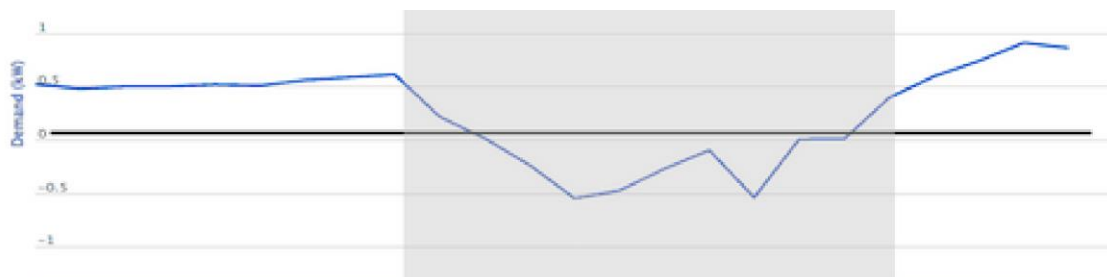
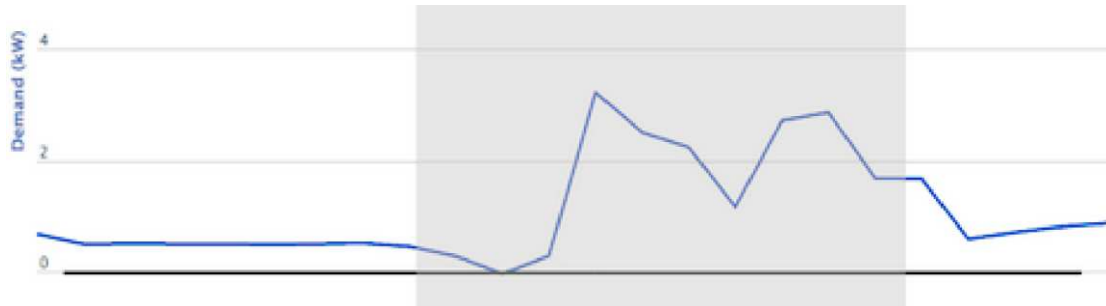


Figura 1.2.1 Giorno 1: Sole con eccesso di generazione durante il giorno



**Figura 1.2.2 Giorno 2: Senza disaggregazione non è possibile comprendere se fosse una giornata nuvolosa o se l'utente stesse utilizzando il carico durante il giorno**



**Figura 1.2.3 Giorno 3: Mentre c'era generazione la potenza netta era positiva. L'utente consumava più energia di quella prodotta durante il giorno.**

Si evince che non è possibile estrarre direttamente la generazione di energia fotovoltaica dal segnale di potenza netta sulla base degli andamenti nei dati [7].

## CAPITOLO 2

### Descrizione dell'algoritmo per la disaggregazione solare

L'apprendimento automatico (anche detto *machine learning*) è una branca dell'intelligenza artificiale che utilizza metodi statistici per migliorare la performance di un algoritmo nell'identificare pattern nei dati [6].

*Pattern* è un termine inglese, che significa "disposizione". Tuttavia, viene utilizzato per descrivere, a seconda del contesto, un "*disegno, modello, schema, schema ricorrente, struttura ripetitiva*" e, in generale, può essere utilizzato per indicare la ripetizione di una determinata sequenza all'interno di un insieme di dati grezzi oppure la regolarità che si osserva nello spazio e/o nel tempo [7].

Il riconoscimento di pattern è una branca dell'apprendimento automatico. Esso consiste nell'analisi e identificazione di pattern all'interno di dati grezzi al fine di identificarne la classificazione. La maggior parte della ricerca nel campo riguarda metodi di apprendimento supervisionato e non supervisionato.

Nell'ambito dell'informatica, l'apprendimento automatico è una variante della programmazione tradizionale nella quale in una macchina si predispone l'abilità di apprendere qualcosa dai dati in maniera autonoma, senza istruzioni esplicite.

Il *machine learning* è strettamente legato al riconoscimento di pattern e alla teoria computazionale dell'apprendimento ed esplora lo studio e la costruzione di algoritmi che possano apprendere da un insieme di dati e fare delle predizioni su questi, costruendo in modo induttivo un modello basato su dei campioni.

Scopo del presente studio è quello di implementare un algoritmo per disaggregare la generazione di energia dei sistemi fotovoltaici dalle misurazioni aggregate di una famiglia, attraverso l'elaborazione di dati provenienti da identificazione del carico non intrusivo (NILM).

Il metodo proposto può essere suddiviso di diverse fasi:

- Acquisizione dati;
- Pre-elaborazione;
- Addestramento dell'algoritmo.

L'identificazione del carico può essere classificata come intrusiva o non intrusiva, in funzione del numero di sensori utilizzati per monitorare i carichi in una rete elettrica. Il monitoraggio intrusivo richiede l'applicazione di un sensore per ogni apparecchio elettrico; perciò, fornisce profili di carico molto accurati. Le tecniche di monitoraggio non intrusivo (NILM), invece, richiedono meno dispositivi di monitoraggio grazie all'aggregazione dei carichi.

Gli algoritmi di apprendimento automatico sono fondamentali negli studi di ricerca che applicano i metodi NILM, in quanto implementano dati storici per separare un profilo di carico specifico dalle misurazioni aggregate.

Gli elementi principali di ogni progetto di *machine learning*, che concorrono a costruire un sistema che apprende delle relazioni tra i dati e le memorizza in un modello, sono:

- *Algoritmo*: un insieme di regole, generalmente basate su metodi statistici, per estrarre dai dati degli schemi ricorrenti;
- *Modello*: una rappresentazione di un contesto reale tramite l'algoritmo con opportuni parametri;
- *Dataset di training*: l'insieme dei dati forniti all'algoritmo che permettono la costruzione del modello;
- *Dataset di test*: insieme dei dati che a seguito della fase di training vengono utilizzati per valutare la qualità del modello in termini di precisione e accuratezza.

Gli algoritmi di *machine learning* sono molti e sono divisi in categorie:

- apprendimento supervisionato (Supervised) che verrà approfondito in questa sede;
- apprendimento non supervisionato (Unsupervised);
- apprendimento semi-supervisionato (Semi-Supervised);
- apprendimento per rinforzo (Reinforced Learning).

L'apprendimento supervisionato permette agli algoritmi di "imparare" dai dati storici/di addestramento e applicarli a input sconosciuti per ricavare l'output corretto. L'apprendimento supervisionato prevede diverse fasi:

1. La preparazione dei dati di addestramento dove ad ogni insieme di dati di input è associato il corretto output. L'assegnazione dell'output viene fatta con la supervisione umana, ma quasi sempre con sistemi di raccolta massiva dei dati con cui si acquisiscono sia le caratteristiche sia il corrispondente e corretto output;
2. La divisione dell'intero dataset in training e test in base alla strategia di addestramento;
3. L'addestramento dell'algoritmo sul training dataset; in questa fase si apprendono le relazioni tra features e output tramite cui sarà possibile fare previsioni con input diversi da quelli di addestramento;
4. Il test, la fase in cui si prova il modello sul dataset di test, che sono dati di input ignoti al momento dell'addestramento. Il confronto tra l'output vero e quello predetto dal modello consente la valutazione della qualità di questo secondo diverse metriche.

Gli algoritmi di apprendimento supervisionato sono divisi in due macro-classi:

- *Algoritmi di classificazione* in cui l'output, detto "label" o "etichetta", è una variabile qualitativa, ad esempio di tipo binario;
- *Algoritmi di regressione* in cui l'output è un valore numerico continuo, ovvero può assumere tutti i valori intermedi di un intervallo.

Oltre al tipo di output, gli algoritmi di classificazione e di regressione differiscono nella metrica per valutare la qualità del modello; nella classificazione si contano le etichette correttamente, i falsi positivi e i falsi negativi e si costruiscono dei coefficienti tipo accuratezza, precisione, specificità, sensibilità, etc, mentre nella regressione si valuta una funzione di errore, tra cui per esempio il *Mean Squared Error* o *errore quadratico medio*.

L'*apprendimento non supervisionato* avviene quando non si hanno a disposizione gli output relativi ai dati di input; il modello viene costruito solo coi dati di input e l'algoritmo scopre da solo le relazioni nascoste nei dati. Questo tipo di apprendimento si utilizza

quando si cerca una tendenza nascosta nei dati, si utilizza quando si dispone di un insieme anche molto voluminoso di dati a cui non è associato un valore di output o un'etichetta.

