ALMA MATER STUDIORUM
UNIVERSITÀ DEGLI STUDI DI BOLOGNA

Scuola di Ingegneria e Architettura
Campus di Cesena
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

ENERGY MANAGEMENT SYSTEM: PROGETTAZIONE E SVILUPPO DI UNO STRUMENTO DI STIMA E ANALISI DI VARIABILI ENERGETICHE

Tesi in Project Management

Tesi di Laurea di:
ELIA ZOFFOLI

Relatore:
Prof. MARCO ANTONIO BOSCHETTI

Correlatore:
Ing. CLAUDIO GAMBETTI

ANNO ACCADEMICO 2013–2014
SESSIONE III

PAROLE CHIAVE

Supervisory Control and Data Acquisition System

Energy Management System

Stima di variabili

Analisi di Regressione

On.Energy

Alla mia famiglia ai mie amici e compagni per avermi sostenuto ed accompagnato durante tutto il percorso di studi.
Alle mie bariste preferite per avermi aiutato a superare i momenti no con un sorriso.
Un ringraziamento speciale alla Onit Group ed al professor Marco Antonio Boschetti per avermi dato la possibilità di intraprendere questo percorso di tesi.

Indice

Introduzione

Al giorno d'oggi i sistemi con cui un'azienda si trova a dover operare sono diventati sempre più complessi e di difficile gestione. L'apporto che il settore elettronico–informatico ha portato nell'ultimo decennio in questo ambito è stato sempre più rilevante, difatti è nata un'intera branca dedicata allo sviluppo di sistemi automatizzati per la gestione dei vari aspetti della vita aziendale. Tra le diverse tipologie di sistemi informatici ed elettronici di supporto alle attività aziendali figurano i *Supervisory Control and Data Acquisition System* (SCADA) che si occupano dell'acquisizione di informazioni dai diversi processi produttivi, per permettere ai manager aziendali di effettuare operazioni di analisi e controllo per supervisionare l'andamento dell'azienda stessa.
Viste le diverse e specifiche necessità che ogni settore presenta, anche i sistemi SCADA sono stati adattati e diversificati per rispondere alle varie esigenze riscontrabili nei diversi ambiti aziendali, pertanto sul mercato sono disponibili diverse tipologie di sistemi SCADA specializzate per ciascun settore di applicazione.

Uno dei campi di applicazione che negli ultimi anni ha visto un note-

vole sviluppo e cambiamento è quello energetico. I sistemi di supervisione e controllo che si occupano del settore di produzione e consumo di energia sono detti SCADA Energy Management (SCADA/EMS). È proprio questo l'ambito che è stato scelto per lo sviluppo del progetto di tesi, grazie alla collaborazione con l'azienda Onit Group srl che ha deciso di estendere le funzionalità del suo sistema On.Energy Management System potenziandone le capacità di analisi fornite al cliente.

Questo elaborato di tesi è nato con l'esigenza di sviluppare un nuovo modulo per la stima di variabili energetiche da inserire nel software On.Energy, per dare la possibilità a tutti gli utilizzatori di capire quanto i valori osservati nella realtà si discostino da un modello teorico appositamente creato, e fornire quindi un ulteriore strumento di analisi per mantenere sotto controllo il sistema nell'ottica del miglioramento continuo. Il risultato sarà uno strumento che verrà provato in via sperimentale sugli stabilimenti di due aziende leader in Italia in settori produttivi largamente differenti (Amadori - alimentare, Pfizer - farmaceutico), ma accomunati dalle esigenze di monitorare, analizzare e efficientare i consumi energetici.

La tesi è stata suddivisa in 5 capitoli:

1. Nel primo capitolo di è cercato di dare un'introduzione ai sistemi SCADA ed Energy Management in particolare, catturandone tutte le peculiarità e le funzionalità considerate fondamentali per sistemi di questo tipo. Per i sistemi SCADA/EMS ci si è concentrati sul con-

cetto di miglioramento continuo che questi hanno alla base definendo standard e caratteristiche da rispettare.

2. Il secondo capitolo definisce uno dei metodi di stima tra i più noti in letteratura, l'analisi di regressione, considerandone gli aspetti generali e matematici, analizzando i metodi di risoluzione conosciuti, oltre che gli indicatori per verificarne l'efficienza e l'efficacia. In questo stesso capitolo ci si è chiesti anche se fossero disponibili strumenti informatici di supporto all'applicazione del metodo di regressione, andando alla ricerca di librerie che forniscano le funzionalità necessarie.

3. Nel terzo capitolo si è approfondito il software On.Energy Management System sviluppato dall'azienda Onit Group srl. Lo scopo di questo terzo capitolo è uno studio dei concetti e delle funzionalità presenti nel software, che è utile conoscere prima di poter iniziare con l'analisi del nuovo modulo.

4. Nel quarto capitolo sono state affrontate le fasi di analisi dei requisiti, del problema per permettere al lettore di capire quali siano state le motivazioni che hanno spinto alla realizzazione del nuovo modulo e quali siano i vincoli, caratteristiche e criticità attinenti a ciò che si è sviluppato. Il capitolo prosegue con la definizione delle tecnologie utilizzate prestando particolare attenzione al pattern di progettazione che si è deciso di adottare per il sistema in esame.

5. Infine nel quinto capitolo viene analizzata la struttura che si è deciso di dare al nuovo modulo e successivamente vengono presentate le

interfacce che un utente incontra durante il suo utilizzo descrivendole nei dettagli.

Capitolo 1

Supervisory Control and Data Acquisition System

Questo primo capitolo ha l'obiettivo di fornire un'introduzione ai sistemi di supervisione e controllo per consentire al lettore di comprendere l'importanza e lo scopo di un loro utilizzo in ambito aziendale.
In particolare verranno analizzati sistemi di tipo Energy Management System, che sono una particolare tipologia di sistemi SCADA che lavorano nell'ambito della produzione di energia e del controllo dei consumi, questi sistemi sono anche detti SCADA/EMS.
È in questo settore che verrà sviluppato il progetto di tesi ed è quindi importante fornirne un'introduzione per dare al lettore la possibilità di avere una visione completa a riguardo.

1.1 SCADA System

SCADA (Supervisory Control and Data Acquisition), nell'ambito dei controlli automatici, indica quell'insieme di sistemi utilizzati per il monitoraggio e controllo elettronico remoto di sistemi fisici. Come suggerisce l'acronimo stesso le funzionalità si dividono in due parti: l'acquisizione di dati dal mondo esterno e il controllo e la supervisione del sistema da gestire. Per controllo in particolare si intende la possibilità di un utilizzatore di cambiare in tempo reale i parametri del sistema in modo da modificarne il funzionamento.

Una caratteristica che ha portato ad una così ampia diffusione dei sistemi SCADA è la possibilità di avere componenti distribuiti geograficamente, infatti non è necessario che le varie parti siano situate nello stesso luogo (spesso questo non è nemmeno possibile), ma è sufficiente avere l'infrastruttura per permetterne la comunicazione. Si possono avere quindi casi in cui i componenti di raccolta dati siano distribuiti in vari punti nel territorio e comunichino (ad esempio attraverso internet) con un'unità centrale di elaborazione a cui si possono collegare i diversi dispositivi di controllo in possesso degli utilizzatori. Perciò si può parlare di un sistema centralizzato di controllo ed elaborazione e di un'infrastruttura di sensori (raccolta dati) e terminali dislocati sul territorio.

A livello industriale la diffusione dei sistemi SCADA è dovuta principalmente alla crescita del numero e della complessità dei macchinari utilizzati,

diventati ingestibili senza supporto informatico. Inoltre volendo intraprendere un percorso di miglioramento continuo risulta molto importante il controllo dei processi aziendali e la storicizzazione dei dati da essi derivati. I sistemi SCADA permettono di avere informazioni utili in tempo reale su apparecchiature distribuite (e quindi individuare possibili sprechi e malfunzionamenti) e di memorizzarle per un'analisi successiva.

La distribuzione dei componenti e l'utilizzo di canali di comunicazione nei sistemi SCADA porta anche un insieme di problemi ad essi legati. È innanzitutto importante la sicurezza dei dati, che non devono essere accessibili da estranei; solitamente viene utilizzata la crittografia ed il controllo degli accessi nei terminali, questo però rallenta ed appesantisce la computazione e la trasmissione dei dati.
Altro problema è legato alle tempistiche di aggiornamento delle informazioni e di trasmissione dei comandi. Spesso infatti si vorrebbero avere informazioni real–time e si vorrebbe che i comandi impartiti fossero subito presi in carico ed attuati (in caso di malfunzionamenti critici potrebbe essere fondamentale), ma in sistemi distribuiti questo non è scontato (si deve considerare il tempo di trasmissione ed il carico della rete).

1.1.1 Funzionalità di base

Indipendentemente dal campo di applicazione ogni sistema SCADA possiede delle caratteristiche (o funzionalità) comuni fondamentali per il loro impiego. Queste sono:

- Acquisizione dei dati - È una funzionalità di base indispensabile per il funzionamento di qualsiasi sistema SCADA e deve precedere l'esecuzione delle altre funzionalità. La difficoltà dell'implementazione consiste nello scambio delle informazioni tra sistemi estremamente eterogenei sia a livello di tecnologie sia a livello di protocolli utilizzati per lo scambio di informazioni. Va inoltre aggiunto che alla base della rilevazione dei dati ci sono comunque dei sensori che producono segnali elettrici, perciò è necessario infrapporre dei sistemi di conversione che traducano il segnale elettrico in un dato digitale che possa essere correttamente interpretato dal sistema che svolgerÃ l'elaborazione.

- Supervisione dei dati - È necessaria per un costante controllo del sistema; tramite questa funzionalità, l'utente supervisore ha la possibilità di verificare la corretta evoluzione del sistema. L'attività di supervisione può essere svolta in due modalità: la prima permette la verifica dello stato attuale, mentre la seconda permette un'analisi più ampia dei dati storicizzati per monitorarne l'andamento.

