Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Gestione fisica degli edifici ed interventi in caso di danno

Modellazione di oggetti cognitivi per la gestione energetica efficiente e la qualità indoor in edifici intelligenti

Emilio Greco, Antonio Francesco Gentile, Andrea Vinci, Danilo Cistaro

RT- ICAR-CS-20-01 **Marzo 2020**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 8-9C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sezione di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.icar.cnr.it*
– Sezione di Palermo, Via Ugo La Malfa, 153, 90146 Palermo, URL: *www.icar.cnr.it*

Indice generale

Introduzione

Il processo di identificazione dei danni e di valutazione delle condizioni delle strutture edilizie negli ultimi anni ha suscitato un interesse crescente nella comunità di ricerca perché molte strutture edilizie hanno raggiunto o stanno per avvicinandosi alla fine della loro vita utile a causa del deterioramento naturale dei materiali su lungo termine ed a causa di eventi estremi come i terremoti o semplicemente perché sottoposti a continui carichi o sollecitazioni, che ne riducono la loro vita utile. Se non viene eseguita una adeguata manutenzione in tempo, gli edifici possono subire crolli parziali o completi senza preavviso con conseguenti perdite di vite umane causando anche un grave impatto economico. Quindi l'identificazione del danno e la valutazione dello stato delle strutture edilizie è fondamentale durante il loro ciclo di vita, soprattutto quando l'edificio è vecchio o si sospetta che sia stato sottoposto a sovraccarichi eccessivi.

A causa del maggior numero di incidenti su edifici e ponti è diventato sempre più importante sviluppare metodi per il rilevamento del degrado o del danno delle strutture in modo da poter intervenire con azioni mirate di manutenzione.

Molte tecniche di rilevamento dei danni, ancora oggi in uso, consistono in approcci sperimentali come onde di stress, onde ultrasoniche, raggi X, onde acustiche, metodi termici o correnti parassite che costituiscono quell’insieme di metodologie di rilevamento strumentale non distruttive.

Tuttavia, l'applicazione di tali metodi richiede una conoscenza a priori dei possibili siti danneggiati ed inoltre possono essere applicati solo a porzioni della struttura, ovvero a quelle parti che sono accessibili. Un altro problema è che i risultati della valutazione strumentale sono spesso inconcludenti o difficilmente valutabili in quanto tale valutazione a volte può comportare la misurazione dello stress locale, le deformazioni e le accelerazioni attraverso sensori installati su alcuni elementi critici.

A causa di questi inconvenienti, i ricercatori hanno sviluppato metodi che possono essere applicati alla struttura a livello globale, in tempi più brevi, eliminando le difficoltà delle precedenti metodologie.

Le tecniche di identificazione del danno basate sulle vibrazioni (VBDD) si presentano come degli ottimi metodi globali e sono in grado di valutare la condizione dell'intera struttura con pochi punti di misura. Questa metodologia, anche se nota da diverso tempo, è diventata fattibile solo nel corso negli ultimi decenni grazie ai progressi ottenuti nel campo dell’elettronica e dell’informatica e nell’aumento della potenza di calcolo di sensori e dell’hardware di acquisizione dati.

Utilizzando i dati di vibrazione è possibile arrivare al rilevamento del danno in quanto il danno altera le proprietà fisiche della struttura come massa, rigidità e smorzamento, che a loro volta influenzano le sue caratteristiche dinamiche, vale a dire, frequenze, forme modali e smorzamento. Quindi analizzando la dinamica di una struttura soggetta a vibrazioni, possono essere dedotte diverse

informazioni strutturali sui danni, compresa la loro posizione e gravità. Ci sono diversi vantaggi nell'uso di questi metodi, tra questi c'è il fatto che, in molti casi, può essere sufficiente un numero limitato di sensori per il rilevamento della gravità del danno e della sua ubicazione. Inoltre, le apparecchiature di misura come accelerometri, estensimetri e l'hardware di acquisizione dati sono diventati relativamente compatti e poco costosi in questi ultimi anni.

Tuttavia, in pratica, le tecniche VBDD hanno una serie di problemi, tra questi vi è la sensibilità agli errori di misura, l'acquisizione di forme modali incomplete, la natura non univoca delle soluzioni e l'influenza dei fattori ambientali. Negli ultimi anni sono state condotte un gran numero di ricerche nel campo di identificazione dei danni basata sulle vibrazioni e sono stati sviluppati molti algoritmi. La maggior parte del lavoro svolto utilizza dati modali come frequenze naturali e forme modali. Fino a poco tempo fa il metodo basato sulle frequenze naturali era particolarmente popolare come strumento per identificare il danno. Questo perché le frequenze naturali sono facili da ottenere attraverso l'analisi strumentale. Tuttavia, i metodi di identificazione del danno basati esclusivamente su di essi generalmente sono utili per capire l'esistenza di un danno ma sono poco attendibili per l'identificazione della posizione, poiché danni prodotti in punti diversi possono produrre la stessa frequenza naturale. Inoltre, in molti casi le frequenze naturali si sono rivelate insensibili al danno strutturale, specialmente per danni di minore gravità e sono fortemente influenzate da fattori ambientali, come le fluttuazioni di temperatura o umidità che ne pregiudica l'applicazione sul campo. I vettori di forma modale associati a ciascun modo di vibrare sono un'altra proprietà dinamica di base di una struttura. Questi sono correlati alla massa, alla rigidezza e allo smorzamento di un sistema e diventano quindi un elemento di interesse quando il danno modifica uno di questi parametri.

Le forme modali sono proprietà intrinseche di una struttura, non dipendono dalle forze o dai carichi che agiscono su di essa. I cambiamenti nelle forme modali sono molto più sensibili ai danni locali rispetto ai cambiamenti nelle frequenze naturali e nei rapporti di smorzamento.

Un grande vantaggio dei metodi di rilevamento dei danni basati sulla forma modale, essendo una proprietà spaziale, rispetto a quelli basati sulla frequenza è la capacità di fornire informazioni sull'esistenza e sulla posizione del danno. Tuttavia, le forme modali sono difficili da misurare e può essere necessario un gran numero di posizioni di misura per caratterizzare accuratamente i vettori delle forme modali e fornire una risoluzione sufficiente per la determinazione della posizione del danno. I metodi basati sulla forma modale presentano, dunque, la maggior parte delle stesse limitazioni dei metodi basati sulla frequenza. In altre parole deve essere rilevabile una quantità significativa di danno per rilevare un effetto sulle forme modali e molti tipi di danno potrebbero non essere quantificate.

I metodi VBDD più recenti sono essenzialmente basati sull'utilizzo dell'intelligenza artificiale. Questi metodi risultano adatti per rilevare i danni utilizzando misurazioni ottenute in modalità online in quanto fanno uso di strumenti di alta precisione, affidabili e di basso costo.

Poiché le reti neurali artificiali (ANN) sono note per la loro capacità di modellare relazioni non lineari e complesse, è possibile modellare tramite esse la relazione inversa tra le risposte strutturali e le caratteristiche strutturali.

Grazie alla loro eccellente capacità di riconoscimento dei modelli, la capacità di auto-associazione o correlazione, l'auto-organizzazione, autoapprendimento e per la semplicità di utilizzo nella modellazione non lineare le ANN hanno costituito un valido aiuto per superare alcune criticità dei tradizionali metodi di rilevamento dei danni e migliorare l'accuratezza e l'affidabilità degli stessi.

I metodi basati sulle ANN operano essenzialmente combinando i dati prodotti dal modello ad elementi finiti della struttura da testare e con i dati di misura reali. L'utilizzo delle ANN per il rilevamento dei danni presenta numerosi vantaggi tra cui l'insensibilità al rumore di misura e la possibilità di avere una diagnosi dei guasti attraverso una procedura automatizzata.

Tuttavia, le ANN di solito richiedono un enorme sforzo computazionale, specialmente quando sono coinvolte strutture con molti gradi di libertà. Per questo motivo, la maggior parte delle applicazioni ANN per il rilevamento dei danni sono limitate a piccole strutture con un numero limitato di gradi di libertà.

Modello dinamico della struttura

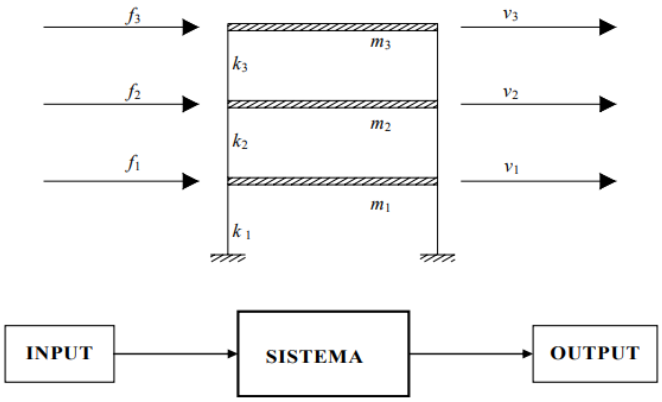
Come anticipato nell'introduzione le forme modali ricavate dall'analisi strumentale di una struttura sono rappresentative delle proprietà intrinseche della struttura stessa. Esse non dipendono dalle forze o dai carichi che vi agiscono. I cambiamenti nelle forme modali sono quindi molto più sensibili ai danni locali rispetto ai cambiamenti nelle frequenze naturali e nei rapporti di smorzamento. Pertanto i vettori di forma modale sono una proprietà spaziale e possono fornire informazioni sull'esistenza e sulla posizione del danno.

L'idea alla base delle applicazioni delle ANN nella rilevazione dei danni è quella di costruire un modello per fornire una relazione tra parametri modali e parametri strutturali attraverso un processo di apprendimento. Una volta stabilita la relazione, il modello addestrato è quindi in grado di rilevare i danni a partire dai dati modali.

In questo paragrafo cercheremo di fare un'anali preliminare sui dati che ci vengono forniti con l'intendo di capire come questi sono correlati. Ovvero analizzeremo la relazione che lega le proprietà fisiche della struttura come massa, rigidità e smorzamento, alle caratteristiche dinamiche, vale a dire,

frequenze, forme modali e smorzamento. Questo studio ci consente di comprendere meglio il modello che si vuole replicare su una ANN e quindi ci viene utile per guidarci nella scelta della tecnologia più adatta.

consideriamo pertanto l'analisi dinamica di una struttura semplice bidimensionale come la seguente:



Possiamo ipotizzare il moto della struttura come il moto di un pendolo inverso del tipo oscillatorio smorzato. Con buona approssimazione possiamo affermare che la legge fisica che caratterizza il sistema è la seguente:

$$\mathbf{M}\ddot{\mathbf{v}}(t)+\mathbf{C}\dot{\mathbf{v}}(t)+\mathbf{K}\mathbf{v}(t)=\mathbf{f}(t)$$

Dove K, C, M sono, rispettivamente, le matrici delle rigidezze, degli smorzamenti e delle masse, mentre v(t) è lo spostamento. La derivata di v(t) è la velocità e la derivata seconda l'accelerazione. Le frequenze naturali, gli smorzamenti e le deformate modali sono ricavate quindi dall'equazione omogenea associata (moto libero):

$$\ddot{x}+\frac{\beta}{m}\dot{x}+\frac{k}{m}x=0$$

Da cui, dividendo per m, si ha:

$$\ddot{x}+\frac{\beta}{m}\dot{x}+\frac{k}{m}x=0$$

Introducendo il parametro ζ, fattore di smorzamento ed ω_n pulsazione naturale si ottiene:

$$\ddot{x}+2\zeta\omega_n\dot{x}+\omega_n^2x=0$$

In definiva abbiamo visto che la relazione che lega le caratteristiche strutturali a quelli dinamiche è un'equazione differenziale del secondo ordine a coefficienti costanti, con soluzione del tipo:

$$x=e^{\lambda t}$$

Tuttavia le matrici delle rigidezze, degli smorzamenti e delle masse K, C, M, diventano incognite del problema, insieme allo spostamento. Questo perché per periodi di tempo molto lunghi le proprietà fisiche di una struttura variano, soprattutto in presenza di danno. Tuttavia la massa come abbiamo visto nell'equazione precedente può essere trascurata in questo tipo di analisi, pertanto le incognite che dovranno essere misurate o ricavate complessivamente diventano tre: Spostamento, Smorzamento e Rigidezza, o se consideriamo l'ultima equazione: pulsazione naturale, fattore di smorzamento e spostamento.