In questo capitolo argomenteremo per ordine tutti i passaggi del metodo proposto per implementare l'algoritmo, dalla fase di pre-elaborazione dei dati, fino alla selezione del metodo di regressione. Possiamo schematizzare quanto detto con il seguente flow chat [10]:

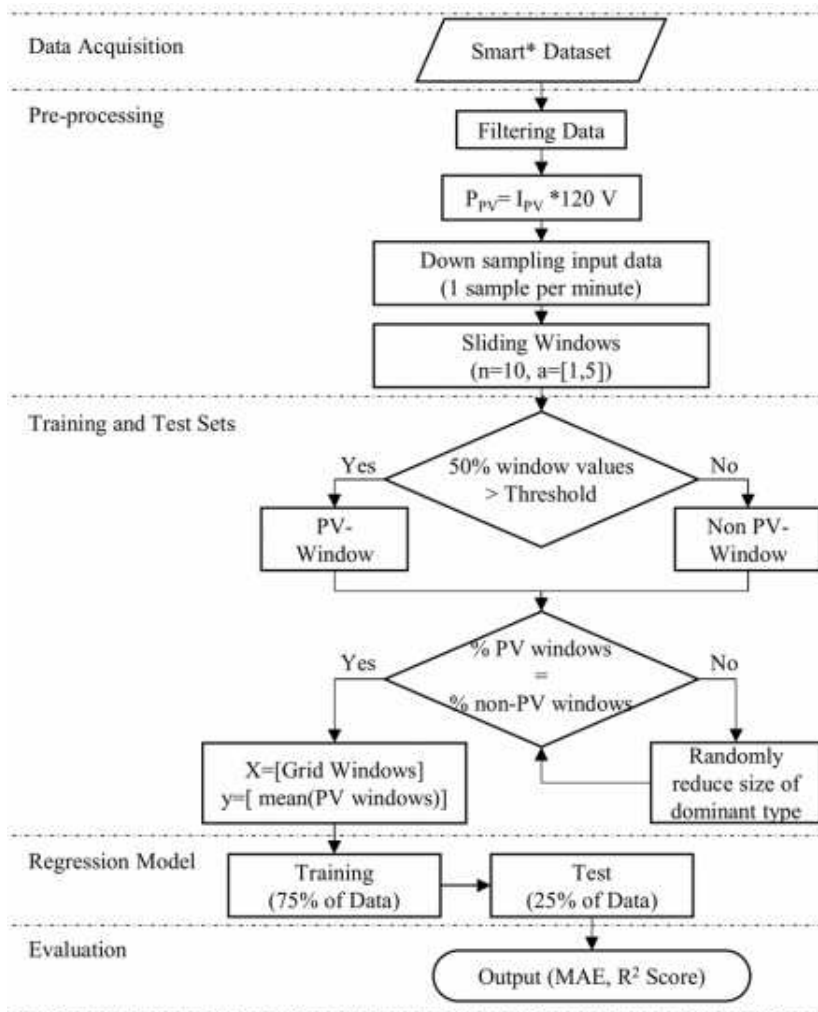


Figura 2: Flowchart algoritmo



## 2.1 Interpretazione set di dati

Utilizzeremo i set di dati pubblici del progetto Smart\* [11]. L'obiettivo di tale progetto è quello di ottimizzare i consumi energetici domestici. L'attenzione del progetto Smart\* si è concentrata sulla raccolta di quanti più dati possibili da ogni utenza domestica, piuttosto che sulla varietà, ovvero la raccolta di dati da quante più utenze possibili. I dati che sono stati acquisiti sono:

- l'utilizzo medio di elettricità domestica;
- l'utilizzo di ogni circuito e ogni carico della presa;
- generazione di elettricità da pannelli solari e turbine eoliche;
- dati meteorologici esterni;
- temperatura e umidità degli ambienti interni.

I dati utili all'implementazione dell'algoritmo sono la produzione di energia elettrica da impianti fotovoltaici e la domanda di energia di una famiglia residenziale. I dati ambientali sono stati scartati a causa della non corrispondenza temporale con le misurazioni della generazione dei pannelli.

## 2.2 Pre-processing

Come si può osservare nel flowchart, la fase di pre-processing consiste nella selezione delle funzionalità, che prevede due fasi: il filtraggio e la creazione di un profilo di carico aggregato virtuale.

Nella fase di filtraggio i dati grezzi provenienti dal set di dati pubblici Smart\* devono essere depurati dagli offset e i valori no-sense.

Nella seconda fase, si deve creare un profilo di carico aggregato virtuale ( $P_{grid}$ ), ottenuto sottraendo al consumo energetico dell'abitazione ( $P_{load}$ ) la generazione fotovoltaica ( $P_{PV}$ ), per ogni intervallo temporale.

$$P_{grid}(t) = P_{load}(t) - P_{PV}(t) \quad \forall t \in T \quad [W]$$

I dati ottenuti vengono utilizzati per addestrare e testare dei modelli di regressione.

Di seguito, nella Figura 2.2.1 viene rappresentato l'andamento della potenza  $P_{grid}(t)$  nell'arco di 48 ore:

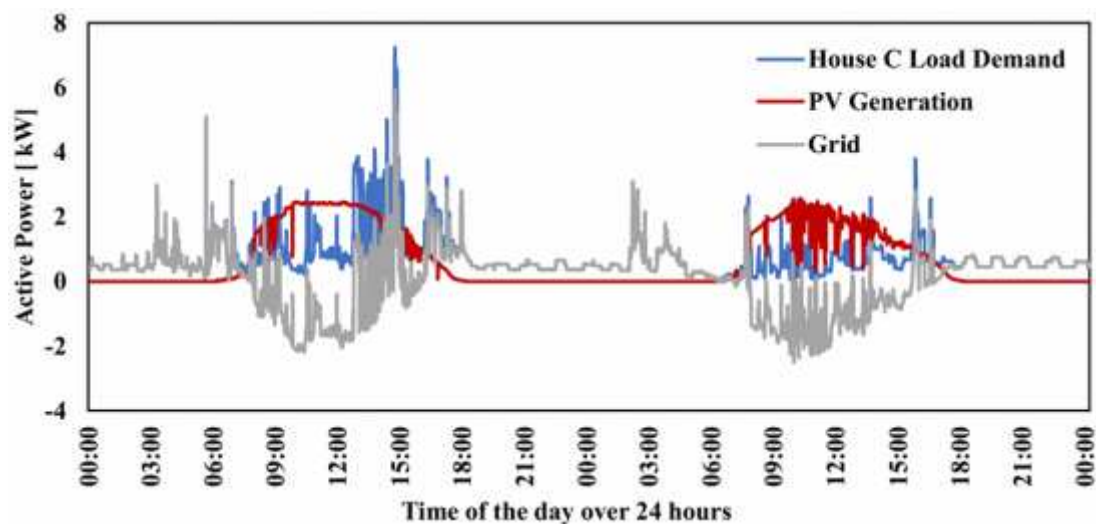


Figura 2.2.1: Creazione di profili di carico domestico, di generazione fotovoltaica e di rete applicando le fasi di pre-elaborazione

In linea con l'obiettivo di prevedere la potenza generata in tempo reale, invece di utilizzare le misurazioni a tempo pieno, utilizzeremo una tecnica chiamata “*finestra scorrevole*” che consiste nel raggruppare un numero definito ( $n$ ) di campioni consecutivi per ogni finestra sull'intero set di dati, considerando uno specifico valore di scostamento  $a$ , sempre minore di  $n$ .

Il valore ( $a$ ) coincide con il numero di campioni che le finestre stanno facendo scorrere.

A titolo di esempio, se la prima riga sarà formata unendo i primi ( $n$ ) campioni, la riga successiva sarà composta dai ( $a + 1$ ) campioni fino ( $n + a$ ) campioni.

Questa tecnica consente di non perdere eventi importanti nei bordi di ogni finestra.

Riportiamo una rappresentazione grafica del modello a finestra scorrevole nella Figura 2.2.2:

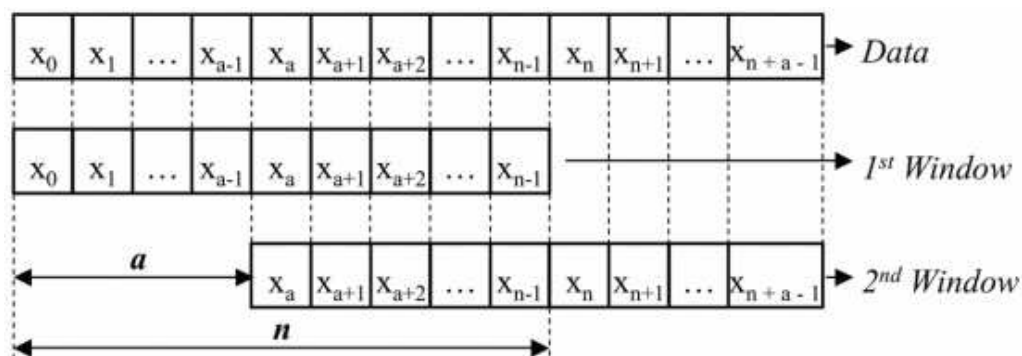


Figura 2.2.2: Processo della finestra scorrevole

## 2.3 Set di addestramento e test

L'algoritmo di apprendimento automatico richiede come input la matrice del profilo di rete, ossia le finestre dei carichi domestici meno la potenza del sistema fotovoltaico. L'output previsto invece è la potenza generata dall'impianto fotovoltaico.

Le finestre fotovoltaiche verranno create con la metodologia della finestra scorrevole, con uguali valori di  $(n)$  e  $(a)$ . Successivamente ogni finestra viene classificata come fotovoltaica (PV) o non fotovoltaica (NO\_PV) nella sua fase di generazione, se, dato un valore di soglia prestabilito per la potenza minima, la maggioranza delle misurazioni risulteranno superiori. Lo scopo è quello di scartare le misurazioni troppo basse dal set di dati originali.

Una volta scartati i valori non caratterizzanti, si andranno a confrontare i risultati relativi alle finestre PV con quelle NO\_PV, procedendo escludendo i dati relativi alla finestra con informazioni sovrabbondanti, in modo tale da avere un profilo di rete con uguali finestre PV e NO\_PV.