- Controllo dei dati - Rappresenta la capacità di intervenire su un sistema e decidere come cambiarne l'evoluzione. Normalmente la stessa architettura che viene utilizzata per l'acquisizione del dato è sfruttata anche per modificare il dato stesso.

1.1.2 Componenti del sistema

I componenti costituenti un sistema SCADA possono essere molti e di tipo diverso, ognuno però svolgerà una delle funzioni sopra descritte comunican-

do con gli altri per realizzare il sistema nel suo complesso.

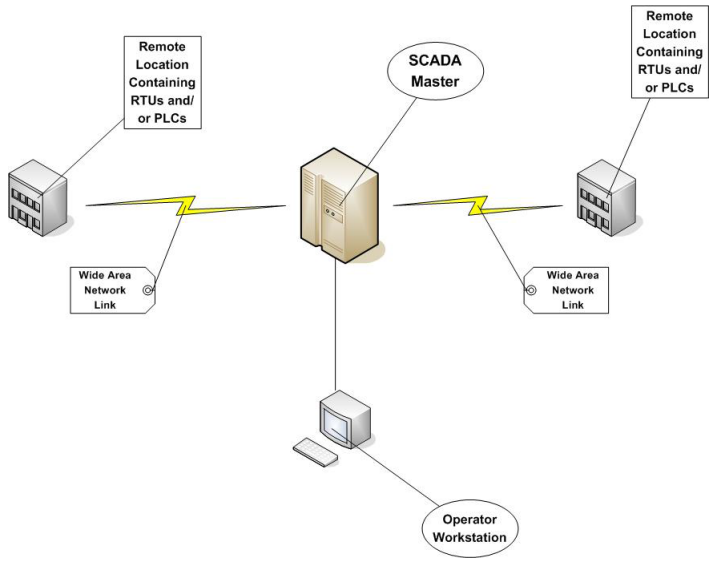


Figura 1.1: Esempio di sistema SCADA [25]

In genere il sistema è composto da:

- Uno o più sensori o attuatori (definiti anche come dispositivi sul campo o field equipment) che interagiscono con device di interfaccia. I sensori possono essere considerati come "gli occhi o le orecchie" del sistema (ad esempio termostati, misuratori di livello, etc.), mentre gli attuatori possono essere visti come le "mani" (ad esempio valvole elettriche, interruttori) che permettono il controllo automatizzato del sistema. Comunque, prima di poter effettuare qualsiasi operazione, si rende necessario convertire i segnali elettrici prodotti da questi dispositivi in dati gestibili da un sistema informatico (o viceversa nel caso degli attuatori). Per raggiungere questo obiettivo si utilizzano dei

dispositivi elettronici di interfaccia come RTU (Remote Terminal Units) o PLC (Programmable Logic Controller). Questi dispositivi oltre ad eseguire una mera conversione di segnale possiedono una capacità computazionale e possono essere programmati per eseguire anche operazioni di controllo e campionamento (ad esempio si può programmare un RTU o PLC per governare l'apertura di una valvola e mantenere un livello di liquido programmato a meno di comandi diversi impartiti dall'utente, oppure si può decidere di inviare il dato ogni 15 minuti o solo quando esso varia), in questo modo si evita di avere in circolazione informazioni ridondanti ed inutili.

- L'unità centrale o Central Host rappresenta il collegamento tra i dispositivi sul campo e gli utilizzatori, questa ha un triplice compito: Front-End, Computazione e archiviazione dei dati.
  Per front-end si intende la capacità di rendere il sistema fruibile da parte degli utenti che vogliono accedervi; l'unità centrale dovrà fornire dei servizi per permettere e controllare l'accesso.
  L'elaborazione dei dati è un altro compito molto importante; i dati ricevuti dai vari PLC e RTU dovranno essere elaborati in modo da essere resi visibili e comprensibili agli utenti; questa è una funzione fondamentale, in quanto il semplice dato potrebbe non essere molto informativo per l'utente, mentre lo stesso dato rappresentato graficamente potrebbe essere molto più utile.
  L'archiviazione dei dati è utile per operazioni di analisi successive, ad esempio per capire l'andamento del sistema in un periodo. La mole di

informazioni da memorizzare può essere anche molto elevata, è quindi importante avere strutture appositamente progettate (ad esempio un sistema informativo ad hoc).

- La rete di comunicazione costituisce la spina dorsale dei sistemi SCADA. Questa si occupa della trasmissione di informazioni tra i dispositivi dislocati sul campo (sensori/attuatori), l'unità centrale e i terminali in mano agli utenti. I mezzi trasmissivi che è possibile utilizzare possono essere i più disparati: da segnali elettrici attraverso cavo (ad esempio coassiale), a trasmissioni satellitari, fino a onde radio (WI-FI).

  Storicamente si utilizza una rete dedicata per sistemi SCADA, ma con il crescere dell'utilizzo di reti LAN e WAN ci si è spostati su questo tipo di soluzione in modo da non dover investire su un'infrastruttura dedicata.

- Un ultimo componente da considerare nei sistemi SCADA è il software usato per interagire con il sistema. Con l'avvento del web questo si è sempre più tradotto in una interfaccia fruibile via web browser e residente direttamente nell'unità centrale, ma esistono comunque casi di software installati su macchine locali che comunicano con l'unità centrale solo per lo scambio di dati. Questi software sono molto spesso proprietari e sviluppati dagli stessi produttori hardware del sistema, perciò difficilmente comunicano con componenti estranei.

  Una parte importante del software consiste nella gestione degli allarmi. Gli allarmi sono eventi inattesi del sistema, che devono essere

comunicati all'utente per consentire l'attuazione di contromisure. È importante trovare il giusto modo di segnalare questi eventi, ad esempio può essere inserito un messaggio nell'interfaccia software oppure inviata una mail o un messaggio automaticamente.

Nell'immagine 1.1 si può vedere un tipico esempio di sistema SCADA composto da due dispositivi sul campo che comunicano con l'unità centrale e una workstation che permetterà all'utente di interagire con il sistema.

1.1.3 Evoluzione dei sistemi SCADA

I sistemi SCADA si sono evoluti in parallelo con le moderne tecnologie di computazione. Si possono individuare quattro generazioni di sistemi SCADA:

1. Monolithic system
2. Distributed system
3. Networked system
4. Internet of things generation

Monolithic system

Alla nascita dei sistemi SCADA l'informatica era basata sui *mainframe*, la rete ancora non era sviluppata, come risultato un sistema SCADA era composto da un'unica grande unità di computazione senza connettività con il mondo esterno.

In pratica il mainframe era posto vicino al sistema fisico da tenere sotto controllo e collegato con i sensori tramite bus, utilizzando un protocollo di comunicazione proprietario e diverso da produttore a produttore.

Distributed system

La seconda generazione di sistema SCADA si sviluppa sull'evoluzione della rete locale LAN per distribuire la computazione su più terminali. Tutti i terminali sono connessi gli uni agli altri scambiandosi informazioni ed ognuno realizza una funzione specifica (ad esempio, alcuni terminali si occupano del protocollo di comunicazione, altri dell'interfacciamento con l'utente, mentre altri dell'elaborazione e memorizzazione dei dati). I singoli terminali sono dei mini-computer, perciò si ha una riduzione dei costi rispetto al mainframe.
La distribuzione delle funzionalità non è utile solamente per aumentare la potenza di calcolo, ma anche per aumentare l'affidabilità del sistema e la ridondanza dei dati (più sicuro in caso di guasti).
Il principale problema di questa seconda generazione rimane l'utilizzo di protocolli di comunicazione proprietari che non ne permettono l'utilizzo su reti di tipo WAN e li limitano a reti locali LAN.

Networked system

La terza generazione di sistemi SCADA è molto simile alla seconda, con la differenza dell'uso di protocolli standard per le comunicazioni (come ad esempio il protocollo IP). Questo ne permette un ulteriore evoluzione grazie all'utilizzo di reti non proprietarie ed internet, e quindi un'ulteriore distri-

buzione dei componenti che non hanno praticamente più limiti in termini di distribuzione spaziale.

Internet of things generation

In generale il concetto di "Internet of Things" (IoT) è usato per descrivere uno scenario dove gli oggetti e la loro rappresentazione virtuale sono contenuti in una organizzazione virtuale come internet. Quando un oggetto è creato e messo in circolazione tutte le informazioni su esso sono continuamente aggiornate senza l'intervento dell'uomo, l'idea è aumentare il valore degli oggetti sapendo esattamente quando questi hanno bisogno di interventi [36].
Il concetto di IoT è perfetto per i sistemi SCADA. Oggetti in grado di trasmettere il proprio stato si tradurrebbero in molte più informazioni da poter raccogliere ed analizzare, tutte queste informazioni sarebbero rese disponibili online, infatti l'idea dell'IoT si basa sul concetto di cloud computing.
Con la commercializzazione delle infrastrutture cloud la distribuzione dei sistemi SCADA si è potuta ulteriormente evolvere sfruttando l'estrema scalabilità del cloud stesso. Inoltre l'utilizzo di protocolli sicuri standard come il Transport Layer Security (ampiamente utilizzati nel cloud computing) ha permesso un miglioramento della sicurezza e dell'affidabilità nelle trasmissioni di dati.

1.2 Energy Management System

In questa sezione si introdurranno i sistemi di gestione e controllo di energia (SCADA/EMS); questi sistemi si dedicano al monitoraggio dei consumi e della produzione di energia.
Un Energy Management System (EMS) è un sistema a supporto del processo continuo di miglioramento nel modo in cui un'organizzazione utilizza le proprie fonti di energia (sia essa elettrica o di qualsiasi altro genere) [33]. Lo scopo di un sistema EMS è produrre business value considerando il fattore energetico come un importante input nei processi aziendali e costruendo un processo duraturo per monitorarne e pianificarne l'uso.
Un sistema di gestione efficiente fornisce un insieme di best-practice per permettere ad un'organizzazione di raggiungere i propri obiettivi in termini di risparmio nei consumi energetici.