Pertanto, per modellare correttamente il problema abbiamo necessariamente bisogno di misurare almeno due parametri modali. Nel nostro studio abbiamo deciso di misurare e/o ricavare dal calcolo strutturale tre informazioni modali come riportato nella seguente tabella:

Modi di vibrare	**Frequenza**	**Smorzamento**	**Deformate modali**		
	Hz	%			
1° modo	2.44	3.00	+0.458	+0.814	+1.000
2° modo	6.50	2.51	+1.000	+0.310	-0.810
3° modo	9.69	3.00	-0.578	+1.000	-0.484

Anche se la struttura che andremo a monitorare è composta da due piani a cui corrispondono complessivamente sei modi di vibrare, precedenti studi hanno dimostrato che i primi tre modi di vibrare dell'intera struttura, sono sufficienti a determinare la collocazione del danno.

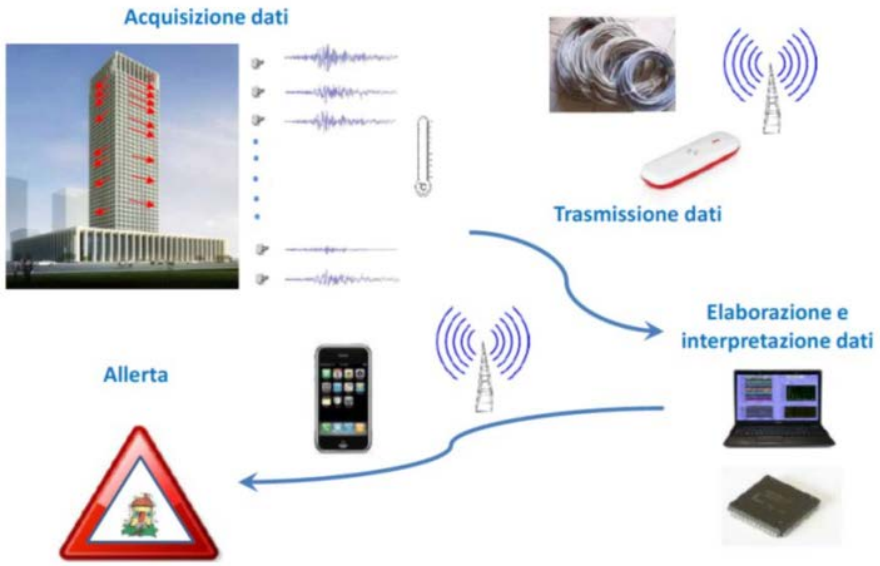
Architettura del sistema

L'SHM (STRUCTURAL HEALTH MONITORING) è una complessa pratica ingegneristica generata dall'unione di diversi strumenti tecnologici.

In generale, un tipico sistema di monitoraggio SHM è costituito dai seguenti sistemi:

- una rete cablata o wireless di sensori, montati sulla struttura, per il rilevamento di grandezze fisiche significative della risposta strutturale, delle azioni e delle condizioni ambientali;
- un sistema di acquisizione dei dati provenienti dai sensori;
- sistemi di trasmissione dei dati verso le unità di processo locali o remote;
- componenti software, più o meno complesse, per il pre-processamento, l'analisi e l'interpretazione dei dati (identificazione del danno), la valutazione della vita residua e il supporto alle decisioni;
- un sistema decisionale e di allerta per la gestione di situazioni di emergenza.

I primi due sistemi sono generalmente installati sulla struttura, gli altri invece sono dislocati all'interno degli uffici di controllo del gestore dell'infrastruttura. Tutti questi sistemi permettono all'ingegnere strutturale di inquadrare lo stato di integrità di una struttura.



I sensori sono dei dispositivi che vengono installati sulla struttura e costituiscono di fatto un "sistema nervoso" in grado di misurare i movimenti della struttura, come rotazioni, spostamenti assoluti e relativi, accelerazioni. Questi dispositivi presentano caratteristiche diverse sia per il campo di applicabilità sia per la precisione delle misure da essi fornite, nonché hanno costi differenti a seconda di quanto sono sofisticati. Ai sensori devono essere abbinati gli appositi software, attraverso i quali si riesce a mettere insieme ed interpretare in modo veloce la grossa mole di dati registrati e caratterizzare gli eventuali danni, prevedendone lo sviluppo. Inoltre, ci sono strumenti in grado di misurare paramenti ambientali come la temperatura interna o esterna, l'umidità e la forza o velocità del vento, che influenzano i risultati finali.

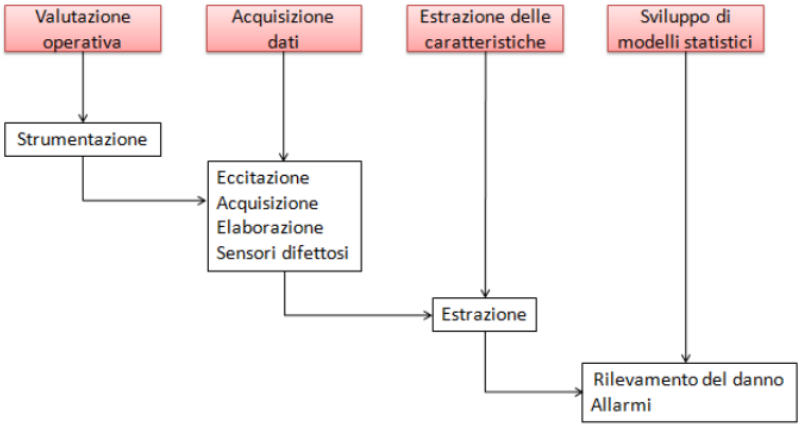
Per cui attraverso questi strumenti è possibile eliminare il cosiddetto rumore, cioè quegli spostamenti dovuti al variare della temperatura o dell'umidità, che non portano danni alla struttura. I sensori sono collegati a un sistema di acquisizione dati attraverso i cavi elettrici o via wireless. Il sistema di acquisizione dati permette di memorizzare la grande mole di dati provenienti dai sensori in modo automatico, secondo un intervallo di tempo preimpostato dall'operatore. Quest'ultimo è collegato ad un modem che permette la trasmissione dei dati ad un dispositivo remoto, ad esempio, un PC in un laboratorio, per essere analizzati e interpretati, cioè riceve i segnali generati dai sensori, li converte e li trasmette ai computer per l'elaborazione.

Più recentemente, sono stati sviluppati sensori che forniscono elaborazione e memorizzazione parziali dei dati. Dato che durante il monitoraggio viene impiegata una grande quantità di sensori,

l’acquisizione, la trasmissione e l’immagazzinamento dei dati rappresentano un’operazione molto delicata. Si registra quindi la necessità di sviluppare sistemi che vadano oltre il puro accumulo di dati. A tal scopo, interviene un software di gestione dei dati, che elabora tutti i dati arrivati dai sensori nel tempo e ne fornisce una risposta facile da analizzare per l’utilizzatore. Questi permettono di mettere insieme ed interpretare in modo veloce la grossa mole di dati registrati e caratterizzare gli eventuali danni, prevedendone lo sviluppo. Alcuni software più sofisticati hanno, in base al tipo di struttura, delle soglie che questa non deve superare e segnalano in automatico quando queste soglie sono state superate. I risultati finali forniti dai software possono essere visualizzati in forma tabellare o in forma grafica.

Le fasi del monitoraggio

Il processo di monitoraggio può essere suddiviso in quattro fasi:



1. Valutazione operativa;
2. Acquisizione, normalizzazione e pulizia dei dati;
3. Estrazione delle caratteristiche;
4. Sviluppo dei modelli.

La valutazione operativa tenta di rispondere a quattro domande relative alla capacità di identificare il danno e il suo sviluppo futuro:

- Quali sono le ragioni economiche e/o di salvaguardia della vita umana per eseguire il monitoraggio?

- Come viene definito il danno per il sistema monitorato e, in presenza di più danni, quali sono i casi più preoccupanti?
- Quali sono le condizioni, sia operative che ambientali, sotto le quali il sistema può funzionare?
- Quali sono le limitazioni nell'acquisizione dei dati nell'ambiente operativo?

La valutazione operativa stabilisce i limiti su ciò che verrà monitorato e su come verrà realizzato il monitoraggio, cioè la tecnica. Questa valutazione cerca di adattare il processo di identificazione del danno alle peculiarità del sistema cercando di acquisire conoscenze sulle caratteristiche del danno che dovrà essere rilevato.
Le motivazioni economiche e quelle di sicurezza, in termini di vita umana e strutturale, sono le linee guida per lo sviluppo di un sistema di monitoraggio completo ed affidabile in grado di garantire la funzionalità della struttura, durante tutto il periodo della sua vita utile.
La seconda questione messa in luce è quella relativa al danno: quando un sistema SHM viene implementato su di una struttura, è importante che si definisca e si quantifichi il danno che si sta cercando, facendo in modo che la probabilità con cui esso sarà rilevato sia sufficiente, impiegando in modo razionale ed intelligente le risorse a disposizione.
L’ acquisizione dati, invece, coinvolge la parte operativa del sistema di monitoraggio e prevede la selezione dei metodi di eccitazione, della tipologia, della quantità, e della posizione dei sensori, dell'hardware di acquisizione, memorizzazione, trasmissione dei dati. Ancora una volta, questo processo sarà specifico dell'applicazione. In questa fase, le considerazioni economiche svolgono un ruolo importante sulle decisioni che saranno prese. L'intervallo in cui i dati dovrebbero essere raccolti è un'altra considerazione importante che deve essere fatta, in quanto va ad influenzare la quantità di memoria necessaria per l’acquisizione.
Essendo, inoltre, i dati raccolti sotto diverse condizioni è necessario effettuare una loro normalizzazione al fine di identificare univocamente il processo di danno. Per normalizzazione dei dati si intende il processo di separazione delle modifiche, introdotte nei dati raccolti dal sensore, causate dai danni da quelle causate dalla variazione delle condizioni operative e ambientali. Una delle procedure più comuni è quella di normalizzare la risposta misurata sulla base di input noti.
Quando la variabilità ambientale e operativa rappresenta un problema, può sorgere la necessità di normalizzare i dati in fasce temporali soggette agli stessi cicli operativi e a simili condizioni ambientali così da poter permettere un confronto delle misurazioni effettuate. Le fonti di variabilità nel processo di acquisizione dei dati devono essere identificate e minimizzate il più possibile. Va osservato, però, che non tutte le fonti di variabilità posso essere eliminate perciò non tutti i dati possono essere normalizzati. Anche quando non è possibile una normalizzazione dei dati va

comunque effettuato uno studio statistico su come le condizioni operative ed ambientali possano influire sulle risposte della strumentazione del monitoraggio. Infine, si ha la pulitura dei dati che è un processo di scelta selettiva dei dati da trasmettere al processo di estrazione delle caratteristiche della struttura.

Il processo di pulitura dei dati si basa generalmente sulle conoscenze acquisite da personale specializzato e direttamente coinvolto nella acquisizione dei dati. A tal proposito, se un sensore viene montato liberamente da personale non specializzato, esso può raccogliere dati non particolarmente significativi al fine di una caratterizzazione della struttura, perciò i dati di tale sensore possono essere eliminati dal processo di estrazione delle caratteristiche.