L'implementazione della tecnica di gestione dei dati per separare le due classi di finestre contribuisce ad ottenere un set di dati bilanciato adeguato ad addestrare un modello che risulti applicabile a qualsiasi evento in un'abitazione, riuscendo così ad

evitare di sovradimensionare il sistema e consentendogli di funzionare in uno scenario generale.

L'ultima fase riguarda il calcolo del valore di potenza fotovoltaica generata e da prevedere per ogni finestra. Per contenere l'elevata variabilità della generazione fotovoltaica è stata calcolata la media di ciascuna finestra fotovoltaica. Pertanto, la dimensione della matrice delle finestre fotovoltaiche è passata da un ordine ( $m \times n$ ) a un ordine ( $m \times 1$ ).

Riassumendo, l'input del modello di previsione è la matrice delle finestre fotovoltaiche e l'obiettivo dei modelli sono i valori medi della matrice PV. A questo punto si assegna una opportuna percentuale di valori per l'addestramento e un'altra a scopo di test, lasciando dei dati per valutare le prestazioni dei modelli di regressione.

## 2.4 Modelli di regressione

Le tecniche convenzionali di apprendimento automatico scelte per il nostro studio sono il *k-Nearest Neighbors (kNN)* e il *Random Forest*.

Il *kNN* non necessita di conoscenze preliminari sulla varianza dei dati; perciò, è ottimo per prevedere la generazione fotovoltaica da misurazioni aggregate.

Il *kNN* è un algoritmo utilizzato nel riconoscimento di pattern per la classificazione di oggetti, che si basa sulle caratteristiche degli oggetti vicini a quello considerato. Nei problemi di regressione viene presa la media dei *k-Nearest Neighbors* per fare una previsione su una classificazione; si utilizza per valori continui.

Nella fase di classificazione,  $k$  è una costante definita dall'utente e un vettore senza etichetta (una *query* o un punto di test) viene classificato assegnando l'etichetta più frequente tra i  $k$  campioni di addestramento più prossimi a quel punto di query.

Il valore  $k$  nell'algoritmo definisce quanti campioni verranno controllati per determinare la classificazione di un punto di query specifico.

Un esempio di classificazione mediante  $kNN$  è rappresentato nella Figura 2.4.1:

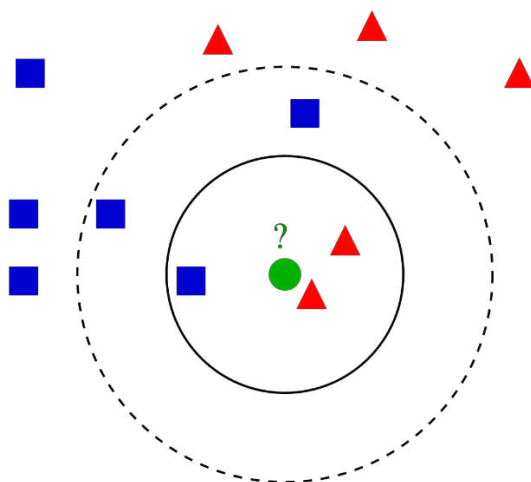


Figura 2.4.1: Esempio classificazione con modello  $kNN$

Le classi sono due:

- quella dei triangolini rossi;
- quella dei quadratini blu.

Se  $k = 3$  (cioè vengono considerati i 3 oggetti più vicini), allora il pallino verde viene inserito nella stessa classe dei triangolini rossi perché sono presenti 2 triangolini e 1 quadratino. Se  $k = 5$  allora viene inserito nella stessa classe dei quadratini blu perché sono presenti 3 quadratini e 2 triangolini.

Definire un valore di  $k$  scorretto può portare a problemi di overfitting o underfitting. Sottostimare  $k$  comporta una variabilità elevata, ma una bassa distorsione, viceversa, sovrastimare  $k$  può portare a una distorsione elevata e una variabilità inferiore. La scelta di  $k$  dipenderà in gran parte dai dati di input poiché i dati con più valori anomali o rumore probabilmente funzioneranno meglio con valori più elevati di  $k$  [12].

L'overfitting si verifica quando il modello si adatta eccessivamente ai dati di training. In pratica, il modello funziona bene sui dati di addestramento, quelli usati per costruire il modello, ma funziona male sui dati che non conosce (dati di test).

L'underfitting è un problema di apprendimento che si verifica quando la classificazione si basa su pochi parametri. La classificazione soffre di un'eccessiva discrepanza; è troppo semplice.

Gli algoritmi *kNN* sono in grado di utilizzare pesi uniformi per calcolare la media dei *Nearest Neighbors* e fornire una previsione. L'inverso della distanza può essere utile invece per il calcolo della previsione.

Definendo quindi un peso uniforme, i *Nearest Neighbors* vengono individuati tenendo conto della minore distanza euclidea.

La previsione può essere espressa matematicamente, definendo:

- $k$  il numero dei *Neighbors*;
- $\hat{y}_i$  la previsione  $i$  - esima;
- $\hat{y}_{iy}$  la previsione del  $j$  - esimo *Neighbors*, con  $0 < j \leq k$ .

$$\hat{y}_i = \frac{1}{k} \sum_{j=1}^k \hat{y}_{iy}$$

Il fattore più importante per massimizzare le previsioni del metodo è trovare un appropriato numero di *Neighbors* ( $k$ ).

L'algoritmo *Random Forest (RF)* si basa sugli alberi decisionali.

Gli alberi decisionali sono algoritmi che separano il dataset in più regioni e ad ogni input associano la corrispondente regione (o foglia) di appartenenza. L'algoritmo procede ad ogni passo separando i dati in base alla caratteristica che produce il miglior risultato e procede ricorsivamente fino alla classificazione di tutti i dati; la controparte è che se la profondità dell'albero non viene limitata è molto facile avere overfitting.

Nel *Random Forest* l'algoritmo costruisce più alberi su sottoinsiemi del dataset di addestramento selezionati casualmente, aggrega le previsioni di ciascun albero per scegliere poi la previsione migliore. Il *Random Forest* permette di valutare l'importanza delle caratteristiche, ovvero tra i dati di input è possibile dare una classifica di quanto sono importanti nella determinazione dell'output. Il *Random Forest* fa parte di un gruppo di algoritmi detti "Ensemble" perché lavorano su una combinazione di algoritmi di Machine Learning per realizzare un modello predittivo con migliori performance. In generale i metodi di tipo ensemble sono divisi in macrocategorie, tra cui *bagging* e *boosting* per la modalità di aggregazione. Il *bagging* consiste nel campionare  $N$  volte il

dataset iniziale e addestrare N volte lo stesso algoritmo, alla fine si aggregano i risultati facendo la media degli output in caso di regressione, la categoria con maggior frequenza nel caso di classificazione; è un metodo che aumenta i dati di addestramento per diminuire la varianza nelle previsioni. Il *boosting* è un processo sequenziale in cui ad ogni passo si migliora il modello precedente correggendone gli errori; ogni modello dipende dal precedente e tende a diminuire l'errore.

Si deve creare un numero definito di alberi (r) in funzione del numero di dati di addestramento. Il processo decisionale parte alla base di ogni albero e valutando le caratteristiche dei campioni, si troverà la decisione finale sui nodi della foglia.

Un esempio di schema decisionale basato sul *Random Forest* è rappresentato nella Figura 2.4.3:

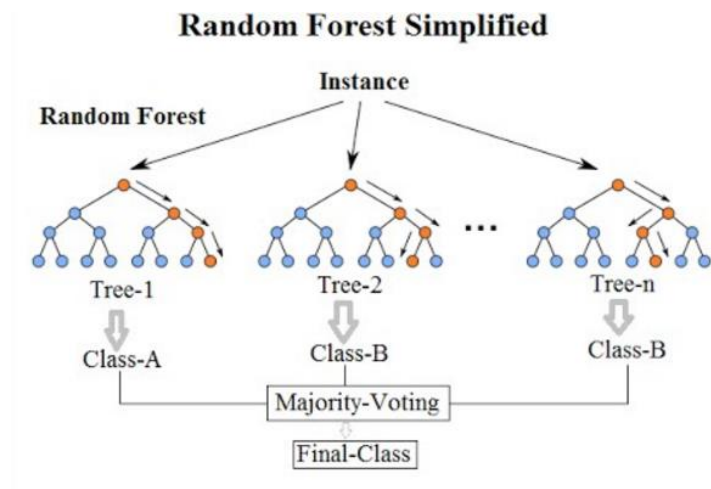


Figura 2.4.3: Schema decisionale basato sul RF

Anche in questo caso, la decisione è determinata come media di tutte le previsioni della foresta.

Matematicamente il *Random Forest* può essere espresso:

- r il numero degli alberi (stimatori);
- $\hat{y}_i$  la previsione i-esima;
- $\hat{y}_{iy}$  la previsione del j-esimo albero, con  $0 < j \leq r$ .

$$\hat{y}_i = \frac{1}{r} \sum_{j=1}^r \hat{y}_{iy}$$

## 2.5 Valutazione modelli

Per la valutazione dei modelli vengono utilizzate due metriche, *mean absolute error* (MAE) e *R-squared* ( $R^2$ ).