Oggigiorno con le leggi sempre più stringenti in termini di controllo dei consumi energetici e delle emissioni e gli incentivi statali ed europei per le aziende rispettose dei regolamenti, l'utilizzo di sistemi di gestione energetica è diventato fondamentale anche in termini economici. L'International Organization for Standardization (ISO) ha stabilito un insieme di requisiti per implementare, mantenere e migliorare l'uso di sistemi EMS (ISO 50001) con lo scopo di fornire un approccio sistematico nell'ottica del miglioramento continuo delle performance energetiche e riduzione dei consumi. I benefici di questo approccio sono:

- Riduzione dei costi.

- Rispetto delle emissioni imposte dalla legge o dall'organizzazione stessa.
- Aumento della reputazione sociale dell'organizzazione.
- Riduzione dell'impatto ambientale.

1.2.1 Plan-Do-Check-Act

Il processo di costruzione di un sistema EMS definito dallo standard ISO 50001 segue un approccio basato su quattro attività principali: pianificazione, implementazione di ciò che è stato pianificato, verifica del rispetto degli obiettivi e attuazione di azioni correttive in caso di scostamenti (Plan-Do-Check-Act).

Come si può notare da figura 1.2 il processo definito dall'ISO è circolare e mira ad evidenziare un percorso di miglioramento continuo per l'organizzazione, che non può fermarsi una volta raggiunti gli obbiettivi prefissati, ma deve procedere controllando che i risultati raggiunti siano mantenuti, e applicando misure correttive in caso di "uscita dai binari".

Fase di Plan

L'attività di *planning* mira a identificare e formalizzare gli obiettivi che si pone l'organizzazione in ambito energetico e i vincoli esistenti. È importante che a questa attività partecipino i rappresentanti dell'organizzazione stessa (che hanno potere decisionale) e che siano definiti obiettivi concreti e raggiungibili.

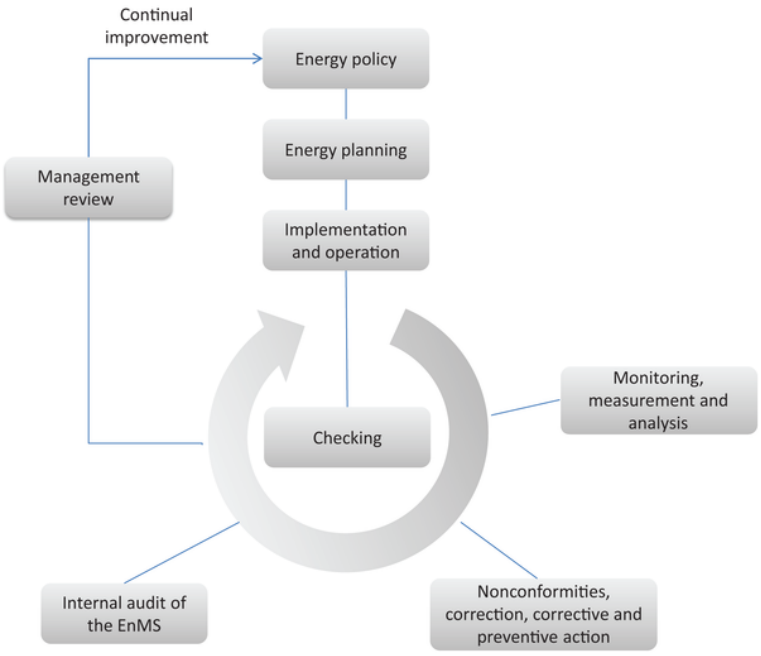


Figura 1.2: Attività principali per un sistema di gestione energetica [29]

In questo stadio vanno quindi definite le policy, ovvero degli obiettivi che siano legati al tempo e quantificabili in modo che sia chiaro quando questi siano raggiunti. Quando si procede a definire le policy è importante considerare i processi e le peculiarità dell'organizzazione in modo che queste non vadano contro il business stesso. Successivamente verrà formato un team con le competenze necessarie nella gestione dell'energia. Il team si occupperà di informare l'intero staff sulle decisioni prese e di implementare il sistema per il raggiungimento degli obiettivi.

Fase di Do

Una volta concluse le operazioni di pianificazione si può procedere con l'*implementazione* vera e propria del sistema. In questa fase si procede identificando tutti quei processi aziendali considerati "energivori" e sviluppando una serie di interventi atti a limitarne i consumi. È importante che tutte le persone interessate comprendano l'importanza del cambiamento e ricevano un'adeguata formazione per poter essere in grado di svolgere i propri compiti in accordo con gli obiettivi energetici fissati.
Inoltre è a questo punto che vengono assegnate le risorse al progetto sia in termini di budget sia in termini di personale.
Durante la fase di implementazione del sistema è fondamentale anche la comunicazione con i fornitori delle fonti energetiche considerate. È importante che questi capiscano il nuovo metro di valutazione aziendale e che forniscano il supporto richiesto per il controllo continuo dei consumi.

Fase di Check

La fase di *check* è tra le più importanti dell'intero processo in quanto fulcro del processo di miglioramento continuo. Il processo di controllo deve essere pervasivo in tutte le fasi in modo da poter fornire un efficace metodo di confronto e determinare le performance del sistema. Comunicazioni frequenti e ad alto contenuto informativo sono la chiave per il successo di un sistema EMS [33].
L'organizzazione deve regolarmente tenere traccia delle proprie performance energetiche valutando come il sistema implementato abbia aiutato a rag-

giungere gli obiettivi.
L'uso di *Key Performance Indicator (KPI)* è molto efficace per questo scopo e aiuta ad individuare potenziali problemi da dover affrontare.
É importante che tutte queste informazioni siano documentate e che sia mantenuto uno storico per successive valutazioni.

Fase di Act

Il concetto di miglioramento continuo prevede l'esecuzione di attivià correttive utili per migliorare le prestazione dell'organizzazione.
L'ultima fase definita dall'ISO prevede la modifica del sistema EMS in caso in cui non si stiano raggiungendo gli obiettivi prefissati o in caso in cui vi siano possibilità di miglioramenti.
La fase di *check* si sposa perfettamente con la teoria dei sistemi di controllo automatici SCADA sopra presentata. È infatti necessario un supporto per la raccolta e l'analisi dei dati relativi alle prestazioni energetiche dell'azienda, inoltre con l'aggiunta di un sistema di allarmi si potrebbe subito essere informati di anomalie nei consumi e si potrebbero individuare le cause.

1.2.2 SCADA Energy Management System

I sistemi SCADA denominati Energy Management sono una specializzazione dei sistemi di controllo SCADA e come questi ultimi svolgono il triplice compito di controllo, monitoraggio ed acquisizione dei dati, ma si concentrano sulle informazioni riguardanti i consumi o gli sprechi di energia (che può essere elettrica come di qualsiasi altro genere) sia a livello domestico

che industriale. L'obiettivo principale che porta all'utilizzo di questi sistemi è la verifica dei consumi e l'individuazione delle perdite o malfunzionamenti negli impianti allo scopo di intervenire per limitare lo spreco di risorse e limitare i costi. Nell'ottica del miglioramento continuo questo è un requisito fondamentale da raggiungere.

I sistemi EMS sono ampiamente utilizzati anche nel settore di produzione e distribuzione di energia, in questo caso le necessità sono leggermente diverse, infatti non risulta molto importante l'analisi dei consumi in se, ma è fondamentale capire l'efficienza degli impianti verificando quanto la produzione attuale si distacchi da quella teorica, considerando le condizioni in cui essi si trovano ad operare. Ad esempio, per analizzare l'efficienza di un impianto fotovoltaico è importante considerare le condizioni meteo, la velocità del vento, la temperatura, la posizione dei pannelli, ecc. E considerate tutte queste informazioni si potrà procedere al monitoraggio del sistema per capire se è necessaria una manutenzione o un intervento di qualsiasi altro genere.

In generale ciò che maggiormente differenzia un sistema EMS da un sistema SCADA generico è la parte software di elaborazione, monitoraggio e controllo (le funzionalità di raccolta, comunicazione e memorizzazione dei dati rimangono invariate), questo perchè l'ambito in cui essi sono utilizzati ha esigenze particolari e si ha più la necessità di un confronto di informazioni che di una semplice analisi statistica, è perciò necessario utilizzare dei report che evidenzino le differenze tra due serie di dati (ad esempio grafici). Le principali funzionalità richieste ad un sistema di gestione energetico pos-

sono essere riassunte nei seguenti punti:

- calcolo ed elaborazione di dati per evidenziare i punti dell'impianto a maggior impatto energetico, in modo da attirare l'attenzione e concentrare gli sforzi.
- visualizzazione di informazioni di confronto e di serie storiche per analizzare l'andamento del sistema considerato.
- calcolo di KPI (Key Performance Indicator) che sono ottimi indicatori riassuntivi sulle prestazioni.

Capitolo 2

Analisi di regressione

Per individuare le aree e le possibiltà d'azione a livello energetico è necessario innanzitutto conoscere i consumi attesi per poi eseguire un confronto con i dati reali e capire quanto questi si discostino dalle previsioni (o dati ideali). Per calcolare il valore atteso di un consumo o di una qualsiasi altra variabile quantitativa è necessario costruire un modello matematico che la rappresenti in relazione ad altre variabili note del sistema. Trovato il modello più rappresentativo si possono applicare varie tecniche di predizione per determinare i parametri e quindi calcolare i valori attesi della variabile desiderata.

Uno dei metodi più comuni per il calcolo dei parametri di un modello è l'analisi di regressione. La regressione utilizza un insieme di osservazioni storiche fatte sul sistema in oggetto per costruire un modello di una variabile e permetterne una stima teorica.

Ovviamente la precisione con cui si può prevedere l'andamento di una variabile dipende da molti fattori, dall'accuratezza con cui è stato costruito il

modello alla lunghezza del periodo da considerare fino alla precisione con cui vengono misurati gli input al modello.

Nelle sezioni successive verranno spiegati gli aspetti più importanti riguardanti l'analisi di regressione e presentati i principali metodi di calcolo.