Infine, va notato che l'acquisizione dei dati, la normalizzazione e la pulitura di un sistema di monitoraggio non devono essere statiche. Le informazioni che saranno acquisite dal processo di estrazione delle caratteristiche e dal processo di sviluppo del modello forniranno informazioni sulle modifiche da apportare per migliorare il processo di acquisizione dei dati.

Il sistema di rilevamento e trasmissione dati: Caso di studio edificio scolastico SCIGLIANO (CS)

Il sistema di rilevamento sensoriale è il primo e uno dei più importanti elementi costituenti l’architettura del monitoraggio strutturale.

Tutti i metodi di monitoraggio strutturale richiedono dei dati della struttura di interesse per eseguire analisi e determinare se si sono verificati danni o se è probabile che si verifichino danni. La maggior parte delle ricerche nel campo dello SHM si è concentrata sull'impiego dei sistemi di sensori per raccogliere le informazioni necessarie per l'analisi.

Le rilevazioni hanno il compito di misurare delle quantità che si traducono principalmente in tre tipologie di parametri:

1. Sorgenti di carico: ambientali (vento, azione sismica) o artificiali (traffico);
2. Risposte strutturali: spostamenti, deformazioni, accelerazioni e inclinazioni;
3. Effetti ambientali: temperatura, precipitazioni atmosferiche, umidità, particelle inquinati presenti nell’aria;

Le variabili ambientali come temperatura e umidità sono comunemente monitorate per isolare la risposta della struttura. Di solito, le quantità di primario interesse sono le risposte strutturali. Pertanto, la nostra analisi si concentrerà sui sensori che misurano solo queste quantità anche se queste dovrebbero essere combinate con i dati ambientali per analizzare correttamente lo stato di una struttura.

L’accuratezza e la precisione delle previsioni formulate attraverso il monitoraggio strutturale sono inevitabilmente correlate all’accuratezza e alla precisione degli strumenti di rilevazione. L’importanza che la qualità del dato fornito dalle misurazioni assume è pari a quella dell’intero modello matematico utilizzato nell’elaborazione dei dati.

I sensori per definizione costituiscono solo una componente dei più complessi trasduttori. Questi sono strumenti che trasformano grandezze fisiche che definiscono la risposta di un sistema come spostamenti, velocità, accelerazioni, tensioni, deformazioni o forme, in segnali elettrici elaborati successivamente dal sistema di acquisizione dati.

I trasduttori possono essere di tipo analogico, ossia il segnale in uscita è una grandezza elettrica che varia in modo continuo, o digitale, dove il segnale in uscita è composto da uno o più segnali digitali che possono assumere solo due livelli di tensione: attivo (non ha bisogno di alimentazione per essere prodotto) o passivo (in tal caso è richiesta una alimentazione).

In generale tutti i sensori devono soddisfare le caratteristiche prestazionali come: sensibilità, risoluzione, portata, linearità, isteresi, accuratezza, precisione, isolamento, basso costo e durabilità.

Negli ultimi anni si sono sviluppati una vastità di nuovi strumenti di misura e sensori, sempre più tecnologici e precisi, nonché più economici. L’aspetto economico non è da trascurare, infatti, nel campo del monitoraggio strutturale abbiamo un grande range di prezzi, che vanno dalle centinaia ad alcune migliaia di euro. Pertanto, si sta cercando di sviluppare strumenti che costino sempre meno, ma allo stesso tempo abbiano sensibilità e precisione elevate.

In particolare, in questo lavoro, saranno presentati dei sensori che allo stato attuale sono quelli più comunemente usati, quindi sono stati ben consolidati e testati.

Oggetto di approfondimento di questo lavoro inoltre sarà il monitoraggio dinamico in continuo di una struttura, sviluppato per ottenere l’individuazione del danno ed incentrato su algoritmi in grado di elaborare i cambiamenti delle forme modali o dei parametri del sistema strutturale (come la frequenza naturale, i modi di vibrare e il rapporto di smorzamento). A tal fine, una tipica architettura dei sistemi di monitoraggio dinamico è basata sull'impiego di sensori periferici, in particolare accelerometri, direttamente connessi tramite cavi ad un sistema di acquisizione centralizzato.

La scelta dei sensori più appropriati parte quindi dalla loro precisione che, però da sola, non è sufficiente a garantire l’efficienza del monitoraggio, si deve garantire la conservazione nel tempo dello strumento in ottime condizioni, in particolare per gli strumenti di tipo elettronico, dove il funzionamento è influenzato in maniera considerevole dalle condizioni ambientali.

Tra i numerosi tipi di sensori che si possono utilizzare per il monitoraggio, per i nostri fini, abbiamo scelto di utilizzare gli accelerometri. Questi dispositivi misurano l'accelerazione indotta da vibrazioni naturali oppure da forzanti in una particolare posizione di una struttura. Nel corso dell’ultimo

decennio questi sono stati i dispositivi di misurazione più utilizzati in quanto i dati da essi prodotti possono fornire preziose informazioni sulle caratteristiche dinamiche di una struttura. Attraverso le procedure di post-elaborazione, le misure di accelerazione possono essere utilizzate per calcolare la frequenza, lo smorzamento e le forme modali di una struttura utili nella valutazione globale della salute strutturale.

Sebbene gli accelerometri siano stati a lungo impiegati per le tecniche di monitoraggio, sussistono ancora problemi con la produzione di errori durante l'integrazione numerica dei dati. Questi dispositivi generano anche una grande quantità di dati che possono richiedere un'elaborazione intensiva. Un altro inconveniente dell'uso di tali strumenti è l'elevato costo dovuto non solo all'accelerometro vero e proprio (qualche migliaio di euro), ma al sistema di acquisizione dei dati (amplificatori, trasformatori analogico-digitali, computer per l'analisi dei dati e software dedicati).

Nonostante queste limitazioni, gli accelerometri rappresentano ad oggi la migliore opzione per misurare le proprietà dinamiche di una struttura.

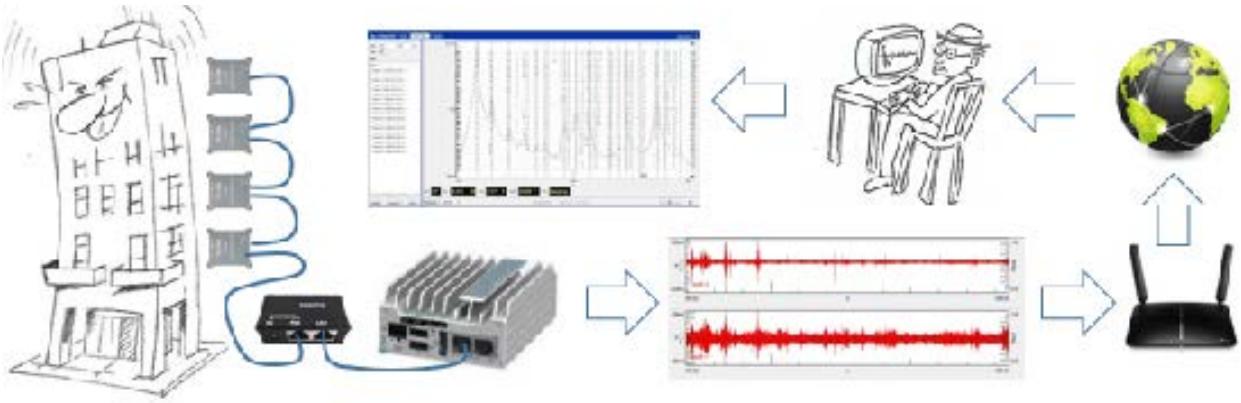
Sul mercato troviamo una moltitudine di sensori sia di tipo wired che wireless. La caratteristica dominante di questi ultimi sistemi elettronici emergenti, che viene presa come pretesto e come elemento distintivo ed indicativo di migliori prestazioni, è quasi sempre la trasmissione wireless del segnale, che se va bene per misure statiche, con una quantità di dati decisamente trascurabile, non è per niente adatta a quelle dinamiche. Anche con frequenze di campionamento non proibitive i consumi sono talmente elevati da rendere inevitabile l'alimentazione da rete e, a questo punto, lo stesso cavo di alimentazione può essere utilizzato per trasportare il segnale con conseguente maggiore sicurezza nella trasmissione e nella sincronizzazione.

La scelta è quindi ricaduta sui prodotti MonoDAQ-E-gMeter di Dewesoft che al momento hanno dimostrato di essere la soluzione ottimale sul mercato tra tutte quelle vagliate. L'azienda ha sviluppato un modulo di acquisizione accelerometrico triassiale, con un trasduttore MEMS incorporato (MEMS - Micro-ElectroMechanical System), ovvero un condizionatore di segnale integrato con un accelerometro. Quindi realizza un unico dispositivo di acquisizione dati a canale singolo EtherCAT. L'obiettivo del dispositivo integrato è quello di ottimizzare il risultato in termini di rapporto segnale/rumore rispetto alle soluzioni presenti sul mercato.

Questi trasduttori presentano una densità di rumore spettrale di soli 25 µg/√Hz e 100 µg di rumore residuo ad una larghezza di banda 50 Hz. L'accelerometro utilizzato al suo interno risulta ad oggi la soluzione più conveniente per le misure sismiche, in quanto viene utilizzata con una doppia funzione. Anche se utilizzato principalmente per la misura dell'accelerazione, lo stesso può essere utilizzato per misurare con precisione lo spostamento, integrando l'accelerazione due volte e applicando un filtraggio appropriato. Questo compito viene eseguito dall'ambiente software di dotazione gratuito DewesoftX.
Gli accelerometri MEMS IOLITEd-3xMEMS-ACC sono stati utilizzati per misurare accelerazioni e inclinazioni, mentre il software di acquisizione dati Dewesoft X viene utilizzato per:

- gestione dell'acquisizione
- impostazione dei parametri per tutti i sensori
- visualizzazione della cronologia temporale dei segnali acquisiti
- FFT istantanea
- salvataggio dei dati.

Per quanto riguarda i rilievi, la grandezza dinamica scelta per le misure è costituita dalla risposta in frequenza della struttura in termini di accelerazioni nei punti definiti.
In particolare, per un edificio residenziale o monumentale, un tipico sistema di monitoraggio dinamico modulare è costituito da una serie di 3 moduli MEMS-ACC DAQ opportunamente distribuiti sui vari piani e alimentati tramite Ethernet (PoE).
Nel caso di edifici regolari in pianta e in altezza sono richieste installazioni minime con 3xMEMS-ACC alle fondazioni (zero sismico), oltre che una coppia disposta, ad esempio, all'ultimo piano. Questa è l'ipotesi di solai a taglio e quando si vogliono analizzare solo le prime forme modali. Con l'aumentare dell'interesse per le forme modali più elevate, si verifica un progressivo coinvolgimento di tutti gli altri piani, fino a quando non sono tutti interessati.
A fronte di edifici più complessi, il numero dei moduli tende inevitabilmente ad aumentare in funzione della volontà di ottenere analisi sempre più veritiere. I dati acquisiti vengono poi memorizzati secondo opportuni algoritmi selettivi sul supporto di massa del PC industriale posto in un apposito pannello insieme ad un router. Il router viene utilizzato per fornire una connessione Internet sul terminale remoto. Normalmente è attivo anche un sistema per l'invio di messaggi al superamento delle soglie precedentemente impostate.