Il *MAE* fornisce una metrica dell'errore medio tra i valori attesi e no, fornendo pesi omogenei a tutti i singoli errori. Il *MAE* può essere calcolato come segue:

- $y_i$  è l'*i-esimo* valore vero;
- $\hat{y}_i$  è l'*i-esimo* risultato proiettato;
- $n$  è il numero di campioni.

$$MAE(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} |y_i - \hat{y}_i|$$

Il MAE può assumere valori compresi nell'intervallo  $[0, \infty]$  e più il valore atteso tende a 0, migliore saranno le prestazioni.

Il punteggio  $R^2$  stima la bontà di adattamento della tecnica di regressione. Questa metrica può essere calcolata:

- $(y_i - \hat{y}_i)$  è l'*i-esima* predizione
- $\bar{y}$  è il valore medio del vettore dei valori veri

$$R^2(y_i - \hat{y}_i) = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

Il punteggio può variare nell'intervallo compreso tra  $[0,1]$  e 1 corrisponde con il miglior punteggio ottenibile.



# CAPITOLO 3

## Sperimentazione

### 3.1 Descrizione dei dati

I dati che abbiamo selezionato appartengono al set di dati pubblico Smart\*, in particolare nel set di dati *UMass Smart\* Home Data Set* che contiene informazioni provenienti da 3 utenze domestiche (Home-A, Home-B, Home-C) [12]. In questo set sono disponibili dati elettrici, ambientali ed operativi. Le 3 utenze sono situate nel Massachusetts occidentale, per il nostro studio abbiamo selezionato Home-C.

Home-C ha una superficie di circa 325 m<sup>2</sup> e si sviluppa su due piani. A causa delle grandi dimensioni, ha richiesto due quadri elettrici separati per un totale di 60 circuiti. Nel set di dati sono stati monitorati il consumo di elettricità dell'intera abitazione e le condizioni meteorologiche esterne. Home-C è dotata di 3 pannelli solari, 2 microturbine eoliche e una batteria. I dati sono stati raccolti ad intervalli di 5 secondi.

I dati pubblicati da Smart\* relativi a Home-C ci hanno fornito 3 cartelle di misurazioni:

- Home-C-environmental;
- Home-C-generation;
- Home-C-power.

Il dataset Home-C-environment, relativo ai dati meteorologici, abbiamo deciso di non considerarlo nel nostro algoritmo, poiché non vi è corrispondenza temporale con le misurazioni elettriche dell'abitazione.

Per i nostri scopi abbiamo selezionato 3 mesi di corrente elettrica generata dai pannelli fotovoltaici, dal *01-08-2011* al *31-10-2011*, appartenenti alla cartella generation, e 2 mesi e mezzo di potenza attiva consumata, dal *15-04-2012* al *31-06-2012*, appartenenti alla cartella power per addestrare e valutare la tecnica NILM supervisionata. I dati sono forniti in formato CSV.

## 3.2 Preparazione datasets

### A. PRE-ELABORAZIONE PER LA SELEZIONE DELLE FUNZIONI

L'approccio è stato implementato in Python utilizzando un processore Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz e 8,00 GB (7,87 GB utilizzabile) di RAM installata su un DELL a 3,2 GHz con 8 GB di memoria principale, utilizzando i pacchetti *numpy*, *pandas* e *scikitlearn* per l'analisi dei dati e i pacchetti *matplotlib* e *seaborn* per la visualizzazione.

Il primo passo necessario per la sperimentazione dell'algoritmo è stato quello di unire i dataset di Smart\* (originalmente divisi per singoli giorni) in un unico DataFrame tramite la libreria *Pandas*, come mostrato nella Figura 3.2.1:

```
def load_df(directory_name, column_names):

    all_files = [directory_name + file for file in os.listdir(directory_name) if file.endswith('.csv')]
    # Lista di dataframe
    df_list = []

    for file in all_files:
        df = pd.read_csv(file, names=column_names)
        df_list.append(df)

    # Creo un unico dataframe concatenando tutti i singoli presenti nella lista "df_list"
    df = pd.concat(df_list)
    return df
```

**Figura 3.2.1: Creazione di un dataset unico**

In questa prima fase ci siamo preoccupati di filtrare e creare il profilo di carico aggregato virtuale.

Per il primo ci siamo occupati del vettore (*df\_pv*), responsabile della generazione fotovoltaica. Abbiamo selezionato i dati dalla cartella Home C-generation, contenente informazioni sulle correnti generate dalle due turbine, dai tre pannelli e dalla tensione generata dalle batterie. I dati relativi alle turbine eoliche e all'accumulo di energia delle batterie li abbiamo trascurati, prendendo in considerazione solo la corrente elettrica generata dai pannelli.

Nella prima fase di raccolta dati abbiamo selezionato il pannello 1 e concatenato tutti i CSV dei 3 mesi di misurazioni, ottenendo una matrice di 1550053 righe e 1 colonna, riportata nella Figura 3.2.2. L'edificio residenziale è collegato ad un sistema split-phase convenzionale degli Stati Uniti, perciò, per convertire la corrente in potenza attiva abbiamo moltiplicato ogni CSV per 120V, così da ottenere la potenza generata del pannello.

<b>Campioni</b>	<b>Watt</b>
0	472.932
1	472.932
2	472.932
3	472.932
4	472.932
5	472.932
6	472.932
7	472.932
8	472.932
9	472.932
10	472.932
11	472.932
12	472.932
13	472.932
14	472.932
15	472.932
16	472.932
17	472.932
18	472.932
19	472.932
20	472.932
21	472.932
22	472.932
23	472.932
24	472.932
25	472.932

Figura 3.2.2: prime 25 righe di df\_pv

Nella Figura 3.2.3 mostriamo la potenza generata dai singoli pannelli, dove in ascissa riportiamo i singoli campioni e in ordinata la potenza:

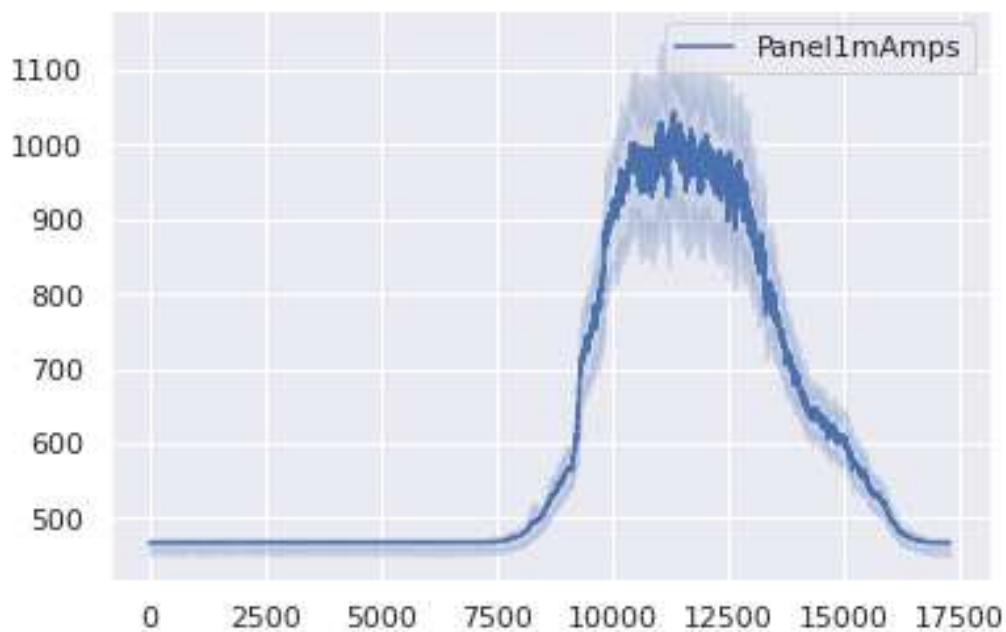


Figura 3.2.3: Andamento generazione fotovoltaica Home-C

Per il secondo vettore (*df\_load*) abbiamo selezionato i dati dalla cartella *HomeC-power*, contenente le misurazioni relative alla domanda di potenza dell'abitazione. Anche in questo caso abbiamo creato *df\_load* concatenando tutti i CSV dei 2 mesi e mezzo di misurazioni, ottenendo una matrice di 6727632 righe e 1 colonna, riportata in parte nella Figura 3.2.4:

Campioni	Watt
0	280.5
1	280.3
2	282.6
3	280.3
4	280.0
5	280.7
6	278.9
7	283.1
8	280.3
9	281.3

10	279.8
11	280.6
12	283.7
13	281.7
14	280.2
15	280.4
16	280.0
17	279.8
18	280.3
19	281.1
20	282.9
21	280.3
22	279.5
23	280.5
24	280.1
25	280.9

Figura 3.2.4: prime 25 righe di df\_load

Mostriamo nella Figura 3.2.5 la richiesta di potenza della Home-C, dove abbiamo riportato in ascissa i campioni e in ordinata la potenza:

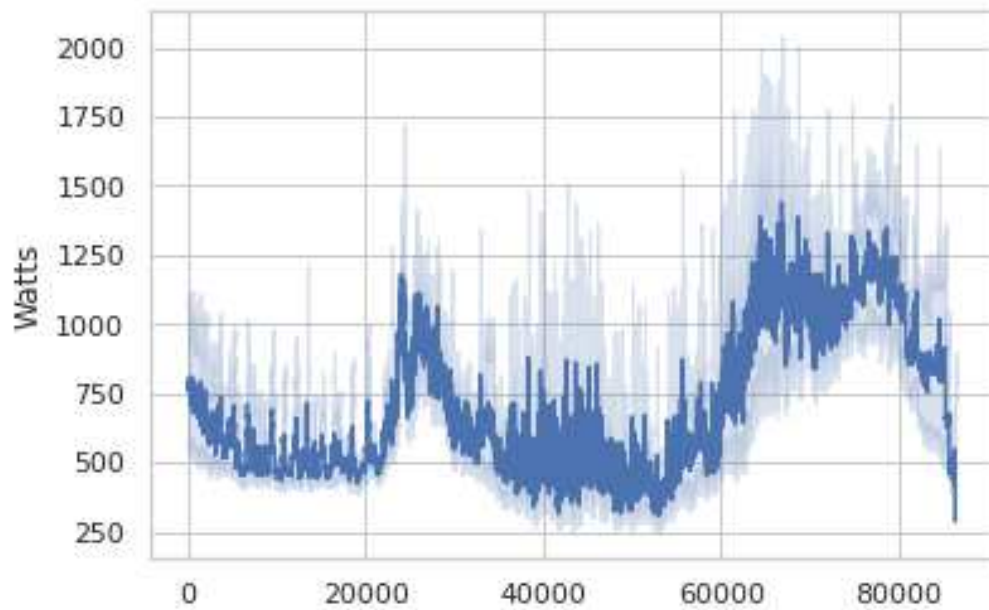


Figura 3.2.5: Andamento potenza richiesta Home-C

Per creare il profilo di carico simulato dell'abitazione, abbiamo sottratto entrambi i profili di carico.

Prima di procedere con l'operazione però, abbiamo osservato che i due profili di carico sono stati ottenuti come misurazioni indipendenti, quindi registrati in tempi differenti, perciò, per poterli sommare abbiamo dovuto ridurre la frequenza di entrambi per renderli dimensionalmente confrontabili e scartare le informazioni temporali dei dati eccessivi.

I dati appartenenti a  $df\_load$  sono stati ridotti da 60 campioni al minuto a 1 campione al minuto; invece, i dati relativi a  $df\_pv$  sono stati ridotti da 12 campioni al minuto a 1 campione al minuto. Nella figura 3.2.6 mostriamo l'andamento di  $df\_load$  in arancio e l'andamento di  $df\_pv$  in blu:

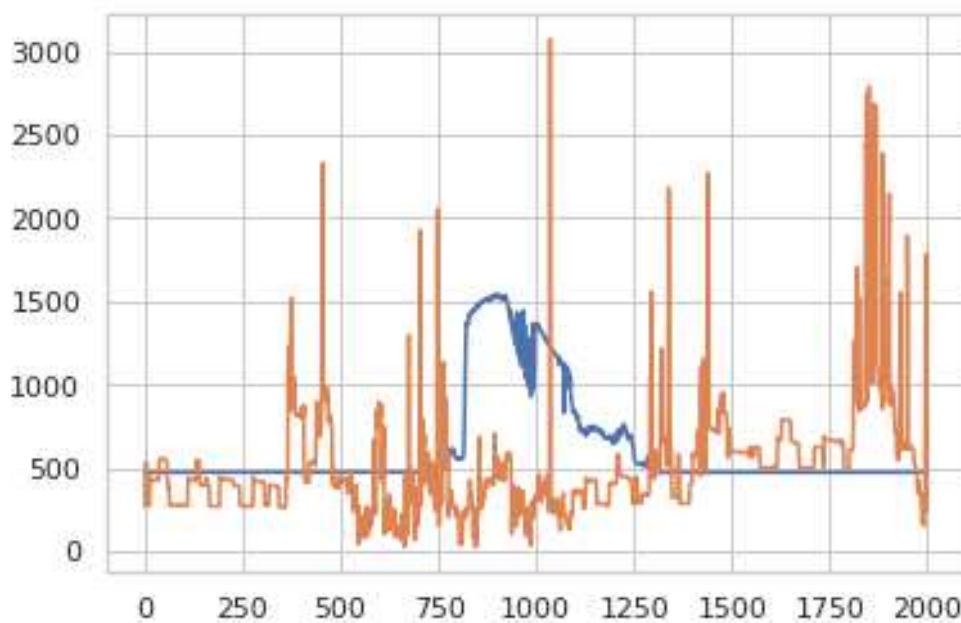


Figura 3.2.6: Andamento potenze  $df\_load$  e  $df\_pv$

A questo punto abbiamo potuto sommare i due vettori, ottenendo il profilo di carico virtuale ( $df\_grid$ ):

$$df_{grid}(t) = df_{load}(t) - df_{pv}(t) \quad \forall t \in T \quad [W]$$

Nella Figura 3.2.7 abbiamo rappresentato l'andamento di  $df\_grid$ , descritto dalla curva verde, confrontato con le due potenze  $df\_load$  in arancio e  $df\_pv$  in blu:

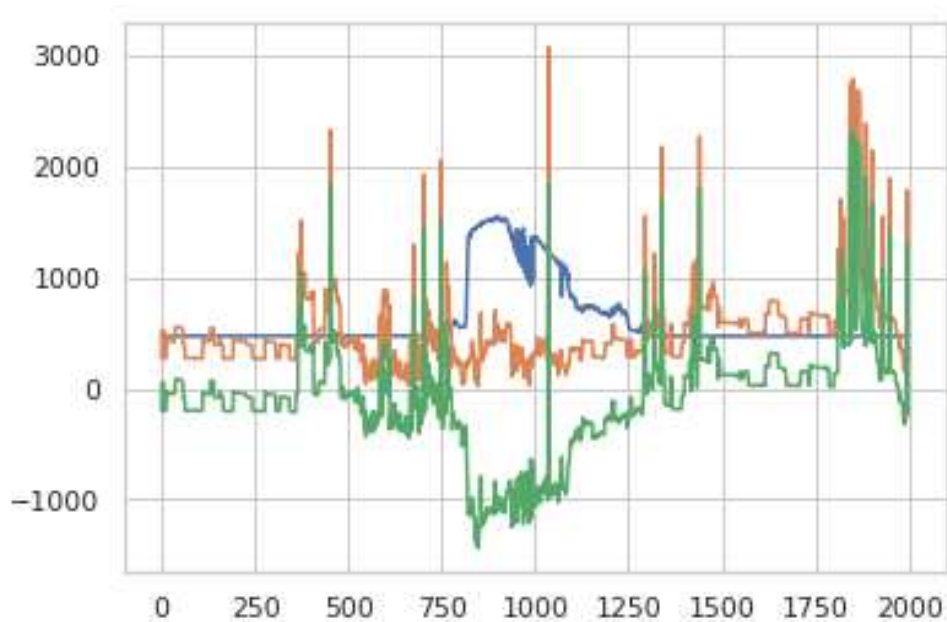


Figura 3.2.7: Andamento potenze *df\_load*, *df\_pv* e *df\_grid*

## B. FINESTRA SCORREVOLE

Per simulare la previsione in tempo reale della potenza generata dalle misurazioni aggregate, applichiamo sia ad *avg\_df\_load* che ad *avg\_df\_pv* la tecnica di finestra scorrevole descritta nel Cap.2.

Abbiamo posto  $n = 10$  e il valore di scorrimento  $a = 1$ , ottenendo così due matrici *mat\_load* e *mat\_pv* di ordine  $112119 \times 10$ . La struttura della funzione è mostrata nella Figura 3.2.8

```
import itertools
import numpy as np

def matrix(arr):
    row_number = len(arr) - 9
    col_number = 10
    result_matrix = [[0 for _ in range(col_number)] for _ in range(row_number)]

    for i, j in itertools.product(range(row_number), range(col_number)):
        index = i + j
        if (index < row_number+10):
            result_matrix[i][j] = arr[index]
    return np.array(result_matrix)

mat_load = matrix(avg_df_load)
mat_pv = matrix(avg_df_pv)
```

Figura 3.2.8: sliding window

### C. SET DI ADDESTRAMENTO E TEST

L'algoritmo richiede come input il profilo di rete, come output invece ci restituirà la potenza generata dall'impianto fotovoltaico.

Il profilo di rete ( $P_{grid}$ ) si ottiene sottraendo le matrici  $mat_{load}$  e  $mat_{pv}$ , ottenute utilizzando gli stessi valori di (n) e (a).