2.1 Aspetti generali dell'analisi di regressione

L'analisi di regressione è una tecnica usata per analizzare l'andamento di una variabile dipendente (y) in funzione di n variabili indipendenti (x_1, x_2,...x_n) a meno di un errore ϵ. È possibile utilizzare questa tecnica anche per prevedere il valore della variabile dipendente; in questo caso le incognite saranno i coefficienti del modello ed ϵ rappresenterà il valore residuo, cioè la differenza tra il valore misurato e quello previsto di y.
Il modello risultante dall'analisi sarà del tipo $y = ax_1 + bx_2 + c + \epsilon$, dove y è la variabile dipendente, x_1 e x_2 sono variabili indipendenti (o regressori), c rappresenta il termine noto costante e con ϵ si indica l'errore del modello.

Esistono due tipologie di regressione:

- lineare: la variabile dipendente viene calcolata in funzione di un modello lineare. La linearità e da considerarsi tra la variabile dipendente ed i coefficienti del modello, tra la prima e le variabili indipendenti può esservi anche una relazione non lineare (ad esempio la funzione $y = ax_1^2 + bx_1^3 + cx_2 + dx_2^4 + k + \epsilon$ è considerato un modello di re-

gressione lineare). Modelli di questo tipo producono una retta, un piano o un iper–piano (a seconda del numero di parametri in gioco) che approssimerà l'andamento della variabile dipendente.

- non lineare: la variabile dipendente viene calcolata in funzione di un modello non lineare. Spesso per risolvere questi modelli ci si serve di tecniche di linearizzazione per riportarsi al caso precedente. Questo tipo di regressione comunque richiede tecniche di calcolo avanzate che verranno brevemente trattate in seguito.

2.1.1 Bontà del modello

Una volta trovato e calcolato il modello è necessario verificarne la bontà, ovvero stabilire quanto il modello è rappresentativo dei dati. I controlli di bontà comunemente usati sono due:

Valutazione dell'indice R^2

R^2 o coefficiente di determinazione è un indice di valore compreso tra 0 e 1. Più questo è vicino ad 1 più il modello lineare può essere considerato buono. L'indice R^2 misura fino a che punto un modello lineare consente di approssimare la realtà dei dati osservati e solitamente viene calcolato come il rapporto tra le varianze dei valori previsti e dei valori osservati $\frac{\sum_{i=0}^{n}(\hat{y}_i-\bar{y})^2}{\sum_{i=0}^{n}(y_i-\bar{y})^2}$, dove $\hat{y}$ rappresenta i valori di y stimati e $\bar{y}$ le relative medie.
In realtà spesso viene utilizzato un secondo metodo equivalente per il calcolo del coefficiente di determinazione definito come $1 - \frac{\sum_{i=0}^{n}(y_i-\hat{y})^2}{\sum_{i=0}^{n}(y_i-\bar{y})^2}$, questo può assumere valori negativi se il rapporto risulta maggiore di 1, in questo caso

non deve essere interpretato come indicatore di bontà del modello ma come segnale della mancanza del termine noto (termine costante del modello), è quindi opportuno ricostruire il modello tenendo conto di questa informazione, inserendo quindi un coefficiente non legato alle variabili indipendenti (in questo caso anche il termine noto costante rappresenta un'incognita del modello da calcolare).

Il problema di R^2 come indicatore di bontà di un modello è che questo può essere aumentato artificialmente aggiungendo regressori al sistema, ma ciò non significa che il modello sia più rappresentativo. Per ovviare a questo problema si utilizza un R^2 corretto:
$\bar{R^2} = 1 - \frac{N-1}{N-k-1}(1-R^2)$, dove N sono il numero di osservazioni e k il numero di regressori.
È evidente che $\bar{R^2}$ decresce all'aumentare dei regressori mentre aumenta solo nel caso in cui vengano aggiunte variabili che migliorino il modello (aumenta R^2 di un valore significativo).

Analisi dei residui

Un secondo metodo per verificare la bontà di un modello di regressione consiste nel calcolare e analizzare le differenze tra i valori previsti e quelli misurati (residui) per capire quanto il modello si distacchi dai dati reali. Uno dei metodi di analisi dei residui più noti è costituito dalle carte di controllo CUSUM, queste eseguono una somma cumulata (cumulantive-sum) dei residui cercando di capire quando l'errore commesso rischia di non essere più sotto controllo.

CUSUM - In generale le carte di controllo CUSUM si basano sulla rappresentazione grafica delle somme cumulate degli scarti di un'osservazione da un opportuno valore di riferimento (solitamente la media) [38]. Se si considerano i residui come osservazioni da sottoporre all'analisi CUSUM e il valore 0 come riferimento si otterrà una perfetta rappresentazione dell'andamento dei residui: $C_n = \sum_{i=0}^{n}(y_i - \hat{y}_i)$ dove y_i sono i valori osservati e $\hat{y}_i$ quelli previsti e $C_{-1} = 0$. Una buona stima dovrebbe avere i valori calcolati con la formula appena definita che si aggirano attorno allo 0.
Come detto lo scopo di questo tipo di analisi è la verifica che gli scarti dal valore di riferimento rimangano sotto controllo, solitamente vengono quindi definiti due livelli di soglia calcolati come multipli dello scarto quadratico medio delle osservazioni (un valore di $\pm 5\sigma$ è considerato un buon livello di soglia) che fungono da limite superiore ed inferiore per le somme cumulate. Se i valori di C_n superano le soglie si dice che il sistema è fuori controllo ed è necessario intervenire per ripristinare la situazione.

2.1.2 Scelta delle variabili indipendenti

La scelta dei regressori utilizzati per costruire il modello non può essere casuale, ma deve essere fatta in modo che questi siano in grado di spiegare l'andamento della variabile dipendente. Per questo è necessaria un'analisi preventiva che indichi il livello di legame tra le coppie (x_n, y), dove x_n rappresenta una variabile indipendente scelta e y la variabile dipendente.
Il metodo in letteratura più utilizzato per verificare la relazione tra due variabili viene detto *correlazione*. In statistica per correlazione si intende una

relazione tra due variabili statistiche tale che a ciascun valore della prima corrisponda con una "certa regolarità" un valore della seconda [7] (ad esempio è possibile determinare la relazione tra il prezzo e la domanda di una merce).

La correlazione si dice diretta o positiva quando al variare di una variabile anche l'altra varia nello stesso modo (aumento di valori della prima si traducono in un aumento anche nella seconda e viceversa); mentre si dice inversa, indiretta o negativa se al variare in un senso della prima variabile l'altra varia in senso opposto (i valori di una variabile aumentano mentre i valori dell'altra diminuiscono).

Il grado di correlazione fra due variabili viene espresso mediante i cosiddetti indici di correlazione. Esistono vari tipi di indici di correlazione; la scelta di un indice in particolare dipende in generale da vari fattori: il tipo di livello di misurazione di ciascuna delle due variabili; la natura della distribuzione sottostante (continua o discreta); le caratteristiche della distribuzione (lineare o non lineare) [35].

Indipendentemente dal tipo di indice di correlazione che si intende usare, tutti gli indici presentano alcune caratteristiche comuni:

- le due variabili devono essere associate agli stessi eventi o individui;
- i valori assunti dagli indici sono compresi tra -1 e +1. Entrambi i valori rappresentano una correlazione perfetta, mentre 0 rappresenta l'assenza di relazione;
- un valore positivo dell'indice indica una relazione diretta;
- un valore negativo dell'indice indica una relazione inversa;

- sono indici adimensionali, quindi non dipendono dall'unità di misura delle variabili

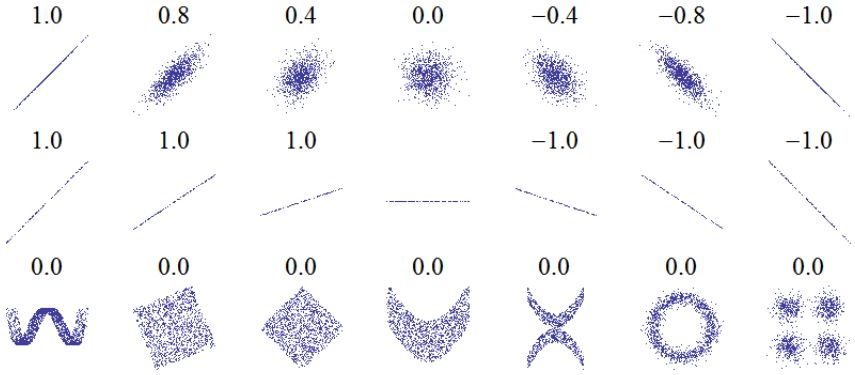


Figura 2.1: Esempi di correlazioni [7]

Una rappresentazione grafica come in figura 2.1 può aiutare a capire meglio come funziona un indice di correlazione. Si può notare come l'indice di correlazione diminuisce all'aumentare della dispersione dei dati, mentre assume valori sempre più vicini ad 1 (o -1) più l'andamento delle variabili è approssimabile ad una retta.

Gli indici in letteratura più noti sono due: l'indice di Pearson e l'indice di Spearman. In seguito verranno brevemente introdotti definendone le modalità di calcolo.

Indice di correlazione di Pearson – esprime la relazione lineare presente tra due variabili. Questo indice non è in grado di individuare relazioni diverse da quella lineare, quindi il suo utilizzo è molto limitato.
La correlazione di Pearson viene calcolata come il rapporto tra la covarianza

delle due variabili e il prodotto tra le loro deviazioni standard: $\frac{\sigma_{xy}}{\sigma_x \sigma_y}$, dove $\sigma_{xy} = \sum_{i=0}^{n} \frac{(x_i - \bar{x})(y_i - \bar{y})}{n}$ e $\sigma_x = \sqrt{\frac{\sum_{i=0}^{n}(x_i - \bar{x})}{n}}$

Coefficiente di correlazione di Spearman – è una specializzazione del coefficiente di Pearson, ma diversamente da quest'ultimo, il coefficiente di Spearman, non misura solamente una relazione lineare ed è in grado di catturare una qualsiasi relazione tra due variabili.