L’architettura di rete realizzata, secondo le specifiche del committente, ha richiesto l’utilizzo della rete TIM 4G e l’archiviazione dei dati su cloud Noovle.
I componenti della rete sono descritti nell’immagine sottostante:

Costituita da:

1. rete di accelerometri MEMS IOLITEd-3xMEMS-ACC
2. rete di alimentazione costituita da passive PoE injector e power supply
3. accentratore dati NUC
4. switch e router 4G per la trasmissione su cluod Noovle

Fase di estrazione delle caratteristiche e definizione del modello

La fase del processo di STRUCTURAL HEALTH MONITORING che ha ricevuto maggiore interesse nella letteratura tecnica è l'estrazione delle caratteristiche, in quanto consente di distinguere tra strutture danneggiate e strutture non danneggiate. Uno dei metodi più comuni di estrazione delle caratteristiche si basa sulla correlazione delle quantità misurate dal sistema, come l'ampiezza o la frequenza delle vibrazioni, con una prima osservazione dello stato di degrado.
La comprensione delle caratteristiche appropriate può essere acquisita da diversi tipi di studi sia analitici che sperimentali e generalmente i risultati sono combinati per ottenere informazioni ancor più attendibili. Quando sono previsti confronti di molti set di caratteristiche, ottenuti nel corso della vita della struttura, è necessaria e vantaggiosa una condensazione dei dati. Inoltre, poiché i dati verranno acquisiti da una struttura per un lungo periodo di tempo e in un ambiente operativo, devono essere sviluppate solide tecniche di condensazione dei dati per mantenere la sensibilità delle caratteristiche ai cambiamenti strutturali di interesse in presenza di variabilità ambientale e operativa.
Al contrario di ciò che è avvenuto per lo studio della fase di estrazione, lo sviluppo di modelli previsionali ha ricevuto una minima attenzione nella letteratura tecnica.
Terminata la fase di estrazione delle caratteristiche, dunque, si procede con lo sviluppo dei modelli previsionali che prevedono l'implementazione di algoritmi che operano sulle caratteristiche estratte per quantificare lo stato di danneggiamento della struttura. Gli algoritmi utilizzati nello sviluppo di modelli previsionali di solito rientrano in tre categorie:

- "Group classification"
- "Regression analysis"
- "Outlier analysis"

I primi due sono algoritmi che possono essere ricondotti alla famiglia dei cosiddetti "supervised learning", ovvero di quelli che possono essere impiegati quando sono disponibili dati sia per la struttura non danneggiata che per quella danneggiata.
L'ultimo invece fa parte della categoria dei cosiddetti "unsupervised learning", cioè di quegli algoritmi utilizzabili quando non sono a disposizione dati riferiti allo stato danneggiato dell'opera.
Lo stato di danno di un sistema può essere descritto in cinque passi attraverso la risposta a cinque quesiti, così come proposto da Rytter (1993):

I. Esistenza: è presente il danno nella struttura?
II. Localizzazione: dove si trova il danno nella struttura?
III. Tipologia: di quale tipologia di danno si tratta?
IV. Estensione: quanto è grave il danno presente nella struttura?
V. Prognosi: quanta vita utile rimane alla struttura?

Rispondere a queste domande nell'ordine proposto significa accrescere la conoscenza sullo stato di danno presente. A tal proposito, vengono in aiuto i modelli previsionali presentati precedentemente che consentono di dare una risposta chiara e quantificabile. Quando si applicano algoritmi unsupervised learning, i modelli prodotti vengono generalmente utilizzati per rispondere a domande sull'esistenza e sulla posizione del danno. Una risposta sul tipo, estensione e sulla prognosi che il danno comporta, richiede l'utilizzo di algoritmi supervised learning accoppiati all'uso di modelli analitici.
A tale proposito i metodi proposti in letteratura si dividono in:

- Level I: individuazione del danno;
- Level II: individuazione e localizzazione del danno;
- Level III: individuazione, localizzazione e quantificazione severità del danno;
- Level IV: individuazione, localizzazione, quantificazione severità e conseguenze del danno;

Questi livelli di identificazione sono gerarchici, ovvero riuscire a identificare un livello comprende la conoscenza di tutti i precedenti. A rigore, mentre le prime tre famiglie di metodi corrispondono effettivamente a livelli diversi di approfondimento della diagnosi, il quarto riguarda più strettamente la prognosi ossia la previsione del comportamento futuro della struttura. Si tratta comunque di un livello di conoscenza più approfondito dei precedenti nel senso che la prognosi viene effettuata a valle di una diagnosi sullo stato attuale della struttura e in questo senso è da intendersi la collocazione di tali metodi come ulteriore affinamento delle procedure di diagnosi.
Con l'aumentare dei livelli si ha un aumento della raffinatezza dei metodi, a ciascuno dei quattro livelli di approfondimento corrispondono approcci diversi al problema così come requisiti diversi della rete di sensori da collocare sulla struttura monitorata. Anche gli algoritmi di identificazione del danno varieranno a seconda del tipo di monitoraggio desiderato. Ad esempio, se sono necessarie informazioni sulla fase uno, allora le misurazioni delle vibrazioni globali potranno essere sufficienti per accertare l'esistenza di danni da qualche parte nella struttura. L'identificazione della posizione del

danno richiederà una rete di sensori notevolmente più ricca e sensori che forniscono informazioni locali più solide. Al fine di determinare il grado di danno oltre alla selezione del sensore, sono necessari algoritmi di danno efficienti e robusti. La previsione della durata residua si basa in genere sulla misurazione della fatica e della frattura e può richiedere sensori e strumenti matematici diversi che conducono alla stima della durata residua o alla valutazione della conformità con le specifiche di progetto. La sfida è nello sviluppo di sistemi in grado di rispondere a tutte e quattro le fasi del monitoraggio dei danni.

Determinazione della posizione geometrica del danno e quantificazione della gravità

La fase a valle del processo di estrazione delle caratteristiche è dunque lo sviluppo di modelli previsionali che operano sulle caratteristiche estratte per quantificare lo stato di danneggiamento della struttura. In questo lavoro ci occuperemo della individuazione, localizzazione e quantificazione della severità del danno, ovvero cercheremo di dare una risposta ai primi tre quesiti proposti da Rytter. Gli algoritmi che utilizzeremo nello sviluppo e nell'analisi dei modelli previsionali rientrano pertanto nelle due categorie: "Group classification" e "Regression analysis".

Ovvero algoritmi che possono essere ricondotti alla famiglia dei cosiddetti "supervised learning", ovvero di quelli che possono essere impiegati quando sono disponibili dati sia per la struttura non danneggiata che per quella danneggiata.

In particolar modo ci siamo concentrati su modelli previsionali basati sulla Regression analysis, in quanto quest'ultimi richiedono, per la fase di addestramento, una mole di dati notevolmente inferiore ai modelli che fanno uso della categoria Group classification. Il principale problema che abbiamo incontrato nella fase di sviluppo del sistema è stata la mancanza di dati per addestrare un modello. Ogni struttura che si và a monitorare ha naturalmente, forma, dimensione e caratteristiche diverse, ma nell'insieme anche i sensori e l'architettura utilizzata variano da progetto a progetto, per cui non sono disponibili dataset online su cui poter fare sperimentazione. L'approccio convenzionalmente utilizzato è quello di produrre un dataset attraverso simulazioni su un modello ad elementi finiti, simulando varie condizioni di danno della struttura. Questo tipo di approccio, produce una serie di dati disomogenei e classi molto sbilanciate per cui è difficile produrre un clustering sui dati. Generalmente per poter usare una group classification analysis, i dati prodotti dal modello ad elementi finiti vengono arricchiti, in fase di simulazione, con segnali di disturbo, in modo da riprodurre le condizioni operative dei sensori. In questo modo si ottiene un dataset più strutturato che ci consente

di fare questo tipo di analisi. I modelli di tipo Regression analysis quindi funzionano meglio in caso di dataset poco popolato.
Lo sviluppo dei modelli è stato condotto con l’uso della libreria Scikit-learn o anche nota come sklearn. Una libreria software di machine-learning gratuita per il linguaggio di programmazione Python. Presenta vari algoritmi di classificazione , regressione e clustering tra cui support vector machines, random forests, gradient boosting, k-means e DBSCAN.

ANN-based damage detection using the modal properties

Il primo studio di applicabilità di questa metodologia risale al 2005 (Lee et al.), anno in cui viene presentato per la prima volta un metodo di rilevamento dei danni basato su reti neurali che fa uso delle proprietà modali della struttura. L’intuizione di partenza era che, rispetto agli altri metodi, l’analisi modale risulta meno sensibile agli errori di modellazione, pertanto un metodo che utilizzi i parametri modali, ad esempio come input per le reti neurali, può produrre dei vantaggi significativi nell’analisi dinamica delle strutture. L'efficacia dell’approccio è stato testato su due esempi numerici, a trave semplice e ad un ponte a più travi. Lo stesso metodo è stato poi applicato ad un modello di un ponte riprodotto in laboratorio e sul ponte Hannam Grand Bridge a Seoul, in Corea. Il metodo ha identificato le posizioni dei danni con una buona accuratezza per tutti i casi di danno, ma la gravità dei danni stimata era minore rispetto ai valori reali con presenza di errori falsi positivi in diverse posizioni.
Nello stesso anno Yeung & Smith hanno presentato una procedura di rilevamento dei danni, utilizzando il pattern recognition per il riconoscimento dell’impronta digitale o firma della struttura, producendo degli ottimi risultati per la risoluzione del problema di livello 1: riconoscimento del danno.
L’anno successivo, nel 2006, Yuen & Lam hanno intuito che era necessario utilizzate per le due fasi di rilevamento dei danni due tipi di reti neurali. Nella prima fase, l'identificazione del danno è stata prodotta utilizzando come input di una rete ANN le firme digitali della struttura con e senza danno. Nella seconda fase, la gravità del danno viene stimata utilizzando un'altra ANN con parametri modali come input. La metodologia di rilevamento dei danni è stata dimostrata utilizzando un edificio di cinque piani. Il danno è stato definito come la riduzione della rigidità di interpiano. L’addestrato delle reti neurali sono state condotte con singolo danno, doppi casi di danno con la stessa estensione del danno e casi di doppio danno con diverse estensioni di danno. L'addestrato ANN ha individuato con successo la posizione del danno in tutti i casi con un singolo danno. Tuttavia, nel caso di doppio

danno, solo quando le due estensioni di danno erano significativamente diverse, la rete ha prodotto la posizione con l'entità del danno maggiore.
Gli studi che si sono poi susseguiti nel corso degli anni, hanno evidenziato come le impronte digitali derivate dall'analisi delle frequenze naturali e dalle forme modali della struttura sono i metodi più popolari per identificare il danno in quanto queste due quantità sono facili da individuare, con un livello di fiducia relativamente alto e un costo relativamente basso. Inoltre è stato evidenziato come i metodi basati sulla frequenza naturale ne limitano l'applicabilità a causa di molte ragioni. Un danno che produce una bassa frequenza richiede misurazioni molto precise, questi piccoli cambiamenti potrebbero non essere rilevati a causa di errori di misurazione. Inoltre, questi metodi non sono in grado di distinguere i danni in posizioni simmetriche in una struttura simmetrica, mentre i cambiamenti nelle forme modali sono molto più sensibili ai danni locali.