Le prime 10 righe della matrice funzione di rete le mostriamo nella Figura 3.2.9:

```

[[ -200.692      50.89466667   58.343      56.793      57.35633333
   56.44966667 -189.45533333 -201.467     -201.55033333 -201.487      ]
 [  50.89466667   58.343      56.793      57.35633333   56.44966667
 -189.45533333 -201.467     -201.55033333 -201.487     -201.46866667 ]
 [  58.343      56.793      57.35633333   56.44966667 -189.45533333
 -201.467     -201.55033333 -201.487     -201.46866667 -201.192      ]
 [  56.793      57.35633333   56.44966667 -189.45533333 -201.467
 -201.55033333 -201.487     -201.46866667 -201.192     -163.95033333 ]
 [  57.35633333   56.44966667 -189.45533333 -201.467     -201.55033333
 -201.487     -201.46866667 -201.192     -163.95033333 -25.81866667 ]
 [  56.44966667 -189.45533333 -201.467     -201.55033333 -201.487
 -201.46866667 -201.192     -163.95033333 -25.81866667 -38.56866667 ]
 [ -189.45533333 -201.467     -201.55033333 -201.487     -201.46866667
 -201.192     -163.95033333 -25.81866667 -38.56866667 -42.28033333 ]
 [ -201.467     -201.55033333 -201.487     -201.46866667 -201.192
 -163.95033333 -25.81866667 -38.56866667 -42.28033333 -44.032      ]
 [ -201.55033333 -201.487     -201.46866667 -201.192     -163.95033333
 -25.81866667 -38.56866667 -42.28033333 -44.032     -45.607      ]
 [ -201.487     -201.46866667 -201.192     -163.95033333 -25.81866667
 -38.56866667 -42.28033333 -44.032     -45.607     -46.46033333 ]]
```

Figura 3.2.9 matrice  $P_{grid}$  (prime 10 righe)

Successivamente abbiamo valutato ogni finestra della matrice  $P_{grid}$  per classificarla in finestra fotovoltaica contenente informazioni sul PV nella sua fase di generazione o PV non in generazione.

Per fare questo siamo partiti dalla matrice  $mat_{pv}$ ; abbiamo impostato un valore di soglia di 100 *Watt* e stabilito che una finestra fotovoltaica era in fase di generazione se almeno il 50% dei valori di una finestra (riga) era superiore al valore di soglia. Così facendo abbiamo evidenziato le misurazioni estremamente basse e rumorose. A questo punto abbiamo segnalato le medesime finestre nella matrice  $P_{grid}$ , identificando così le finestre PV e NO\_PV.



Per fornire all'algoritmo un set di dati bilanciato, le percentuali di finestre classificate come PV deve essere in numero uguale a quelle classificate come NO\_PV; quindi, confrontando le finestre abbiamo scartato quelle di tipo preponderante.

L'algoritmo che sintetizza i seguenti passaggi è riportato nella Figura 3.2.10:

```
# Tolgo tutte le righe aventi più del 50% degli elementi < 100
bool_matrix = np.less(P_pv, 100)

count_true = np.sum(bool_matrix, axis=1)

# Ottengo le righe da togliere
rows_to_delete = np.where(count_true >= (0.5 * P_pv.shape[1]))
n_rows_to_delete = len(rows_to_delete[0])
print(n_rows_to_delete)

# Tolgo da entrambi le matrici
P_pv = np.delete(P_pv, rows_to_delete, axis=0)
P_grid = np.delete(P_grid, rows_to_delete, axis=0)

# tolgo elementi fino ad arrivare al numero di pannelli tolti in precedenza

num_rows_to_delete = P_pv.shape[0] - n_rows_to_delete
random_rows = np.random.choice(P_pv.shape[0], num_rows_to_delete, replace=False)

P_pv = np.delete(P_pv, random_rows, axis=0)
P_grid = np.delete(P_grid, random_rows, axis=0)
```

**Figura 3.2.10** matrice *P\_grid* (prime 10 righe)

Fatto ciò, abbiamo scartato anche le corrispondenti righe nella matrice *P\_grid*, ottenendo di nuovo due matrici di uguale dimensione. Mostriamo nella Figura 3.2.11 l'andamento delle prime 2000 righe della matrice *P\_grid*:

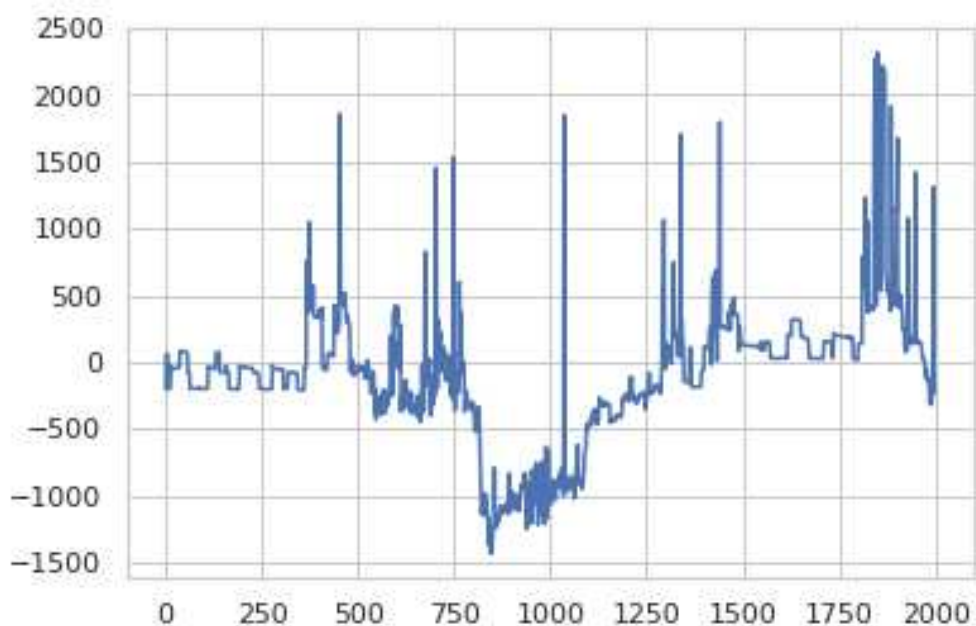


Figura 3.2.11: Andamento prima colonna della matrice P\_grid

Ora per quanto riguarda i dati da prevedere per ogni finestra, a causa della elevata variabilità della generazione fotovoltaica, abbiamo calcolato la media di ciascuna finestra della matrice *mat\_pv*, ottenendo così un vettore colonna (*avg\_matrix*), mostrato nella Figura 3.2.12:

```
[ 472.552  472.622  472.617  472.572  472.622  472.627  472.562
 472.572  472.652  472.567  472.582  478.6109 587.3827 545.0296
870.8822 671.6904 1354.689 926.95 758.2952 703.1545 472.567
472.567 472.582 472.577 472.577 472.642 472.782 472.817
472.817 472.797 472.837 472.807 472.762 490.6972 472.917
472.907 472.902 472.907 472.902 472.922 472.917 472.917
472.932 472.922 472.922 472.782 472.7219 472.7518 472.7818
472.737 472.812 472.757 472.817 472.832 472.8319 483.7323
631.4521 679.6112 715.2757 716.0592 919.5258 472.737 472.837
472.852 472.397 472.422 472.372 472.372 472.382 472.407
472.392 472.387 514.0083 1395.115 1317.7677 1005.2095 472.392
472.432 472.407 472.397 472.397 472.392 491.9345 750.3979
617.3934 474.9286 472.427 472.417 472.447 472.527 472.442
472.507 579.1613 646.8319 694.026 656.6759 480.3649 474.642
474.5674 472.337 ]
```

Figura 3.2.12: Matrice output addestramento dati *avg\_matrix* (prime 100 righe)

Mostriamo nella Figura 3.2.13 l'andamento della matrice *avg\_matrix*:

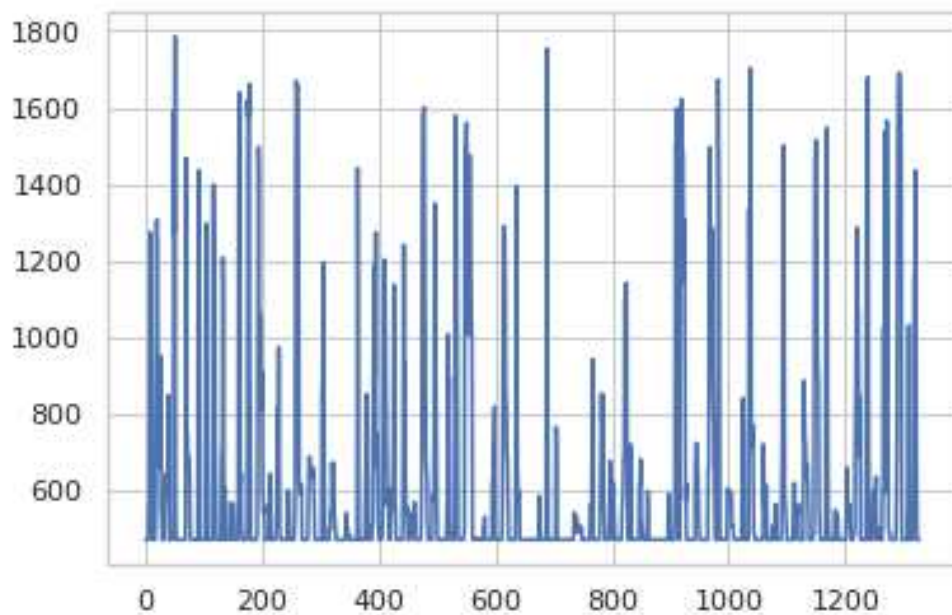


Figura 3.2.13: Andamento di *avg\_matrix*

Nominiamo quindi  $X$  la matrice  $P_{grid}$  di ingresso e  $y$  il vettore colonna *avg\_matrix* contenente i valori medi delle potenze dei pannelli, il 75% dei dati bilanciati lo utilizzeremo per l'addestramento, il restante 25% per i test.