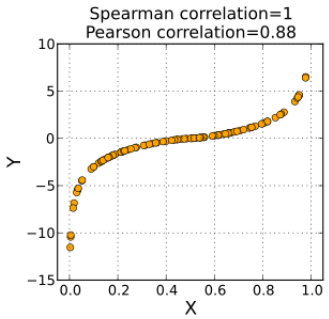


Figura 2.2: Differenza tra i due coefficienti [6]

Come si può vedere da figura 2.2 il coefficiente di Spearman si comporta molto meglio in casi in cui vi è un evidente correlazione tra due variabili anche se non lineare.

La procedura per il calcolo del coefficiente di Spearman è identica a quella definita da Pearson a meno del fatto che al posto dei valori delle variabili vengono utilizzati i ranghi (ovvero un indicatore ordinale dei valori, si veda figura 2.3).

Data 1	Data 2	Rank 1	Rank 2
6	2	2	1
4	9	1	3
7	3	3	2

Figura 2.3: Esempio di ranking

La procedura consiste in:

- Ordinare i valori dal più piccolo al più grande.
- Calcolare i ranghi. In caso di due rilevazioni con lo stesso valore utilizzare la media dei ranghi.
- Utilizzare la formula di Pearson sui ranghi per calcolare la correlazione.

2.1.3 Tipologia di relazione tra due variabili

Un'altra importante considerazione, da fare prima di procedere al calcolo del modello per la stima di una variabile, riguarda la tipologia di relazione presente tra la variabile dipendente e ogni singolo regressore. È infatti importante capire come una variabile indipendente debba essere inserita nel modello (se utilizzata in un polinomio di grado 1, 2 oppure in un'altra tipologia di funzione), questo perchè non sempre, tra ogni coppia di variabili (dipendente, indipendente), vi è una relazione lineare.
É comunque da tener presente che è sempre possibile approssimare la relazione tra due variabili con un polinomio di grado n (infatti anche funzioni complesse possono essere approssimate in questo modo), quindi ci si con-

centrerà su relazioni polinomiali.

L'analisi di regressione ci viene incontro anche nella risoluzione del problema di ricerca della relazione tra due variabili. Da studi effettuati sui modelli ed il coefficiente di determinazione si è notato che aumentando il grado del polinomio che approssima la relazione tra due variabili, e calcolando il modello di regressione su questo polinomio, il coefficiente di determinazione arriverà a toccare il suo massimo per poi diminuire. È quindi possibile costruire un algoritmo che permetta di estrapolare le informazioni necessarie all'individuazione del miglior polinomio che approssima la relazione tra due variabili, basterà trovare quel polinomio che massimizza il valore del coefficiente di determinazione.

Per meglio capire verrà fatto un esempio:
consideriamo due variabili x e y con x indipendente e y dipendente. Possiamo costruire più modelli di regressione con grado di polinomio crescente e calcolare R^2

1. $y = ax + b => R^2 = 0.7$
2. $y = ax^2 + bx + c => R^2 = 0.8$
3. $y = ax^3 + bx^2 + cx + d => R^2 = 0.75$

come si può vedere al secondo passo si è trovato il valore massimo di R^2 e aumentando ulteriormente il grado del polinomio questo è diminuito, quindi si può dire che il miglior polinomio che relaziona le variabili x e y è di secondo grado. È possibile utilizzare questa procedurà per ogni coppia di variabili

(dipendente, indipendente) in modo da trovare un insieme di polinomi che poi potranno essere utilizzati nel modello di regressione completo; ad esempio se ho due variabili indipendenti x_1 e x_2, e trovo che per la prima il miglior polinomio è di 3°grado mentre per la seconda è di 2°grado, il modello di regressione dipendente da x_1 e x_2 sarà del tipo: $ax_1^3+bx_1^2+cx_1+dx_2^2+ex_2+f$, ovvero la combinazione lineare dei due polinomi trovati.

2.2 Metodi di risoluzione per la regressione lineare

I metodi più utilizzati per la stima dei parametri nell'analisi di regressione sono di tre tipologie: Least Square Error, Least Absolute Error e RanSaC; tutti cercano di trovare una soluzione minimizzando l'errore tra quanto osservato e previsto. Ovvero nel caso bidimensionale l'errore da minimizzare può essere rappresentato dalla distanza tra i punti osservati e la retta di regressione.

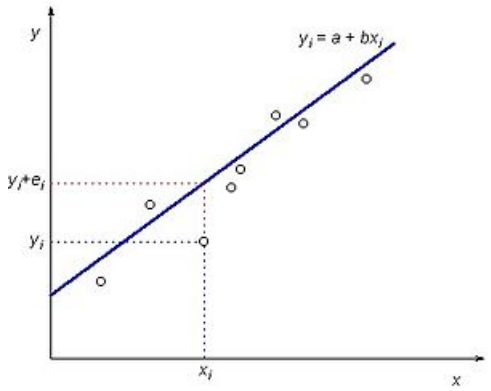


Figura 2.4: Regressione lineare a 2 dimensioni

Figura 2.4 mostra un tipico esempio in cui il modello è costituito dalla retta $y = ax + b$. Come si può notare il valore y_i osservato si distacca dalla retta modellata di un errore e_i rappresentato dalla lunghezza del segmento parallelo all'asse delle ordinate che congiunge il punto y_i con la retta.

2.2.1 Least Square Error

Il metodo dei minimi quadrati (Least Square Method) è il più utilizzato e semplice nell'analisi di regressione lineare, questo considera la somma dei quadrati dei residui $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ come quantità da minimizzare, dove $\hat{y}_i$ rappresenta i valori di y_i previsti. Nel caso generale di più variabili indipendenti questo si traduce in una relazione del tipo $y = a_1x_1 + a_2x_2 + ... + a_nx_n + b$ (quindi nel minimizzare $S = \sum_{i=1}^{n}(y_i - (a_1x_{1i} + a_2x_{2i} + ... + a_nx_{ni} + b))^2$) e la soluzione consiste nel trovare i coefficienti $a_1, a_2, ..., a_n$ e b in modo da esprimere al meglio la relazione tra le x e y.
La scelta dei coefficienti nel caso dei minimi quadrati deve sottostare ad un insieme di ipotesi [20]:

- il valore atteso $\mathrm{E}(\epsilon_i) = 0$;
- lo scarto quadratico $\sigma^2(\epsilon_i)$ è costante;
- la covarianza $\sigma(\epsilon_i, \epsilon_j) = 0, \forall i \neq j$, cioè non vi è legame tra i residui.

Sotto queste ipotesi è possibile trovare il valore dei coefficienti del piano di regressione mettendo a sistema le derivate parziali di S rispetto ai coefficienti e tradurre il tutto in termini matriciali in cui compaiono solamente le variabili indipendenti e la variabile dipendente (i cui valori sono noti da osservazioni).
Il vettore $\boldsymbol{\beta}$ dei coefficienti si traduce perciò in:
$\boldsymbol{\beta} = [X^T X]^{-1} X^T Y$, dove X è la matrice dei valori dei regressori (o variabili dipendenti, questa conterrà una colonna con tutti valori ad uno se nella funzione di regressione è presente un termine noto) e Y è il vettore dei valori della variabile dipendente.

Problemi nel metodo dei minimi quadrati

Vi possono essere casi in cui nonostante siano rispettate tutte le ipotesi sopra presentate il calcolo presenti risultati non corretti [20]:

- multicollinearità: se due o più variabili indipendenti sono tra loro linearmente dipendenti non esiste la matrice inversa $(X^T X)^{-1}$. Questo è molto raro per osservazioni reali, ma può comunque succedere che due variabili siano molto vicine ad esserlo, ciò non pregiudicherebbe il calcolo dei parametri ma diminuirebbe la bontà del modello.

Per evitare questa situazione è necessario individuare ed eliminare le dipendenze tra i regressori, calcolando prima il coefficiente di determinazione (tra ogni coppia di variabili indipendenti) per poi eliminare quei regressori con valori di R^2 troppo vicini ad 1.

- outliers: se tra i dati osservati vi è una forte presenza di outliers (valori anomali) il modello potrebbe risentirne e non rappresentare al meglio l'andamento della variabile dipendente. Per attenuare questo problema si possono prendere due strade:

  1. individuare ed eliminare gli outliers dalle osservazioni. In questo caso è necessario scegliere con cura il valore di soglia per evitare di escludere dati validi.

  2. utilizzare un modello pesato. Si può pesare l'impatto di ciascuna variabile a seconda della sua vicinanza con il modello lineare trovato assegnando un peso maggiore ai dati più vicini, oppure calcolando lo scarto quadratico medio di ogni dato osservato e considerarne l'inverso come peso (i dati più distanti dalla media avranno peso minore).
     Il calcolo dei coefficienti si trasformerebbe così in:
     $\beta = [X^T W X]^{-1} X^T W Y$ con W matrice quadrata diagonale dei pesi (assume valori solo in corrispondenza dell'osservazione considerata).

2.2.2 Least Absolute Value

Un secondo metodo di risoluzione può prevedere la minimizzazione del valore assoluto dei residui. Questo porta ad una serie di vantaggi rispetto al least square che spesso lo rendono preferibile:

- il modello è meno sensibile agli outliers. Utilizzando il valore assoluto al posto del quadrato degli errori le distanze più elevate dal piano di regressione sono meno rilevanti.

- la distribuzione dell'errore non deve rispettare delle condizioni. Perciò non è necessario formulare ipotesi sul valore atteso e sullo scarto quadratico medio dell'errore.

- è utilizzabile anche con un numero basso di osservazioni disponibili.

Di contro il tempo di esecuzione dell'analisi di regressione utilizzando questo metodo è molto più alto rispetto ai minimi quadrati ed è anche per questo che tuttora risulta poco utilizzato.

In questo caso non è possibile procedere come nel caso precedente con le derivate parziali in quanto la funzione valore assoluto non è derivabile su tutto il dominio.
È quindi necessario utilizzare metodi che permettano di ottenere il risultato senza utilizzare direttamente le derivate parziali sulla funzione di valore assoluto.