Preparazione del dataset per l'addestramento

Un metodo di estrazione delle caratteristiche per l'identificazione del danno è quello di applicare difetti, simili a quelli che ci si aspetta nella struttura nelle effettive condizioni operative, a sistemi fittizi e sviluppare un'iniziale comprensione dei parametri sensibili al danno atteso. Il sistema difettoso può anche essere usato per verificare che le misurazioni diagnostiche siano sufficientemente sensibili da distinguere il sistema danneggiato dal sistema privo di danno. L'uso di strumenti analitici sperimentalmente convalidati, come i modelli ad elementi finiti, può essere una grande risorsa in questo ambito, in quanto vengono utilizzati per eseguire sperimentazioni numeriche, in cui il danno viene introdotto nel sistema attraverso la simulazione al computer. I test sull'accumulo di danno, durante i quali componenti strutturali significativi del sistema in esame vengono degradati sottoponendoli a condizioni di carico reali, possono anche essere utilizzati per identificare specifiche caratteristiche.
A tal fine sono state condotte diverse simulazioni considerando vari gradi di danneggiamento che hanno coinvolto uno o più pilastri contemporaneamente.
I dati prodotti da una singola simulazione sono descritti nella tabella sottostante:

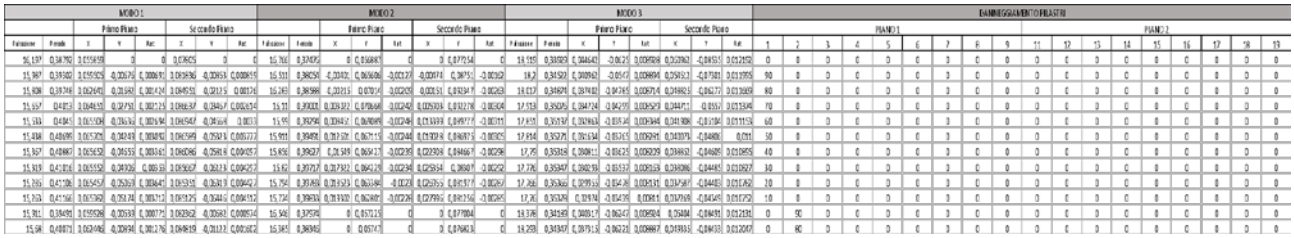
PULSAZIONI E MODI DI VIBRAZIONE														
	Modo	Pulsazion	Periodo	Smorz	Sd/g	Sd/g	Sd/g	Sd/g	Sd/g	Sd/g	Piano	X	Y	Rot
	N.ro	(rad/sec)	(sec)	Mod(%)	SLO	SLD	SLV X	SLV Y	SLC X	SLC Y	N.ro	(m)	(m)	(rad)
	1	15,365	0,40894	5		0,254	0,192	0,192			1	0,065568	-0,04519	0,003264
											2	0,086102	-0,05651	0,003979
	2	15,847	0,39649	5		0,254	0,192	0,192			1	0,014578	0,065861	-0,00239
											2	0,021685	0,085252	-0,00299
	3	17,784	0,3533	5		0,254	0,192	0,192			1	0,031135	-0,03686	0,008236
											2	0,039308	-0,0469	0,010932

La tabella riassume i dati modali prodotti da una simulazione con un software di calcolo ad elementi finiti.

Vengono presi in considerazione soltanto i primi tre modi di vibrare dell'intera struttura. In questo caso stiamo monitorando una struttura simmetrica a due piani, pertanto questi tre dati sono ritenuti sufficienti, o comunque contengono sufficiente informazione per poter discriminare una condizione di danneggiamento.

Le informazioni che vengono acquisite per ogni modo di vibrare sono: la pulsazione ed il periodo di ogni modo di vibrare, la percentuale di smorzamento (anch'esso parametro strutturale che varia con il degrado della struttura) ed infine le deformate modali ricavate per ogni singolo piano (informazioni indispensabili per discriminare la locazione geometrica del danno).

Tutte queste informazioni sono state raggruppate e ripulite dai campi non ritenuti utili per essere inserite nel dataset di training.



Le label utilizzate descrivono il tipo di pilastro danneggiato e della percentuale di danno prodotto:

DANNEGGIAMENTO PILASTRI																		
PIANO 1									PIANO 2									
1	2	3	4	5	6	7	8	9	11	12	13	14	15	16	17	18	19	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
70	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	70	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Regression analysis

In questa capitolo verranno descritte tutte le fasi richieste da una analisi di regressione. In particolar modo descriveremo l’uso della libreria sklearn ed andremo a valutare i risultati prodotti da alcuni modelli di regressione applicati al nostro dataset.
Sklearn è un modulo del linguaggio python con funzioni per il machine learning e l'apprendimento automatico. La libreria contiene diversi algoritmi di apprendimento come classificatori o regressori (decision tree, regressione lineare, percetron, ecc.) e diversi dataset didattici di addestramento.
Il successo di questa libreria è dovuto alla sua facilità di utilizzo.
Un algoritmo di ML è suddiviso in tre fasi:

- preparazione dei dati
- creazione del modello previsionale (addestramento)
- verifica del modello

Dopo aver addestrato un modello, questo può essere salvato in un file, per poter essere successivamente utilizzato per eseguire una predizione oppure per essere addestrato nuovamente.
A questo fine esiste una funzione in libreria che serializza il modello e lo salva in un file fisico sul PC, rendendolo persistente.

La preparazione dei dati

Il modulo Sklearn ha il vantaggio di incorporare diversi dataset di default. Pertanto, per chi sperimenta la prima volta la libreria non occorre che carichi il dataset dall'esterno.
Vediamo ad esempio i passi da seguire per il clustering del noto dataser Iris. Per importare il dataset in memoria occorre utilizzare la funzionalità datasets di sklearn e quindi assegnare alla variabile iris il dataset con il metodo load_iris(). A questo punto nella variabile iris c'è tutto il dataset iris.

```
from sklearn import datasets
iris = datasets.load_iris()
```

Gli esempi del training set si trovano con il metodo *data*. Ogni riga contiene un esempio e ogni esempio è composto da 4 attributi che misurano la lunghezza e la larghezza del petalo e del sepalo in centimetri.

Le risposte corrette degli esempi (etichette), invece, sono memorizzate sotto il metodo *target*. Il vettore è composto da 130 etichette zero, uno e due:

0 = Iris-Setosa
1 = Iris-Versicolor
2 = Iris-Virginica

Per semplicità nell'esempio si assegnano gli esempi (contenuti in iris.data) alla variabile X prendendo soltanto due misure, la colonna 2 e 3, mentre le colonne 0 e 1 (le prime due) sono invece eliminate.

```
X = iris.data[:, [2, 3]]
y = iris.target
```

Ora occorre dividere il dataset in training set (insieme di addestramento) e test set (insieme di test). Per farlo si utilizza la funzione *train_test_split*

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

La funzione train_test_split ha quattro parametri in input

- Data. Il dataset degli esempi (X)
- Target. Il dataset delle etichette (y)
- Testize. Indica la percentuale di esempi da usare per l'insieme di test. In questo caso è 0.3 ossia il 30%.
- Random_state. E' l'indicatore di randomizzazione che per il momento è posto a zero.
- Restituisce in output quattro insieme di dati:
- L'insieme di addestramento (X_train)
- L'insieme di test (X_test)
- Le etichette dell'insieme di addestramento (y_train)
- Le etichette dell'insieme di test (y_test)

La creazione del modello previsionale

Per creare un modello previsionale si deve scegliere un algoritmo di apprendimento. Nell'esempio seguente viene utilizzato perceptron, uno dei primi algoritmi di classificazione della storia. Successivamente si settano gli iperparametri del processo di learning (max_iter , tol, eta0). Dove max_iter indica il numero di cicli (epoche) da elaborare mentre il parametro eta0 è il tasso di apprendimento del perceptron.

ppn = Perceptron(max_iter=40, tol=0.001, eta0=0.01, random_state=0)

L'attributo tol indica la tolleranza sufficiente per l'uscita dal ciclo.
A questo punto si può eseguire il process learning tramite il metodo fit per cominciare l'addestramento.

ppn.fit(X_train, y_train)

La verifica del modello previsionale

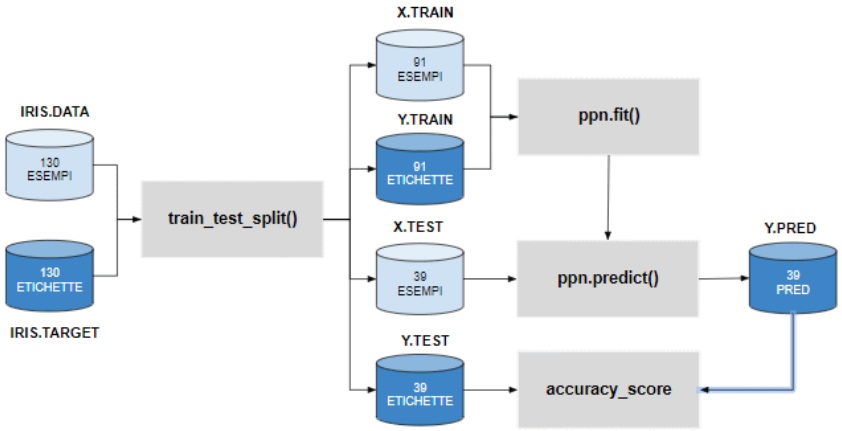
Per verificare il modello previsionale, si applicano gli esempi dell'insieme di test (X.test). Nella libreria scikit-learn c'è un'apposita metodo per verificare il modello. Si tratta del metodo predict().

y_pred = ppn.predict(X_test)

Le risposte del modello sono registrate nella variabile y_pred. A questo punto si va a controllare se le risposte del modello (y_pred) coincidono con le etichette corrette (Y.test). Per fare questo si utilizza la funzione accuracy_score di scikit-learn.

accuracy_score(y_test, y_pred)

Di seguito una rappresentazione schematica delle fasi di elaborazione:



Come creare un dataset personalizzato

Per creare un dataset personalizzato su Python, i dati di addestramento devono essere precedentemente organizzati su un foglio elettronico qualsiasi (es. Excel, Calc, ecc.) e successivamente salvati in formato CSV.

Ad esempio, si crea un dataset banale con cinque esempi e tre attributi (f1,f2,f3).

	A	B	C	D
1	label	f1	f2	f3
2	1	7	1	3
3	0	1	2	0
4	1	8	3	6
5	1	6	1	5
6	0	2	2	0
7				
8				
9				

Se l'apprendimento si basa sul machine learning supervisionato, una colonna del foglio elettronico deve essere destinata alle etichette delle risposte corrette ("label") per ciascun esempio mentre le altre agli attributi (features).

A questo punto può essere caricato il dataset con uno script in linguaggio Python. Su Python si usa la funzione read_csv() del modulo Pandas per importare il file CSV e salvarlo in una variabile qualsiasi (var).

```
import pandas as pd
var = pd.read_csv('temp.csv')
```

Tuttavia, in questa forma è ancora inutilizzabile da un algoritmo di machine learning, bisogna estrarre la colonna "label" del dataset e salvarla in un'altra variabile di Python (y_train).

```
y_train= df['label']
```

Successivamente occorre estrarre le altre colonne del dataset, senza la colonna "label", e salvarla in un'altra variabile (x_train).

```
x_train = df.drop('label', axis=1)
```

A questo punto è possibile eseguire l'addestramento del modello:

```
from sklearn.linear_model import Perceptron
ppn = Perceptron(max_iter=40, tol=0.001, eta0=0.10, random_state=0)
ppn.fit(x_train, y_train)
ppn.predict([[7, 4, 2]]
```

Regression analysis con sklearn

L'analisi di regressione è stata condotta eseguendo alcuni test su diversi metodi proposti dalla libreria sklearn. Libreria che include diversi moduli per la classificazione, la regressione e l'anomaly detection.
Gli algoritmi di regressione che troviamo in questa libreria sono:

- AdaBoost regressor
- Bagging regressor
- extra-trees regressor
- Gradient Boosting for regression
- random forest regressor
- Stack of estimators with a final regressor
- Prediction voting regressor for unfitted estimators
- Histogram-based Gradient Boosting Regression Tree

Contenuti all'interno di "ensemble". Mentre all'interno di "linear_model" troviamo un insieme di metodi destinati alla regressione lineare, cioè è applicabile a quei problemi in cui ci si aspetta che il valore target sia una combinazione lineare delle caratteristiche. In notazione matematica è la seguente:

$$\hat{y}(w, x) = w_0 + w_1 x_1 + \ldots + w_p x_p$$

I modelli forniti da questa libreria sono i seguenti:

- LinearRegression
- Ridge / RidgeCV
- Lasso / LassoCV/ LassoLarsCV / LassoLarsIC / MultiTaskLasso
- ElasticNet / MultiTaskElasticNet / MultiTaskElasticNetCV
- Least-angle regression (LARS)
- LassoLars
- OrthogonalMatchingPursuit
- Bayesian regression / BayesianRidge / ARDRegression /
- Logistic regression/ LogisticRegressionCV
- TweedieRegressor
- SGDRegressor
- PassiveAggressiveRegressor
- HuberRegressor
- TheilSenRegressor
- RANSACRegressor

Mentre all'interno della libreria *sklearn.neighbors* troviamo i seguenti algoritmi:

- KNeighborsRegressor
- RadiusNeighborsRegressor

Ed LinearSVR o SGDRegressor della libreria *sklearn.svm*

Analisi dei residui del regressore

Dopo aver creato un modello di regressione con il machine learning supervisionato, valutiamo la qualità previsionale del modello tramite le metriche che analizzano i residui. Dove per residuo si intende la differenza tra il valore previsto dal modello ed il valore corretto presente nel dataset di test.