Abbiamo effettuato questa divisione sfruttando *sklearn.model\_selection*, tramite la funzione *train\_test\_split*, come mostrato in Figura 3.2.14, ottenendo 995 campioni per il training e 225 per il test.

```
from sklearn.model_selection import train_test_split

X = P_grid
y = avg_matrix

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.75, random_state=42)
```

Figura 3.2.14

### 3.3 Modelli di regressione

Utilizziamo dei modelli di regressione per stimare con la maggior precisione possibile i valori delle variabili di output, partendo dai valori delle variabili in input.

I modelli di regressione che abbiamo scelto sono: *Nearest Neighbors regression*, *Random Forest* ed il *Support Vector Regression*.

Nel capitolo 2 paragrafo 4 sono già state riportate le caratteristiche dei modelli *Nearest Neighbors regression* e *Random Forest*. In questa sezione riporteremo le proprietà del *Support Vector Machine* che rappresenta un'evoluzione prodotta da noi rispetto all'articolo "*Photovoltaic Power Disaggregation using a Non-Intrusive Load Monitoring Regression Model*", pubblicato nel 2021[10].

Il *Support Vector Machine*: è un algoritmo che fornisce il miglior iperpiano tra tutti i possibili che separano i dati in classi; se l'insieme dei dati di addestramento ha (N) dimensioni, viene prodotto un iperpiano di (N-1) dimensioni che separa le classi.

L'algoritmo risulta molto efficace per modelli non lineari tramite l'uso dei *Kernel*, ovvero trasformazioni delle caratteristiche di input in spazi con un maggior numero di dimensioni in modo separare le classi sempre con un iperpiano. I *metodi kernel* si avvicinano al problema mappando i dati in uno spazio di caratteristiche multidimensionale, dove ogni coordinata corrisponde a una caratteristica dei dati dell'elemento, trasformando i dati in un insieme di punti dello spazio euclideo.

Questo algoritmo è molto efficiente con dataset a molte dimensioni; inoltre ottimizza l'uso di memoria perché di tutto il dataset apprende i cosiddetti "*Support Vector*", un sottoinsieme degli esempi di addestramento che identifica l'iperpiano separatore migliore.

Il grafico che segue mostra un esempio di SVM con due classi perfettamente separabili in  $\mathbb{R}^2$ :

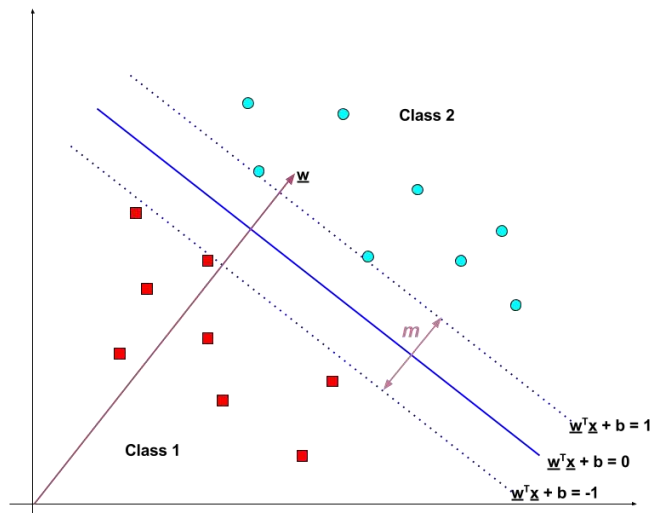


Figura 3.3.1: Esempio SVM

## 3.4 Risultati

Per valutare le prestazioni dei modelli di regressione usati, abbiamo utilizzato due metriche, l'*errore medio assoluto* (MAE) e il punteggio  $R^2$ .

Sia kNN che RF sono stati eseguiti utilizzando la libreria "*scikitlearn*" per i modelli di regressione di machine learning.

### A. valutazione del modello di regressione kNN

Determinare il giusto numero di vicini ( $k$ ) è fondamentale per massimizzare le prestazioni del modello di regressione ( $kNN$ ). Pertanto, abbiamo testato il nostro algoritmo con 4 differenti valori di  $k$ : 2, 3, 5 e 10.

Nella Figura 3.5.1 mostriamo le prestazioni del modello al variare del numero  $k$ :

K	Training size	Test size	MAE	R <sup>2</sup>
2	995	332	697.521	0.7544
3	995	332	702.825	0.7474
5	995	332	701.686	0.7492
10	995	332	723.707	0.7487

**Figura 3.5.1: Tabella prestazioni Knn**

Come si può osservare dalla tabella, il risultato migliore dei test lo abbiamo riscontrato per  $k = 2$ , in quanto ha fornito il minor MAE e il maggior  $R^2$ .

I risultati grafici del test sono presentati in Figura 3.5.2:

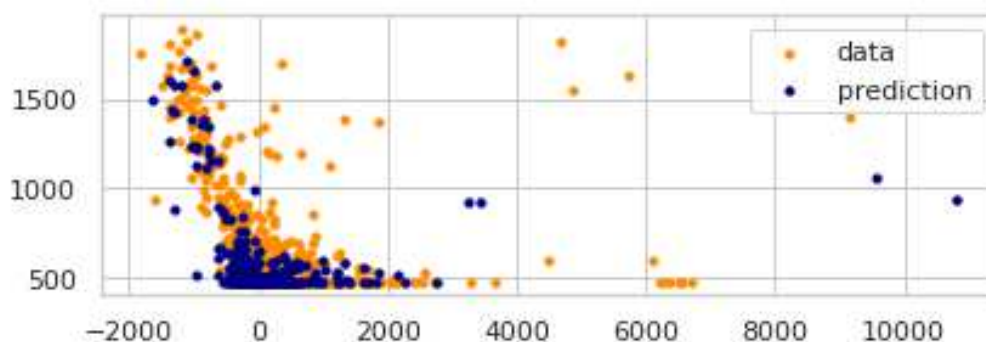


Figura 3.5.2: Prestazioni di regressione *kNN*

## B. valutazione del modello di regressione Random Forest

Anche in questo modello di regressione abbiamo testato diversi valori di  $r$  per poter ottenere la migliore efficacia del modello. Nella Figura 3.5.3 abbiamo riportato la tabella con i risultati dei test al variare di  $r$ :

$r$	Training size	Test size	MAE	$R^2$
100	995	332	672.455	0.8220
500	995	332	674.568	0.8231
1000	995	332	671.581	0.8263
2000	995	332	671.835	0.8260

Figura 3.5.3: Tabella prestazioni RF

Per il modello RF le migliori prestazioni sono state ottenute con  $r = 1000$  raggiungendo un  $MAE = 67.1581$  e  $R^2 = 0.8263$ .

La rappresentazione grafica del modello è mostrata nella Figura 3.5.4:

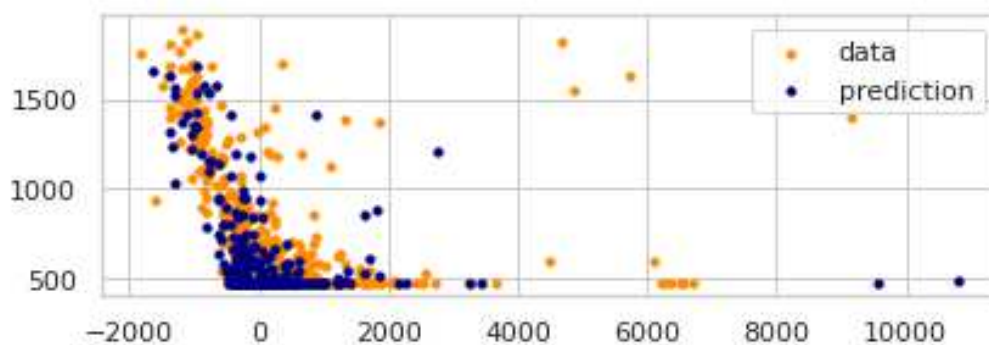


Figura3.5.4: Prestazioni di regressione *Knn*

### C. valutazione del modello di regressione SVR

Il *Support Vector Regression* lo abbiamo testato per diversi valori di  $C$ , ossia il parametro di regolarizzazione, per trovare il valore su cui lavora meglio il modello.

In Figura 3.5.5 riportiamo i risultati dei test:

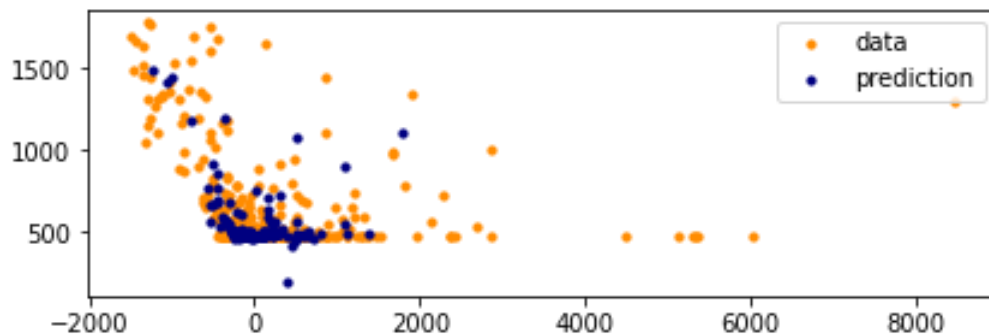
C	Training size	Test size	MAE	R <sup>2</sup>
1e1	995	332	1.249.305	0.2023
1e2	995	332	957.035	0.5194
1e3	995	332	1.010.333	0.5284
1e4	995	332	1.120.396	0.3752

Figura3.5.5: Tabella prestazioni SVR

In questo modello SVR le migliori prestazioni le abbiamo ottenute con  $C = 1e2$ , ottenendo un  $MAE = 95.7035$  e  $R^2 = 0.5194$ .