Soliman - Christensen Algorithm

L'algoritmo di Soliman e Christensen utilizza i minimi quadrati per la risoluzione del Least Absolute.
È un algoritmo che prevede quattro passi principali [26]:

1. Calcolo del modello attraverso i minimi quadrati.

2. Calcolo dello scarto quadratico medio dei residui utilizzando la formula $\sigma = \sum_{i=1}^{n} \sqrt{\frac{(\hat{y}_i + y_{av})^2}{n-1}}$, dove $\hat{y}_i$ rappresenta l'errore commesso per l'osservazione i-esima, y_{av} è la media degli errori e n è il numero di osservazioni.

3. Scarto degli outliers. Si scartano le osservazioni dai regressori e dalla variabile dipendente per cui i residui superano il valore dello scarto quadratico medio precedentemente calcolato.

4. Si ricalcola il modello con il metodo dei minimi quadrati utilizzando le osservazioni rimanenti.

5. Si ordinano le osservazioni rimanenti in ordine di residui (cioè dalle osservazioni più vicine al modello alle più lontane) e si selezionano le prime k osservazioni ottenendo cosi il vettore $\boldsymbol{Y'}$ contenente i valori della variabile dipendente e la matrice $\boldsymbol{H'}$ contenente i valori delle variabili indipendenti precedentemente selezionati. Il parametro k dipende dal numero di coefficienti da trovare e rappresenta il numero minimo di osservazioni necessarie per risolvere il modello.

6. Si trovano i parametri del modello utilizzando $\boldsymbol{\beta} = \boldsymbol{H'^{-1}Z'}$

Il problema del metodo di Soliman - Christensen sta nell'utilizzo della matrice inversa $\boldsymbol{H'^{-1}}$, infatti questo non sempre esiste e potrebbe portare a nessuna soluzione.

Linear Programming Algorithm

Si cerca di minimizzare la funzione costo $\sum_{i=1}^{n} |y_i - a_0 - a_1 x_{i1} - a_2 x_{i2} ...|$. Si può riscrivere il problema come la minimizzazione di $\sum_{i=1}^{n} u_i$
soggetta a:
$u_i \geq y_i - a_0 - a_1 x_{i1} - a_2 x_{i2} ...$ e
$u_i \geq -(y_i - a_0 - a_1 x_{i1} - a_2 x_{i2} ...)$
I vincoli hanno l'effetto di eguagliare u_i a $|y_i - a_0 - a_1 x_{i1} - a_2 x_{i2} ...|$, senza utilizzare il valore assoluto. Perciò la funzione u_i può essere minimizzata utilizzando una qualsiasi tecnica per la programmazione lineare.

2.2.3 RANdom Sample Consensus

RANSAC è un metodo iterativo per la stima dei parametri di un modello che a partire da un insieme di dati contenente outliers (anche molto numerosi), ritorna un risultato corretto con una probabilità che aumenta con il crescere delle iterazioni [41].
RANSAC genera una soluzione candidata usando il minimo numero di osservazioni richieste per stimare i parametri del modello e ad ogni iterazione allarga questa soluzione inserendo un insieme di dati coerenti con il modello stimato (inliers).

I passi dell'algoritmo sono i seguenti:

1. Selezione randomica dei punti appartenenti alla soluzione iniziale (consensus set). Il numero delle osservazioni scelte deve essere il minimo richiesto per determinare i parametri del modello (ad esempio un modello del tipo $y = ax + b$ richiede almeno due osservazioni per stimare i parametri a e b).

2. Risolvere il modello determinando i parametri, utilizzando ad esempio i minimi quadrati.

3. Trovare tutti i punti appartenenti al data–set conformi al modello trovato, considerando un errore massimo e. In pratica per ogni punto viene calcolata la distanza dal modello e se questa è minore di una certa soglia (input dell'algoritmo) il punto viene considerato inlier ed inserito nel consensus set, altrimenti scartato come outlier.

4. Se il numero di inliers totale appartenenti al consensus set supera un certo valore e l'errore totale (ad esempio somma della distanza degli outliers) è al di sotto di una certa soglia, tale modello viene considerato buono come approssimazione dei dati iniziali.

5. Il modello viene ricalcolato con i dati aggiunti al consensus set.

6. I passi vengono ripetuti fino al raggiungimento del numero di iterazioni scelto e viene restituito il modello migliore se trovato, altrimenti se nessun modello ha superato i parametri richiesti non viene restituito nulla.

Un esempio di una possibile implementazione dell'algoritmo RANSAC in pseudo–codice [19]:

```
iterazioni := 0
miglior_modello := nil
miglior_consensus_set := nil
miglior_errore := infinito
while (iterazioni < k) {

    possibili_inlier := n valori scelti casualmente dai dati

    possibile_modello := parametri del modello stimati su possibili_inlier

    consensus_set := possibili_inlier

    for (ogni punto di dati not in possibili_inlier){
        if (punto appartiene a possibile_modello con un errore minore di t) {

            aggiungi punto a consensus_set
        }
    }

    if (numero di elementi in consensus_set > d){
        /*(implica che abbiamo trovato un buon modello, testiamo ora quanto e' buono)*/

        modello_migliorato := parametri del modello stimato su tutti i punti di consensus_set

        errore_attuale :=misura di quanto bene modello_migliorato si adatta ai punti

        if (errore_attuale < miglior_errore){
        /*(abbiamo trovato un modello migliore dei precedenti, salviamolo fino a che non ne
        viene trovato uno ancora piu' buono)*/

            miglior_modello := modello_migliorato
            miglior_consensus_set := consensus_set
            miglior_errore := errore attuale
        }
    }
    incrementa iterazioni
}
return miglior_modello, miglior_consensus_set, miglior_errore
```

input dell'algoritmo:

```
dati - un insieme di osservazioni.

modello - un modello che puo' essere assegnato ai dati.

n - il numero minimo di dati per cui si richiede la appartenenza al modello.

k - il numero di iterazioni eseguite dall'algoritmo.

t - un valore di soglia per determinare quando un dato appartiene al modello.

d - il numero di dati ``vicini'' richiesti per asserire che un modello rappresenta bene i dati.
```

output dell'algoritmo:

```
miglior_modello - parametri del modello che meglio rappresenta i dati
(o nil se nessun modello sufficientemente buono viene trovato).

miglior_consensus_set - dati dai quali il modello e' stato stimato.

miglior_errore - l'errore del modello stimato relativo ai dati.
```

La scelta dei valori dei parametri deve essere attenta e pertinente. I parametri t (soglia per determinare se un dato appartiene al modello – soglia di inlier) e d (numero minimo di inliers per poter definire buono un modello) devono essere relativi all'applicazione e ai dati e dipendono dal contesto in cui ci si trova a costruire il modello ed eseguire l'analisi di regressione. La sovrastima o la sottostima di questi parametri può portare a risultati falsati ed a conclusioni sbagliate. Dalla stima sbagliata della soglia t ad esempio ne conseguirebbe l'inclusione nel modello di outliers oppure l'esclusione di dati significativi, mentre l'errata valutazione di d porterebbe a scartare modelli considerati buoni o viceversa ad includerne di sbagliati.
La soglia di inclusione/esclusione dei dati nel modello può essere calcolata attraverso la distribuzione chi-quadro:

$$\delta = \sigma_\eta \sqrt{F_x^{-1}(P_{inlier})}$$

dove F^{-1} è la funzione di distribuzione cumulativa inversa e σ_η è lo scarto quadratico medio calcolato per le differenze tra i dati osservati e quelli stimati (η). Più è alta la probabilità più sarà grande δ e quindi si avranno più dati appartenenti all'insieme degli inlier e una maggiore probabilità di errore in tal senso (la probabilità P_{inlier} deve essere scelta all'inizio del processo). Oppure può essere scelta una soglia più semplice da calcolare come il solo scarto quadratico medio dell'errore.

Il parametro k (numero di iterazioni) può essere calcolato tramite un'assunzione: consideriamo p, la probabilità che almeno uno dei consensus set scelto casualmente non contenga outlier (solitamente è settata 0.99) e u la probabilità che ogni osservazione selezionata (ovvero ogni dato scelto casualmente nel consensus set iniziale) sia un inlier. Quindi $v = 1 - u$ è la probabilità di avere un outlier.
Si ha quindi che $1 - p = (1 - u^m)^k$ con:
m: numero minimo di punti necessari per la stima del modello
k: numero di iterazioni
Da qui si può ricavare $k = \frac{log(1-p)}{(1-u^m)}$.

Vantaggi e svantaggi

VANTAGGI:
Si ottiene una stima dei parametri con un alto grado di accuratezza nonostante la presenza di outliers.
SVANTAGGI:

Se non si indica un numero massimo di iterazioni il tempo richiesto per la stima dei parametri potrebbe essere elevato (l'algoritmo termina solo quando tutte le possibili combinazioni dei dati osservati saranno state estratte). Se si impone un numero di iterazioni questo problema si risolve ma il modello potrebbe non essere ottimale.

2.3 Metodi di risoluzione per la regressione non lineare

Come già detto in precedenza una regressione si definisce non lineare quando non sussiste una relazione di linearità tra la variabile dipendente ed i coefficienti del modello.

La maggior parte dei metodi per risolvere un modello non lineare prevedono di linearizzare il modello per poi applicare uno degli algoritmi precedenti per risolverlo.

2.3.1 Esempio: utilizzo del least square per modelli non lineari

L'utilizzo del metodo least square su modelli non lineari prevede i seguenti passi [26]:

1. eseguire una prima previsione dei coefficienti.

2. calcolare i residui (R) utilizzando i coefficienti trovati.

3. calcolare i delta (cioè la differenza tra nuovi coefficienti da trovare e quelli già calcolati) utilizzando la formula $[H^T W H]^{-1}[H^T W R]$
   dove H è una matrice contenente le derivate parziali di y rispetto ai coefficienti $\frac{\partial y_i}{\partial \beta_j}$ e W è una matrice diagonale contenente i valori $\frac{1}{\beta_j^2}$

4. se i valori calcolati al punto precedente soddisfano i requisiti di soddisfacimento scelti dall'utente all'inizio dell'algoritmo, stop. Altrimenti si prosegue al punto 5.