Scikit learn mette a disposizione diverse metriche utili:

- mean_absolute_error(y_test,p)
- mean_squared_error(y_test,p)
- r2_score(y_test,p)

Queste metriche restituiscono un valore sintetico sulla qualità previsionale.

Applicando i vari algoritmi al nostro dataset abbiamo stilato la seguente tabella riassuntiva con il metodo applicato alla prima colonna e le metriche nelle ultime tre colonne.

Uno primo studio del sistema è stato condotto una sola label per la predizione dello stato di danneggiamento della struttura e la posizione del danno, ovvero il pilastro danneggiato.

Mod3	Mod3	Mod3	Mod2	Mod3	Mod3	Mod3	Dam
0.33945	0.04462	-0.062468	0.008924	0.060932	-0.085304	0.012186	0
0.3533	0.031135	-0.036856	0.008236	0.039308	-0.046904	0.010932	1
0.35013	0.041357	-0.035837	0.008271	0.055514	-0.046403	0.011103	2
0.3533	0.051228	-0.036856	0.008236	0.070008	-0.046904	0.010932	3
0.34583	0.032831	-0.061491	0.008784	0.042408	-0.083035	0.011862	4
0.33985	0.044612	-0.062457	0.008922	0.060886	-0.08524	0.012177	5
0.34583	0.055013	-0.061491	0.008784	0.076214	-0.083035	0.011862	6
0.3533	-0.031135	0.078452	-0.008236	-0.039308	0.106139	-0.010932	7
0.35013	0.041357	-0.079964	0.008271	0.055514	-0.109035	0.011103	8
0.3533	-0.051228	0.078452	-0.008236	-0.070008	0.106139	-0.010932	9
0.34166	0.041923	-0.057596	0.008744	0.060552	-0.083616	0.012355	10
0.34426	0.04294	-0.049723	0.008588	0.061213	-0.075217	0.012243	11
0.34166	0.045522	-0.057596	0.008744	0.062993	-0.083616	0.012355	12
0.34171	0.038761	-0.061514	0.008788	0.056514	-0.085699	0.012243	13
0.33983	0.044559	-0.062383	0.008912	0.060883	-0.085236	0.012177	14
0.34171	0.049115	-0.061514	0.008788	0.065914	-0.085699	0.012243	15
0.34166	-0.041923	0.064827	-0.008744	-0.060552	0.089347	-0.012355	16
0.34426	0.04294	-0.070509	0.008588	0.061213	-0.09618	0.012243	17
0.34166	-0.045522	0.064827	-0.008744	-0.062993	0.089347	-0.012355	18

Con questo dataset sono state condotti diversi test su algoritmi di regressione presenti nella libreria adoperata. I risultati prodotti da questa sperimentazione sono stati raccolti nella tabella sottostante.

Algoritmo	mean_absolute_error	mean_squared_error	r2_score
LinearRegression	2.96e-12	1.13e-23	1
LogisticRegression	4.94	65.68	-1.189
GradientBoostingRegressor	0.00129	2.148	0.99
RandomForestRegressor	1.23	3.295	0.89
ElasticNet	4.18	25.37	0.154
HuberRegressor	2.313	15.02	0.49
PassiveAggressiveRegressor	5.07	36.65	-0.22
KNeighborsRegressor	1.5789	3.868	0.871
SVR	3.610	20.80	0.306
LinearSVR	2.20	15.899	0.47
AdaBoostRegressor	0.296	0.227	0.99
BaggingRegressor	4.74	30.05	-0.001
StackingRegressor			
VotingRegressor	0.697	1.24	0.958
ExtraTreesRegressor	0	0	1

HistGradientBoostingRegressor	4.736	30	0
Ridge	3.11	17.43	0.41
RANSACRegressor			

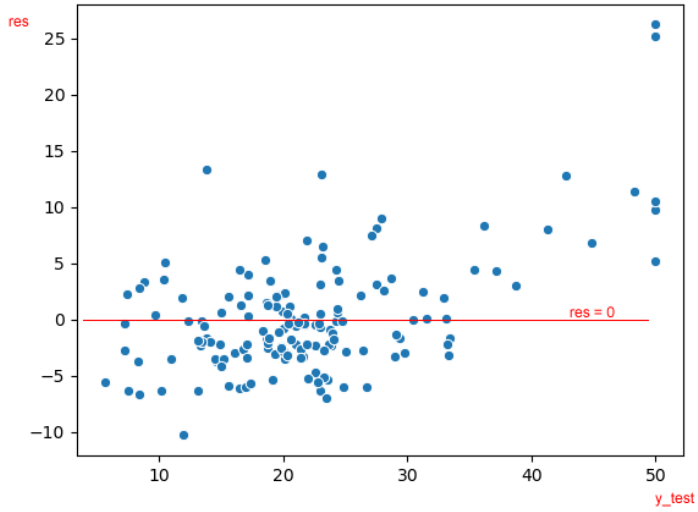
Se si vogliono affinare i risultati e comprendere le cause che hanno portato ad un errore sui residui allora è possibile svolgere ulteriori indagini.

Un primo passo è cercare di capire in quali caratteristiche (feature) l'errore è più alto.

Un primo modo per approfondire la distribuzione degli errori è l'analisi grafica degli errori. Il metodo più immediato per procedere è il calcolo della differenza tra il valore previsto e il valore target per ogni esempi, ossia il residuo e la sua visualizzazione in un diagramma cartesiano ponendo il residuo (res) sull'asse delle ordinate e la variabile target (y_test) sull'asse delle ascisse.

```
import numpy as np

import seaborn as snp

import matplotlib.pyplot as plt

snp.scatterplot(x=y_test, y=res)

plt.show()
```

Il risultato sarà il seguente:



Ogni punto nel diagramma è un esempio. I punti vicini all'ordinata res=0 hanno un errore minimo o nullo. Quanto più un punto è lontano da res=0, tanto maggiore è l'errore.

Dopo questa prima analisi si può approfondire la relazione tra il residuo e le varie features (caratteristiche) del dataset. In questa analisi si costruisce lo stesso grafico precedente, sostituendo la variabile target (y_test) con una feature (caratteristica) per individuare in quali features il modello statistico produce più errori e per quali valori. Questa indagine ci consente di scoprire le correlazioni tra le features e gli errori. Una volta raccolti tutti i dati, si va a modificare il dataset ed eventualmente l'algoritmo di addestramento al fine di ridurre l'errore. Infine, si ripete l'addestramento per produrre un nuovo modello.

Multioutput Regression

La regressione si riferisce generalmente a un problema di modellazione predittiva che implica la previsione di un valore numerico. Ad esempio, la previsione di dimensioni, peso, quantità, numero di vendite o numero di clic su una pagina sono problemi di regressione. In genere, per questo tipo di problemi viene previsto un singolo valore numerico di risposta, date le variabili di input. Alcuni problemi di regressione richiedono invece la previsione di due o più valori numerici. Come ad esempio, la previsione di una coordinata x e y. Questi problemi sono indicati come regressione a più output o semplicemente multioutput. Nella regressione multioutput, in genere gli output dipendono sia dall'input che l'uno dall'altro. Ciò significa che spesso gli output non sono indipendenti l'uno dall'altro e può essere richiesto un modello che preveda come uscita tutti gli output contemporaneamente oppure una sequenza di output che mantiene la dipendenza tra i valori. La previsione di serie temporali a più fasi può essere considerata una regressione di questo tipo, con più output in cui è prevista una sequenza di valori futuri e ogni valore previsto dipende dai valori precedenti nella sequenza.

Alcuni algoritmi di apprendimento automatico di regressione supportano già la regressione multioutput, come ad esempio:

- LinearRegression (e correlati)
- KNeighborsRegressor

- DecisionTreeRegressor
- RandomForestRegressor (e correlati)

A questo punto si pone il problema di come poter valutare la qualità di una previsione o di un modello per una regressione multioutput. La metrica utilizzata in questi casi prende il nome di k-fold cross-validation. Un esempio di applicazione con un modello DecisionTreeRegressor su un problema di test, che utilizza una convalida incrociata di 10 volte con tre ripetizioni è il seguente:

```
# evaluate multioutput regression model with k-fold cross-validation
from numpy import absolute
from numpy import mean
from numpy import std
from sklearn.datasets import make_regression
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
# create datasets
X, y = make_regression(n_samples=1000, n_features=10, n_informative=5, n_targets=2, r
# define model
model = DecisionTreeRegressor()
# define the evaluation procedure
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
# evaluate the model and collect the scores
n_scores = cross_val_score(model, X, y, scoring='neg_mean_absolute_error', cv=cv, n_j
# force the scores to be positive
n_scores = absolute(n_scores)
# summarize performance
print('MAE: %.3f (%.3f)' % (mean(n_scores), std(n_scores)))
```

L'esecuzione dell'esempio valuta le prestazioni del modello dell'albero decisionale per la regressione multioutput sul problema di test. La media e la deviazione standard del MAE (errore medio assoluto) vengono riportate e calcolate su tutte le ripetizioni.

Se invece il nostro problema richiede un algoritmo di regressione diverso da quelli già elencati, una soluzione consiste nel dividere il problema di regressione multioutput in più sottoproblemi. Il modo più ovvio per farlo è suddividere un problema di regressione a più uscite in più problemi di regressione a singola uscita. Ad esempio, se un problema di regressione multioutput richiedeva la previsione di tre valori y1, y2 e y3 dato un input X , allora questo potrebbe essere suddiviso in tre problemi di regressione a output singolo:

Problema 1 : dato X , prevedere y1

Problema 2 : dato X , prevedere y2

Problema 3 : dato X , prevedere y3

Esistono due approcci principali per implementare questa tecnica. Il primo approccio prevede lo sviluppo di un modello di regressione separato per ogni valore di output da prevedere. Possiamo pensare a questo come a un approccio diretto, poiché ogni valore target è modellato direttamente. Il secondo approccio è un'estensione del primo metodo tranne per il fatto che i modelli sono organizzati in una catena. La previsione dal primo modello viene presa come parte dell'input del secondo modello e il processo di dipendenza da output a input si ripete lungo la catena dei modelli. In questo caso possiamo dire che il primo metodo sviluppa un modello con uscite indipendenti, mentre il secondo

sviluppa una sequenza di modelli dipendenti che mantengono inalterate le dipendenze tra i vari output.