La rappresentazione grafica del modello è mostrata nella Figura 3.5.6:





**Figura3.5.6: Prestazioni di regresione SVR**

In conclusione, il *Random Forest* ci ha fornito una migliore previsione dei valori sconosciuti in termini di metriche. Al contrario del SVR, che è risultato il meno prestante dei tre modelli, ottenendo il *MAE* maggiore e l' $R^2$  minore. Nel complesso però tutti i modelli di regressione hanno prodotto buone prestazioni per la previsione di valori sconosciuti della generazione fotovoltaica delle misure aggregate della Home-C.

## CAPITOLO 4

# Conclusioni e possibili applicazioni dell'algoritmo

Questo documento ha introdotto un nuovo metodo di regressione NILM a bassa complessità per la generazione di energia fotovoltaica su tetto. L'approccio NILM proposto è stato applicato utilizzando un algoritmo di apprendimento automatico supervisionato che mostra un'efficacia ottimale utilizzando un set di dati a bassa frequenza campionato a un tempo di report di 1 minuto. Possiamo affermare che il *Random Forest* con 1000 stimatori e un passo della finestra scorrevole di 1 campione è riuscito a fornire i risultati più prestazionali rispetto agli altri modelli di regressione (kNN e SVR).

I risultati ottenuti hanno offerto l'opportunità di includere il metodo NILM proposto come funzione di misurazione avanzata. L'infrastruttura di misurazione avanzata (AMI) è un'impostazione di misurazione dell'utilità che aiuta nella comunicazione bidirezionale tra diverse applicazioni e i rispettivi fornitori di servizi.

Il metodo NILM veloce e di bassa complessità viene quindi presentato come un algoritmo realistico da implementare nei contatori intelligenti. Le funzioni avanzate dei contatori intelligenti sono necessarie per un'integrazione agevole sia dei sistemi fotovoltaici, che di qualsiasi altra risorsa energetica distribuita nelle reti di distribuzione a bassa tensione. Ciò contribuisce ad aumentare la flessibilità, l'affidabilità e la resilienza di questo lato del sistema elettrico alle dinamiche già mutevoli dalle reti convenzionali alle *smart grid*.

L'algoritmo può aiutare le aziende di servizi pubblici non solo a leggere il consumo energetico di una famiglia, ma anche a informarle sull'esistenza di un'elevata penetrazione degli impianti fotovoltaici in determinate aree, contribuendo a risolvere in modo più efficiente i problemi tecnici causati da questo tipo di tecnologie.

In assenza di strumentazione elaborata oltre la sottostazione di distribuzione, le tecniche di disaggregazione solare possono aiutare i servizi pubblici a comprendere

meglio la generazione solare complessiva, identificare i punti caldi e aggiornare le proprie apparecchiature.

Il lavoro futuro dovrebbe focalizzarsi sulla valutazione del metodo proposto con un insieme di dati con frequenze di campionamento superiori a 1 *Hz*.

Anche l'aggiunta di reti neurali artificiali a bassa complessità e l'implementazione di variabili non convenzionali come input del set di addestramento come il tempo o la temperatura esterna, potrebbe contribuire a fornire un'analisi comparativa di diversi metodi per massimizzare l'esecuzione di semplici tecniche per l'identificazione di carichi complessi come gli impianti fotovoltaici.



# Appendice A

Come descritto nel paragrafo, per classificare una finestra come *PV* o *NO\_PV* nella sua fase di generazione abbiamo stabilito un valore soglia di 100V per eliminare le misurazioni eccessivamente basse e rumorose dai dati originali, per poi scartare le corrispondenti righe nella matrice *P\_grid*. Abbiamo utilizzato la libreria *Numpy*:

```
# Tolgo tutte le righe aventi più del 50% degli elementi < 100
bool_matrix = np.less(P_pv, 100)

count_true = np.sum(bool_matrix, axis=1)

# Ottengo le righe da togliere
rows_to_delete = np.where(count_true >= (0.5 * P_pv.shape[1]))
n_rows_to_delete = len(rows_to_delete[0])
print(n_rows_to_delete)

# Tolgo da entrambi le matrici
P_pv = np.delete(P_pv, rows_to_delete, axis=0)
P_grid = np.delete(P_grid, rows_to_delete, axis=0)

# togliere tot elementi random fino ad arrivare al numero di pannelli tolti in precedenza

num_rows_to_delete = P_pv.shape[0] - n_rows_to_delete
random_rows = np.random.choice(P_pv.shape[0], num_rows_to_delete, replace=False)

P_pv = np.delete(P_pv, random_rows, axis=0)
P_grid = np.delete(P_grid, random_rows, axis=0)
```

**Figura A1**

## Appendice B

Come descritto nel paragrafo 3.3, abbiamo utilizzato dei modelli di regressione per stimare con la maggior precisione possibile i valori delle variabili di output, partendo dai valori delle variabili in input. Il primo modello testato è stato il kNN

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_absolute_error, r2_score

model_knn = KNeighborsRegressor(n_neighbors=2, algorithm="auto")

model_knn.fit(X_train, y_train)

y_pred = model_knn.predict(X_test)

r2_knn = r2_score(y_test, y_pred)
mea_knn = mean_absolute_error(y_test, y_pred)

print(f"r^2 score: {r2_knn}")
print(f"MEA: {mea_knn}")
```

**Figura B1**

## Appendice C

Il secondo modello testato è stato il RF, che ha fornito le migliori prestazioni di tutti i modelli:

```
from sklearn.ensemble import RandomForestRegressor

model_rf = RandomForestRegressor(n_estimators=2000)
model_rf.fit(X_train, y_train)

y_pred = model_rf.predict(X_test)

r2_rf = r2_score(y_test, y_pred)
mea_rf = mean_absolute_error(y_test, y_pred)

print(f"r^2 score: {r2_rf}")
print(f"MEA: {mea_rf}")
```

**Figura C1**

## Appendice D

Per ultimo abbiamo utilizzato il modello SVR che ha fornito le prestazioni peggiori in confronto ai modelli kNN e RF:

```
from sklearn.svm import SVR
svr = SVR(kernel="rbf", C=1e3)

svr.fit(X_train, y_train)

y_pred = svr.predict(X_test)

r2_SVR = r2_score(y_test, y_pred)

mea_SVR = mean_absolute_error(y_test, y_pred)

print(f"r^2 score: {r2_SVR}")
print(f"MEA: {mea_SVR}")
```

**Figura D1**





# Riferimenti

- [1]: <https://www.scienzainrete.it/articolo/trasformare-drasticamente-settore-energetico-azzerare-le-emissioni-nel-2050/jacopo>
- [2]: <https://www.ilsole24ore.com/art/energia-entro-2030-italia-90percento-sara-fonti-rinnovabili-AEWaqyFF>
- [3]: <https://biblus.acca.it/rapporto-gse-del-solare-fotovoltaico-2019/>
- [4]: <https://www.lumi4innovation.it/smart-grid-cose-e-cosa-significa/>
- [5]: <https://www.regalgrid.com/magazine/smart-meter-scopri-come-funzionano-i-contatori-intelligenti/>
- [6]: <https://www.intelligenzaartificiale.it/machine-learning/>
- [7]: [https://dbpedia.org/page/Pattern\\_\(disambiguation\)](https://dbpedia.org/page/Pattern_(disambiguation))
- [8]: [http://nilmworkshop.org/2014/proceedings/mohan\\_solar.pdf](http://nilmworkshop.org/2014/proceedings/mohan_solar.pdf)
- [9]: <https://dl.acm.org/doi/epdf/10.1145/3427771.3429387>
- [10]: <https://ieeexplore.ieee.org/abstract/document/9640183>
- [11]: <https://traces.cs.umass.edu/index.php/Smart/Smart>
- [12]: <https://lass.cs.umass.edu/papers/pdf/sustkdd12-smart.pdf>
- [13]: <https://www.ibm.com/it-it/topics/knn>
- [14]: <https://dl.acm.org/doi/abs/10.1145/3077839.3077848>
- [15]: [https://www.legambiente.it/wp-content/uploads/2021/11/Comunita-Rinnovabili-2022\\_Report.pdf](https://www.legambiente.it/wp-content/uploads/2021/11/Comunita-Rinnovabili-2022_Report.pdf)
- [16]: <https://dl.acm.org/doi/abs/10.1145/3077839.3077848>
- [17]: <https://dl.acm.org/doi/epdf/10.1145/3427771.3429387>
- [18]: <https://www.energiaefficienza.enea.it/pubblicazioni/raee-rapporto-annuale-sull-efficienza-energetica/rapporto-annuale-sull-efficienza-energetica-2022.html>
- [19]: [https://www.gse.it/documenti\\_site/Documenti%20GSE/Rapporti%20statistici/Solare%20Fotovoltaico%20-%20Rapporto%20Statistico%202021](https://www.gse.it/documenti_site/Documenti%20GSE/Rapporti%20statistici/Solare%20Fotovoltaico%20-%20Rapporto%20Statistico%202021)