5. si aggiornano i coefficienti sommandogli i delta trovati al punto 3 e si torna al punto 2.

2.3.2 Segmented regression

La regressione segmentata è un metodo in cui la variabile indipendente viene suddivisa in intervalli o segmenti e su questi intervalli viene eseguita una regressione lineare. È importante la scelta del breakpoint (punto di suddivisione da un segmento ad un altro) che deve essere scelto con cura.

2.4 Librerie .NET a supporto dell'analisi di regressione

In letteratura sono disponibili numerose soluzioni software per il calcolo di modelli di regressione. In questa sezione verranno presentate due librerie .NET che forniscono un insieme di metodi di supporto per l'analisi:

- accord.Net
- math.Net

Entrambe le librerie permettono di trovare un modello che definisce al meglio l'andamento di una o più variabili dipendenti rispetto alle variabili indipendenti. La principale differenza tra le due librerie (per quanto riguarda l'analisi di regressione) è rappresentata dalla possibilità nella seconda di utilizzare il calcolo matriciale, che permette un maggior controllo sull'algoritmo utilizzato anche se potrebbe non essere la scelta ottimale in termini di tempo di calcolo, performance e precisione della soluzione.

In seguito verranno brevemente descritte entrambe le librerie.

2.4.1 Accord.Net

Accord.Net è una libreria software scritta in c# che fornisce supporto per operazioni di machine learning e video/audio e image processing. Nell'ambito del machine learning possiede anche un modulo dedicato alla regressione che permette di automatizzarne i calcoli fornendo in input i valori osservati. Utilizzando questa libreria il calcolo del modello è completamente nascosto allo sviluppatore, che deve semplicemente occuparsi di costruire l'input (variabili indipendenti e variabile dipendente). Il primo passo da eseguire è la chiamata al metodo Regress(double[][] x, double[] y) con cui viene costruito il modello.
Successivamente è possibile recuperare i coefficienti accedendo al campo coefficients dell'oggetto di tipo *MultipleLinearRegression* oppure calcolare

le y previste utilizzando il metodo compute(double[] x), che data in ingresso un'osservazione (valori assunti dalle variabili dipendenti in un certo istante) restituirà il valore della variabile dipendente calcolato applicando il modello trovato. La libreria permette anche il calcolo del coefficiente di determinazione (R^2) con il metodo CoefficientOfDetermination(double[][] x, double[] y) previa costruzione del modello.

```
int n = number_of_independent_variable;
MultipleLinearRegression linearRegression = new MultipleLinearRegression(n);

linearRegression.Regress(H, Y);
//H matrice delle osservazioni delle variabili indipendente
//Y vettore delle osservazione della variabile dipendente

var yi_forcast = linearRegression.Compute(H[i]);

var R2 =  linearRegression.CoefficientOfDetermination(H,Y);
```

Anche l'algoritmo RANSAC è supportato da accord.Net che fornisce una classe che, inizializzata opportunamente, eseguirà tutti i passi previsti dall'algoritmo.

```
int nTries = n_computation_tries;
int n = number_of_independent_variable;

var ransac = new RANSAC<MultipleLinearRegression>(n);
ransac.MaxEvaluations = nTries;

/* la soglia puo essere fissa oppure dinamica cioe ricalcolata ad ogni passo dell'algoritmo */
ransac.Threshold = standardDeviation;

/* Definisco la funzione per il calcolo del modello */
ransac.Fitting = regression_function(H, Y);
//H matrice delle osservazioni delle variabili indipendente
//Y vettore delle osservazione della variabile dipendente

/* Definisco la funzione per separare gli inliers dagli outliers */
ransac.Distances = distance_function(inlier_threshold);


/* Ora che tutti i paramentri sono stati settati si puo
procedere al calcolo del modello. */

int[] inlierIndices;
MultipleLinearRegression regression = ransac.Compute(Y.Count, out inlierIndices);
```

Il metodo è molto simile al caso precedente ma bisogna fare attenzione alla scelta dei parametri.

2.4.2 Math.Net

Math.Net è una libreria che include ogni tipo di funzione matematica ed algebrica, oltre ad un insieme di metodi e funzioni a supporto degli algoritmi più complessi.
Come già detto, anche Math.Net come Accord fornisce un insieme di classi e funzioni per assistere l'analisi di regressione. La parte interessante di questa libreria però è la possibilità di utilizzare direttamente il calcolo matriciale.

Questo metodo di calcolo non è molto performante in quanto è necessario prima costruirsi le matrici e poi eseguire i calcoli, ma permette di ricondursi direttamente alla teoria vista in precedenza. Inoltre utilizzando math.Net si può decidere di calcolare il modello pesando i dati senza dover forzatamente scartare gli outliers, semplicemente costruendo un nuova matrice.

Bisogna però fare attenzione all'utilizzo del calcolo matriciale in quanto non sempre le matrici sono invertibili e si potrebbero avere casi in cui non è possibile ottenere un risultato.

```
var vectorBuilder = Vector<double>.Build;
var matrixBuilder = Matrix<double>.Build;
var HMatrix = matrixBuilder.DenseOfArray(H);
var YVector = vectorBuilder.DenseOfArray(Y);

var parameters = (HMatrix.Transpose() * HMatrix).Inverse() * HMatrix.Transpose() * YVector;
```

Capitolo 3

On.Energy Management System

In questo capitolo verrà presentato il software On.Energy Management System, su cui si basa il caso di studio considerato in questo lavoro di tesi. Verranno presentate le modalità con cui il sistema On.Energy mappa le funzionalità dei sistemi SCADA precedentemente definite ed identificate le peculiarità del sistema stesso; inoltre verranno presentati i benefici ottenibili dall'utilizzo del software in ambito della gestione energetica.

3.1 Introduzione ad On.Energy

On.Energy è un software web-based sviluppato da Onit Group con l'obiettivo di andare incontro alle aziende che desiderano monitorare i propri consumi e iniziare un percorso di risparmio basato sull'utilizzo razionale delle

risorse: il software tende ad evidenziare i punti all'interno di un impianto a maggior costo energetico in modo da poter identificare le criticità ed intervenire per sanarle [39].
On.Energy offre diverse possibilità in base alla tipologia di impianto su cui è utilizzato: può essere installato per la gestione di attività come monitoraggio di produzione di realtà industriali, gestione di edifici residenziali, gestione di impianti a fonti rinnovabili, gestione di impianti di produzione energetica, ecc.

Come per tutti i software web-based di questo genere On.Energy permette l'accesso alle proprie funzionalità solo a seguito di uno step di autenticazione, in questo modo un utente può accedere e modificare solo i dati ad esso associati e di sua pertinenza, proteggendo così la privacy di tutti gli utilizzatori e l'integrità dei dati presenti sull'applicativo.

Il sistema è stato sviluppato su due tecnologie entrambe fruibili da web browser. La prima si basa su piattaforma Silverlight accessibile unicamente da browser desktop e a seguito dell'installazione dell'apposito plug-in fornito da Microsoft. Per questi motivi successivamente è stata sviluppata una seconda versione basata su **HTML5** utilizzabile da qualsiasi dispositivo (anche mobile). Nella seconda versione si è cercato di migliorare il software aggiungendo funzionalità che non erano presenti nella versione precedente, tenendo in considerazione i feedback ricevuti dagli utilizzatori del sistema.

3.2 Architettura On.Energy

L'architettura del sistema On.Energy può essere divisa in tre parti: dispositivi sul campo, Data Retrieval ed il sistema centrale On.Energy.

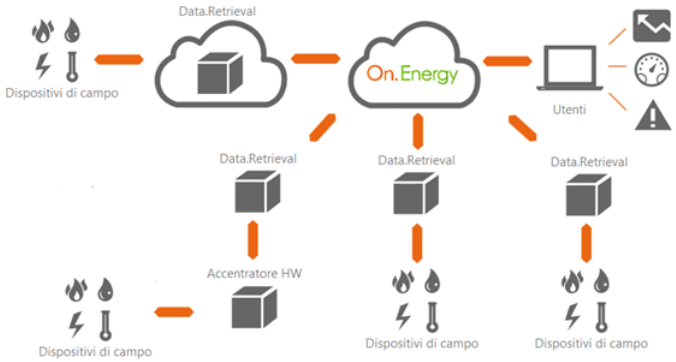


Figura 3.1: Architettura del sistema On.Energy

Come si può notare dalla figura 3.1 le tre parti componenti l'intero sistema possono essere distribuite e comunicano attraverso la rete.
L'architettura è stata progettata per rispecchiare esattamente quanto precedentemente analizzato per i sistemi SCADA (Sezione 1.1.1):

- Dispositivi sul campo: rappresenta la rete di sensori utilizzata da On.Energy per recuperare le informazioni relative agli impianti. Le grandezze gestite da On.Energy riguardano tutto ciò che può essere misurato: l'energia nelle sue diverse forme (elettrica, termica, meccanica, frigorifera, etc.), i fluidi utilizzati nei vari impianti (acqua, gas naturale, liquido refrigerante, vapore, etc.), qualsiasi variabile rilevabile dal campo (temperatura, umidità, pressione, luce, stato antincen-

dio, videosorveglianza, chiusura porte, stato gruppi elettrogeni, stato UPS, caldaie, gruppi frigo, etc.)[39]. In questo livello sono posizionati anche i dispositivi PLC o RTU che possono essere utilizzati come convertitori dei segnali elettrici prodotti dai sensori.

- Data retrieval (accentratore): rappresenta l'interfaccia tra i dispositivi sul campo ed il core del sistema. Esso si occuperà dell'integrazione e pulizia dei dati, ovvero di fornire ad On.Energy dei dati privi di inconsistenze e pronti per essere memorizzati su un database. Inoltre fornisce un sistema affidabile di code (cioè un insieme di buffer su cui mantenere temporaneamente i dati acquisiti e non ancora memorizzati) per permettere la continua operabilità anche in caso di interruzione della connessione con la base di dati.