L'approccio diretto alla regressione multioutput implica la divisione del problema di regressione in un problema separato per ciascuna variabile target da prevedere. Ciò presuppone che le uscite siano indipendenti l'una dall'altra, il che potrebbe non essere un presupposto corretto. Tuttavia, questo approccio può fornire previsioni sorprendentemente efficaci su una serie di problemi e può valere la pena provare, almeno come riferimento per le prestazioni. Ad esempio, gli output per un problema possono, infatti, essere per lo più indipendenti, se non completamente indipendenti, e questa strategia può aiutarti a scoprirlo. Questo approccio è supportato dalla classe MultiOutputRegressor che accetta un modello di regressione come argomento. Quindi creerà un'istanza del modello fornito per ogni output del problema.

```
...
# define base model
model = LinearSVR()
# define the direct multioutput wrapper model
wrapper = MultiOutputRegressor(model)
```

L’altro approccio che utilizza modelli di regressione a output singolo per la regressione a output multipli, consiste nel creare una sequenza lineare di modelli. Il primo modello della sequenza utilizza l'input e prevede un output; il secondo modello utilizza l'input e l'output del primo modello per fare una previsione; il terzo modello utilizza l'input e l'output dei primi due modelli per fare una previsione e così via. Ad esempio, se un problema di regressione a più uscite richiedeva la previsione di tre valori y1, y2 e y3 dato un input X , allora questo potrebbe essere suddiviso in tre problemi di regressione a singola uscita dipendenti come segue:

Problema 1 : dato X , prevedere y1 .

Problema 2 : dati X e yhat1 , prevedere y2 .

Problema 3 : dati X, yhat1 e yhat2 , prevedere y3 .

Ciò può essere ottenuto utilizzando la classe RegressorChain nella libreria scikit-learn. L'ordine dei modelli può essere basato sull'ordine degli output nel dataset (predefinito) o specificato tramite l’argomento " order ". Ad esempio, order = [0,1] predirebbe prima il primo output, mentre un order = [1,0] predirebbe prima l'ultima variabile di output e poi la prima variabile di output nel nostro problema di test. L'esempio seguente mostra come creare prima un modello di regressione a output singolo, quindi utilizzare la classe RegressorChain per eseguire il wrapping del modello di regressione e aggiungere il supporto per la regressione a più output.

```
...
# define base model
model = LinearSVR()
# define the chained multioutput wrapper model
wrapper = RegressorChain(model, order=[0,1])
```

Di seguito quindi riportiamo la valutazione di alcuni algoritmi testati per il dataset multi output.

Algoritmo	mean_absolute_error	Standard_deviation
LinearRegression	5.532	0.730
KNeighborsRegressor	3.575	0.623
RandomForestRegressor	1.779	0.548
DecisionTreeRegressor	1.354	0.578
Wrapper LinearSVR	2.774	0.239
---	---	---

Il modello che produce i migliori risultati è DecisionTreeRegressor, il codice per addestrare il modello è il seguente:

```
import pandas as pd
import numpy as np
var = pd.read_csv('Risultati.csv',sep=";")
y_train=var.loc[:, 'P1':'P18']
X_train = var.drop(['P1','P2','P3','P4','P5','P6','P7','P8','P9','P10','P11','P12','P13',
'P14','P15','P16','P17','P18'], axis=1)
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor()
model.fit(X_train,y_train)
row= X_train.iloc[106,:]
yhat = model.predict(np.array([row]))
print(yhat[0])
```

Produce il seguente output :

```
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. 30.  0.  0.  0.  0.  0.  0.]
```

Ovvero indica che: il pilastro N°11 presenta un danno che può essere quantificato come una riduzione della rigidezza del 30%.

Per la riga 107 data come input otterremo il seguente output:

```
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. 20.  0.  0.  0.  0.  0.  0.]
```

La previsione delle prime 20 tuple del dataset sarà il seguente:

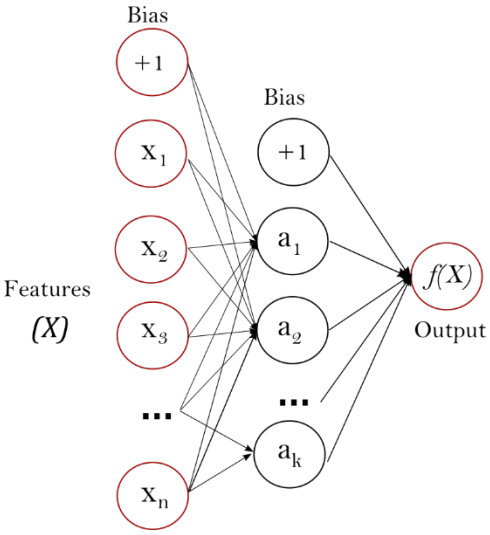
```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[90.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[80.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[70.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[60.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[50.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[40.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[30.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[20.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[10.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[ 0. 90.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[ 0. 80.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
```

```
[ 0. 70.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[ 0. 60.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[ 0. 50.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[ 0. 40.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[ 0. 30.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[ 0. 20.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[ 0. 10.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[ 0.  0. 90.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
```

Regression analysis con rete neurale

Una rete neurale di tipo Multi-layer Perceptron (MLP) è un algoritmo di apprendimento supervisionato che apprende una funzione del tipo $f(\cdot):Rm \rightarrow Ro$. Funzione appresa attraverso l'addestrando su un set di dati, dove "m" è il numero di dimensioni per l'input e "o" è il numero di dimensioni per l'output.

Dato un insieme di funzionalità X=x1,x2,...,xm e un obiettivo y, un MLP può apprendere un approssimatore di funzione non lineare sia per la classificazione che per la regressione. MLP è diverso dalla regressione logistica, in quanto tra il livello di input e quello di output possono esserci uno o più livelli non lineari, chiamati livelli nascosti. La Figura 1 mostra un MLP a un livello nascosto con output scalare.



I vantaggi di Multi-layer Perceptron sono:

- Capacità di apprendere modelli non lineari.
- Capacità di apprendere modelli in tempo reale (apprendimento in linea) utilizzando partial_fit.

Gli svantaggi di Multi-layer Perceptron (MLP) includono:

- Le MLP con strati nascosti hanno una funzione di perdita non convessa dove esiste più di un minimo locale. Pertanto diverse inizializzazioni di peso casuale possono portare a una diversa accuratezza di convalida.
- MLP richiede la messa a punto di un numero di iperparametri come il numero di neuroni, livelli e iterazioni nascosti.
- MLP è sensibile al ridimensionamento delle funzionalità.

La classe MLRegressor implementa un perceptron multistrato (MLP) che si allena utilizzando backpropagation senza funzione di attivazione nel livello di output, che può anche essere visto come l'utilizzo della funzione di identità come funzione di attivazione. Pertanto, utilizza l'errore quadratico come funzione di perdita e l'output è un insieme di valori continui.

La rete neurale viene realizzato attraverso il seguente codice:

```
import pandas as pd
import numpy as np
var = pd.read_csv('Risultati.csv',sep=";")
y_train=var.loc[:, 'P1':'P18']
X_train = var.drop(['P1','P2','P3','P4','P5','P6','P7','P8','P9','P10','P11','P12','P13',
'P14','P15','P16','P17','P18'], axis=1)
from sklearn.neural_network import MLPRegressor
model = MLPRegressor(hidden_layer_sizes=(64,64,64),activation="relu" ,random_state=1, max
_iter=2000)
model.fit(X_train, y_train)
row= X_train.iloc[106,:]
yhat = model.predict(np.array([row]))
print(yhat[0])
```

Produce il seguente output :

[-1.44132465e+00 -8.80285652e-01 4.17084597e+00 -2.64370097e-01 1.77113361e+00 -2.56354682e+00 -8.95298799e-01 1.14421553e+00

-1.22708377e+00 6.58342579e-01 -1.90245829e+00 **1.62880546e+01** 2.83826034e+00 -4.23523040e-01 -2.06519682e-01 1.07777802e-02

-9.87980309e-02 **1.59330099e+01**]

Identifica come il pilastro N°11 con danneggiamento insieme al pilastro N°18.

Con il dataset a disposizione si ottiene la seguente metrica:

```
mean_absolute_error:  4.954238651637497
mean_squared_error:  123.34915313257994
r2_score:  0.2623759510286934
```

Per migliorare le prestazioni del modello occorre quindi realizzare un dataset più popolato.

Ottimizzazione del modello attraverso il ricampionamento del dataset

I metodi di ricampionamento sono uno strumento indispensabile nelle statistiche moderne. Consistono nel prelevare ripetutamente campioni da un set di addestramento e nel riadattare un

modello di interesse su ciascun campione al fine di migliorare le informazioni nel modello adattato. Di solito, l'obiettivo di un progetto di data science è creare un modello utilizzando i dati di addestramento e fare in modo che faccia previsioni su nuovi dati. Spesso questa metodologia viene adoperata per generare un Testset di dati per la convalida di un modello, senza la necessità di raccogliere nuovi dati. Infine un altro uso è per bilanciare un dataset con classi sbilanciate.
Un set di dati di regressione (X, y) può essere quindi ricampionato per mitigare lo squilibrio nella distribuzione utilizzando una delle seguenti strategie:

- sovracampionamento casuale (RO)
- sottocampionamento casuale (RU)
- SMOTER
- rumore gaussiano (GN)
- WERCS
- Rebagg

Il primo metodo va a sovraccampionare i valori rari nella distribuzione, selezionati dall'utente tramite una funzione di rilevanza. Il secondo metodo esegue l'operazione contraria, ovvero va ad sottocampionare i valori che sono invece più abbondanti nella distribuzione.
La funzione SMOTER combina insieme i due metodi precedenti, ovvero sottocampiona i valori abbondanti e sovracampiona i valori rari mediante interpolazione tra punti vicini.
Il quarto metodo aggiunge un rumore gaussiano alla funzione SMOTER nel ricampionamento.
Mentre WERCS utilizza un ricampionamento selettivo, ovvero ricampiona il set di dati selezionando istanze utilizzando i valori di pertinenza specificati dall'utente come pesi.
Infine il metodo REBAGG esegue un ricampionamento su sottoinsiemi del set di dati in modo indipendente.
Attraverso il ricampionamento con metodo SMOTER si ottiene un dataset con cardinalità di 1539 elementi.

```
import numpy as np
import resreg

cl1 = np.percentile(y,10)  # Oversample values less than the 10th percentile
ch1 = np.percentile(y,90)  # Oversample values less than the 10th percentile
relevance = resreg.sigmoid_relevance(y, cl=cl1, ch=ch1)
X_res, y_res = resreg.smoter(X_train, y_train, relevance=relevance, relevance_threshold=0
.5, k=5, over='balance', random_state=0)
```

Producendo un aumento delle prestazioni di previsione:

```
mean_absolute_error:  0.29415294119556157
mean_squared_error:  9.782488203363632
r2_score:  -5.6825127463658e-05
```

Il metodo Gaussian_noise produce il seguente il seguente errore di previsione:

```
import numpy as np
import resreg
relevance = resreg.sigmoid_relevance(y_train, cl=None, ch=np.percentile(y_train, 90))
X_res, y_res = resreg.gaussian_noise(X_train, y_train, relevance, relevance_threshold=0.5
,delta=0.1, over=0.5, under=0.5, random_state=0)

mean_absolute_error:  4.144066080465169
mean_squared_error:  76.88252716253115
r2_score:  0.5408364622068045
```

Ma si avvicina quanto più possibile alla casistica delle misure che verranno prodotte sul campo. Come possiamo notare dalla previsione delle prime venti righe del dataset, la rete riesce ad individuare la collocazione del pilastro danneggiato, ma sulla percentuale di danneggiamento ancora richiede un processo di addestramento.

```
[-1. -2.  2.  2. -0.  1. -3. -3. -2. 11. 12.  6.  0. 12.  1. 10. 15.  6.]
[46. -0. -4. 11. -1.  6.  1. -1.  0.  5.  3. -1.  8. -1. -9. -2.  9. -2.]
[77. 10. -3.  5.  1.  4. -4.  1. -0. -2. -6. -1.  1. -5.  1. -6.  4.  0.]
[ 72.   9.  -7.   2.  -1.   2.  -3.   4.  -2.  -7.  -3.  -2.  -1.   1.
  10. -12.   2.  -1.]
[57.  5. -5.  1. -1.  1. -2.  3. -1. -5. -3. -1. -1.  2.  7. -8.  2. -1.]
[43.  4. -1.  0. -0. -0. -0.  1.  0. -3. -3. -1. -0.  1.  3. -3.  1. -0.]
[31.  3.  1.  0.  0. -1.  1.  0.  1. -1. -2. -1.  0.  1.  0.  1.  0. -0.]
[24.  3.  3. -0.  0. -1.  2. -0.  2.  0. -2. -1.  1.  1. -2.  3.  0.  0.]
[19.  3.  5. -0.  0. -1.  3. -1.  2.  1. -1. -0.  1.  0. -3.  5.  0.  0.]
[16.  3.  5.  0.  0. -2.  3. -1.  2.  1. -1. -0.  1.  0. -4.  6.  0.  0.]
[-4. 27.  6.  0.  3. -1.  3. -4.  3.  0. 19. -2.  2.  8. -3. -1. -1. -1.]
[ 1. 51.  5.  2.  2. -2.  1. -5.  5. -6.  5. -2.  5.  0. -2.  2. -5. -0.]
[ 1. 49.  3.  3. -1. -0. -1. -5.  1. -5.  3. -1.  3.  0. -1.  0. -1. -1.]
[ 0. 43. -1.  1.  0.  0. -2. -4. -0. -4.  2. -1.  2. -1. -0.  1.  1. -0.]
[ 0. 38. -3. -0.  1.  0. -3. -3. -1. -3.  2. -0.  1. -2.  0.  1.  3.  0.]
[ 0. 35. -5. -1.  2.  1. -4. -3. -2. -2.  1. -0.  1. -3.  1.  1.  4.  0.]
[ 1. 32. -4. -2.  3. -0. -4. -2. -2. -1.  2. -0.  0. -4.  2. -0.  3. -0.]
[ 1. 30. -4. -2.  4. -1. -4. -1. -2.  0.  2. -0. -0. -6.  2. -1.  3. -1.]
[ 1. 29. -4. -2.  5. -1. -4. -1. -1.  1.  2. -0. -1. -6.  2. -1.  3. -1.]
[ -1.  -1.  55.   6.  -1.   9.   0.  -2.  -4.   1.   5.  19.  -8.  -6.
  20. -10.   1.  14.]
```

Regression analysis con Keras

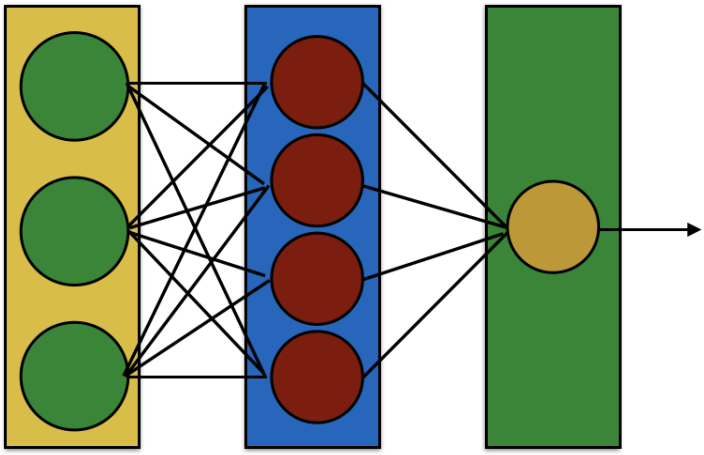
Keras è una libreria open source nata per facilitare lo sviluppo di applicazioni che sfruttano tecniche di apprendimento automatico basate su reti neurali. La libreria è stata scritta in Python ed è stata progettata come un'interfaccia di astrazione per altre librerie di più basso livello, in particolare TensorFlow, Microsoft Cognitive Toolkit (CNTK) e Theano i quali in Keras sono identificati con il nome di Backend.

Keras semplifica, pertanto, la vita a tutti coloro vogliano approcciare al mondo delle reti neurali, ma non hanno interesse ad approfondire le tecniche matematiche connesse all'uso delle stesse. La libreria nel suo complesso è veramente interessante perché è stata progettata con l'obiettivo di consentire una

rapida prototipazione di modelli neurali, inoltre supporta oltre alle classiche reti feedforward, anche modelli convoluzionali e reti ricorrenti e schemi di connettività arbitrari. Essa, lavora inoltre senza problemi sia su CPU e che GPU. Con Keras è infatti possibile, creare modelli complessi con un numero minimo di righe di codice mantenendo lo stesso robusto, e consente nel contempo una grande flessibilità grazie alla sua caratteristica modulare. In Keras il concetto di modello, infatti, è inteso come una sequenza o un grafo di singoli moduli che possono essere assemblati insieme, per creare nuovi modelli, con il minimo numero di restrizioni possibili. Le reti neurali, le funzioni di costo, gli ottimizzatori, gli schemi di inizializzazione, le funzioni di attivazione, gli schemi di regolarizzazione sono tutti visti in Keras come moduli e possono essere combinati tra loro per crearne di nuovi.

Keras basa la sua logica sul concetto di **modello** che è un modo semplice per organizzare i livelli della rete neurale. Il modello più comunemente usato in Keras è quello sequenziale ma esiste anche la possibilità di usare il modello funzionale.

Bisogna tener presente che il **model sequential** è lo strumento che ci consente di infilare in cascata i livelli di una rete neurale, specificando volta per volta il tipo di livello, ad esempio il livello di input (giallo) il livello hidden (blu) ed il livello di output (verde).



Sebbene sia possibile creare reti neurali complesse, nel nostro caso ci limiteremo a costruire una rete neurale a tre livelli, ovvero facendo riferimento alla terminologia usata in keras implementeremo una sola sequenza.

Il model Sequential come già detto è la classe che ci consentirà di aggiungere i Livelli partendo dall'input fino all'output, mentre il dense Layer è un livello generico di una rete neurale feed forward full connected, che si importa nel seguente modo:

```
from keras.layers import Dense
```

La classe Dense, è un normale livello composto da n neuroni; costituisce il classico livello di una rete neurale artificiale in cui gli input vengono pesati e trasferiti attraverso la funzione di attivazione all'output insieme al bias.

Usando questi due strumenti, messi a disposizione in keras, sarà possibile implementare reti neurali generiche composte da uno strato di input, n strati intermedi (hidden) e uno strato di output.

Bene vediamo come sia possibile istruire keras ad implementare una rete neurale, usando Sequential e Dense; basterà istanziare il modello di tipo sequential e definire i livelli di input e di output.

```
model = Sequential()
model.add(Dense(256, input_shape=(X_train.shape[1],)))
model.add(Dense(128))
model.add(Dense(180, activation='sigmoid'))
```

In questo esempio abbiamo definito una rete neurale (MLP) con 24 neuroni nel livello di input, un livello intermedio (hidden) costituito da 128 neuroni ed un livello di uscita con 180 neuroni : una uscita per ogni classe o etichetta, con funzione di trasferimento sigmoide. (funzione logistica)

a questo punto non ci resta che decidere quale algoritmo utilizzare per l'addestramento della rete; in altri termini l'algoritmo che calcolerà il minimo della funzione di costo. Per lo scopo utilizzeremo la funzione **compile** sul modello sequential definito prima ed utilizzeremo come optimizer rmsprop (*Root Mean Square Propagation*) ovvero il tipo di algoritmo che si basa sul calcolo dell'errore quadratico medio per il calcolo del minimo della funzione di costo.

Come metrica per valutare la qualità del training useremo l'accuracy : maggiore sarà l'accuracy migliore sarà l'addestramento (in percentuale) mentre per la funzione di perdita (loss) useremo la 'categorical _crossentropy' che viene usata nei casi di classificazione in cui l'uscita della rete neurale ha più uscite categorizzate:

```
model.compile(loss="categorical_crossentropy", optimizer='sgd', metrics=["ac
curacy"])
```

a questo punto non ci resta che avviare la fase di training e decidiamo di fermare il training dopo 100 epoche con batches di 16 elementi per volta. Con validation_split =0.2 stiamo dicendo che usiamo l'80% del dataset per l'addestramento ed il 20% per la validazione.

```
H = model.fit(X_train, y_train, validation_split=0.2, epochs=100, batch_size
=16, verbose=1)
```

Adesso possiamo misurare l'accuratezza:

```
plt.figure()
plt.plot(np.arange(0, 100), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, 100), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, 100), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, 100), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
```

```
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend()
plt.show()
```

Il codice complete per realizzare un modello di regressione multioutput è il seguente:

```
import pandas as pd
from numpy import asarray
import numpy as np
from matplotlib import pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# get the dataset
def get_dataset():
  var = pd.read_csv('Risultati.csv',sep=";")
  y_train=var.loc[:, 'P1':'P18']
  X = var.drop(['P1','P2','P3','P4','P5','P6','P7','P8','P9','P10','P11','P1
2','P13','P14','P15','P16','P17','P18'], axis=1)
  y_train['P1'] = pd.Categorical(y_train.P1)
  y_train['P2'] = pd.Categorical(y_train.P2)
  y_train['P3'] = pd.Categorical(y_train.P3)
  y_train['P4'] = pd.Categorical(y_train.P4)
  y_train['P5'] = pd.Categorical(y_train.P5)
  y_train['P6'] = pd.Categorical(y_train.P6)
  y_train['P7'] = pd.Categorical(y_train.P7)
  y_train['P8'] = pd.Categorical(y_train.P8)
  y_train['P9'] = pd.Categorical(y_train.P9)
  y_train['P10'] = pd.Categorical(y_train.P10)
  y_train['P11'] = pd.Categorical(y_train.P11)
  y_train['P12'] = pd.Categorical(y_train.P12)
  y_train['P13'] = pd.Categorical(y_train.P13)
  y_train['P14'] = pd.Categorical(y_train.P14)
  y_train['P15'] = pd.Categorical(y_train.P15)
  y_train['P16'] = pd.Categorical(y_train.P16)
  y_train['P17'] = pd.Categorical(y_train.P17)
  y_train['P18'] = pd.Categorical(y_train.P18)
  y = pd.get_dummies(y_train, prefix=['P1','P2','P3','P4','P5','P6','P7','P8
','P9','P10','P11','P12','P13','P14','P15','P16','P17','P18'])
  return X, y

# get the model
def get_model(n_inputs, n_outputs):
  model = Sequential()
  model.add(Dense(256, input_dim=n_inputs, kernel_initializer='he_uniform',
activation='relu'))
  model.add(Dense(128, kernel_initializer='he_uniform', activation='relu'))
  model.add(Dense(n_outputs, kernel_initializer='he_uniform'))
  #model.compile(loss='mae', optimizer='adam')
  model.compile(loss="mean_squared_error", optimizer='adam')
  return model

# load dataset
X, y = get_dataset()
n_inputs, n_outputs = X.shape[1], y.shape[1]
# get model
model = get_model(n_inputs, n_outputs)
# fit the model on all data
```

```
#model.fit(X, y, verbose=0, epochs=100)
H=model.fit(X, y, validation_split=0.2, epochs=100, batch_size=16, verbose=0
)
```