- Software On.Energy: rappresenta il core dell'intero sistema. Questa riceve i dati dal data retrieval e li elabora e memorizza su una base di dati per renderli disponibili ad analisi real-time o storiche da parte dell'utente.

Questo tipo di architettura permette all'azienda Onit di fornire ai clienti diversi tipi di installazioni che meglio si adattano alle loro esigenze:

- Installazione Cloud: la parte di Data Retrieval e del core On.Energy sono interamente installati sul cloud. I dati forniti dai sensori devono passare dalla rete internet per essere elaborati dall'accentratore ed essere trasferiti al sistema centrale. Il cliente può accedere al servizio sempre passando dal web.

- Installazione Semi-Cloud: la parte di Data Retrieval è presa in carico dal cliente mentre il sistema centrale On.Energy rimane sul cloud. Questo permette al Data Retrieval di comunicare con i sensori anche in assenza di connessione internet e di memorizzare temporaneamente i dati grazie al sistema di code, ma l'accesso e l'elaborazione di queste informazioni resta accessibile solamente tramite web.

- Installazione On-site: l'intero sistema viene preso in carico dal cliente. Il software diventa accessibile esclusivamente dalla rete aziendale del cliente, e non si ha la necessità di dover passare dalla rete pubblica per la comunicazione tra i componenti del sistema.

3.3 Elementi appartenenti ad On.Energy

Dopo aver analizzato la struttura fisica del sistema On.Energy è doveroso ora procedere con un'analisi degli elementi dal software, che consentono ad On.Energy di rappresentare la realtà e permettono all'utente di interfacciarsi con i componenti fisici e le grandezze da essi rappresentate.

L'entità principe del software On.Energy è costituito dagli *stream*. Lo stream rappresenta i valori che ha assunto nel tempo una variabile di un impianto ed è il frutto dello scambio di informazioni tra i sensori, il Data Retrieval ed il software On.Energy. Ad ogni valore assunto da uno stream è associata una data che indica quando il dato è stato rilevato. Inoltre il sistema è in grado di campionare le misurazioni fornendo dati ad intervalli

regolari con cui alimentare uno stream.

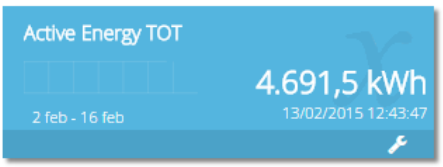


Figura 3.2: Esempio visualizzazione stream

Ad ogni stream sono associate un insieme di informazioni aggiuntive che permettono una descrizione più approfondita di ciò che esso rappresenta nella realtà. Ad ogni stream può essere associato la tipologia di misura (energia, temperatura, luminosità, ecc.), l'unità di misura con cui dovrà essere visualizzato (watt/h, °K, lux), il periodo di campionamento (cioè se la grandezza viene rilevata ad intervalli regolari e quanto spesso) e la tipologia di valore. Le tipologie di stream sono tre ognuna che caratterizza i valori rilevati:

- puntuale: cioè un valore rilevato in un determinato istante di tempo;
- accumulatore: è sempre un valore misurato in un determinato istante, ma che rappresenta una misurazione in continuo aumento ottenuta sommando un valore a quello rilevato nell'istante precedente; ne sono un esempio i contatori di consumo energetici;
- aggregato: rappresenta un valore in un determinato intervallo di tempo dato dall'aggregazione dei valori in quell'intervallo.

Un'altra importante elemento fornito da On.Energy è l'indicatore (o formula periodica). Un indicatore rappresenta un altro flusso di dati, ma

ottenuto per mezzo di calcoli da valori già presenti nel sistema. Si possono eseguire computazioni sia su stream sia su altri indicatori precedentemente creati.

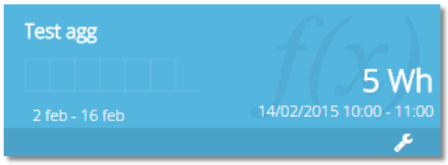


Figura 3.3: Esempio visualizzazione indicatore

L'utilizzo di indicatori è utile in quanto non sempre le informazioni fornite direttamente dagli stream sono utili per un analisi; basti pensare ad uno stream di tipo accumulatore per la misurazione di un consumo; in se questa informazione non è molto utile, mentre assume significato se ad ogni rilevazione viene sottratta quella precedente.
Anche gli indicatori hanno associate le informazioni elencate per gli stream, con in più la possibilità di definite proprie computazioni con cui verranno alimentati.

Ogni entità (stream o indicatore che sia) viene posizionata all'interno di On.Energy in una struttura ad albero. Ogni nodo dell'albero corrisponde ad un elemento della realtà e permette di esprimere un concetto di appartenenza. Ad esempio una gerarchia può essere creata per suddividere i vari reparti di un azienda, oppure suddividere un condominio in piani e appartamenti. Il modo in cui la gerarchia verrà strutturata dipende dall'informazione che

si vuole far emergere.

Figura 3.4: Esempio visualizzazione nodo

Infine ad ogni nodo è possibile assegnare un'ulteriore informazione detta "categoria"; essa permette di individuare intervalli di tempo ed associarvi un nome o etichetta. Ogni etichetta può essere ripetuta più volte per diversi intervalli di tempo; l'unico vincolo presente nella generazione di etichette è rappresentato dal fatto che non è possibile assegnare più etichette appartenenti ad una stessa categoria in uno stesso intervallo di tempo (in pratica le etichette di una stessa categoria non si devono sovrapporre). Un esempio di utilizzo di etichette può essere fornito dalla suddivisione della giornata in giorno/notte oppure di un anno in stagioni.
Grazie a questo concetto è possibile aggiungere un'ulteriore informazione agli stream ed indicatori legati ad un nodo; è infatti possibile aggregare i valori in base alle etichette, oppure suddividerli in base alle fasce temporali definite ed eseguire un'analisi in base a queste (ad esempio consumi gas e luce nei periodi estate/inverno).

3.4 Funzionalità di On.Energy

Come per tutti i sistemi SCADA anche per On.Energy è possibile individuare tre gruppi in cui suddividere le funzionalità fornite dal software:

- Acquisizione dei dati: il software permette il recupero di informazioni direttamente da hardware appositamente configurato. Onit non fornisce direttamente le tecnologie hardware da cui acquisire i dati da analizzare, ma ha stretto particolari partnership con aziende produttrici dei dispositivi utilizzati. È inoltre possibile reperire informazioni da fonti esterne che non prevedono l'interrogazione diretta di hardware.

- Monitoring: una delle caratteristiche fondamentali che un sistema SCADA/EMS deve possedere riguarda la visualizzazione e l'analisi dei dati acquisiti. In On.Energy questa funzionalità è particolarmente evoluta grazie all'ampia disponibilità di strumenti di analisi (quali grafici, tabelle, cruscotti, ecc.) e alla loro forte possibilità di personalizzazione.

- Controllo delle operazioni: lo stesso hardware che permette la lettura dei dati offre anche la possibilità di poterli impostare: anch'essa è disponibile con una grafica ricca e intuitiva.

3.4.1 Acquisizione di dati

Il Data Retrieval è in grado di comunicare con i dispositivi sul campo grazie a protocolli standard e ampiamente diffusi a livello industriale e dei sistemi automatizzati di controllo. Questo permette al software di interfacciarsi

con un gran numero di dispositivi anche se non provenienti direttamente dai produttori di hardware con cui Onit ha stretto legami commerciali. Le principali modalità con cui il software è in grado di acquisire le informazioni da elaborare sono tre:

- lettura di valori da dispositivi sul campo;
- importazione da file;
- inserimento manuale.

Lettura di valori da dispositivi sul campo

Il software è in grado di acquisire dati comunicando direttamente con i dispositivi di controllo PLC o RTU dislocati nei vari impianti, utilizzando i protocolli standard. I più importanti e noti sono:

- Modbus: è un protocollo nato con lo scopo di far interagire diversi controllori logici (PLC). È diventato uno standard de facto nella comunicazione di tipo industriale, ed oggi è uno dei protocolli di connessione più diffusi perchè non impone molte restrizioni riguardo alla trasmissione di dati tra i dispositivi.

  Modbus può essere utilizzato sia su porta seriale che su ethernet permettendo anche l'utilizzo di protocollo TCP/IP per la trasmissione.

  Ad ogni dispositivo che necessita di comunicare per mezzo di Modbus viene assegnato un indirizzo univoco ed ognuno di questi può inviare un comando, sebbene generalmente è solamente uno dei dispositivi ad inviare comandi (master). Ogni comando contiene l'indirizzo di

destinazione e verrà elaborato solo da quest'ultimo (slave).
I comandi possono essere di vario genere, come ad esempio la lettura di un valore salvato sul registro del destinatario (lettura di un valore rilevato da un sensore) oppure la modifica di un valore settato su un registro (comando inviato ad un attuatore).
Solitamente nei sistemi di tipo SCADA il ruolo di master viene assunto dall'unità centrale, mentre i vari dispositivi logici di controllo (PLC o RTU) assumono il ruolo di slave.

Sebbene sia ampiamente utilizzato Modbus non è esente da limitazioni:

  - non è presente una descrizione standard per i valori trasmessi tramite il protocollo;
  - la divisione dei dispositivi in master e slave non permette la trasmissione di un valore solo quando questo è cambiato, dovrà quindi essere il master ad interrogare a polling gli altri dispositivi e verificare i cambiamenti, il che si traduce in uno spreco di banda;
  - non fornisce meccanismi di sicurezza nativi;
  - vi è una limitazione nei dispositivi contemporaneamente presenti su una rete modbus.

- LonWorks: è un protocollo di comunicazione digitale creato per garantire flessibilità e una facile installazione e manutenzione di sistemi di automazione distribuiti, come il controllo dell'illuminazione, del condizionamento che si è diffuso anche a livello industriale (attualmente si

contano diversi milioni di dispositivi che utilizzano questo protocollo di comunicazione).
LonWorks permette di comunicare utilizzando diversi tipi di connessioni tra cui TCP/IP, e a oggi è considerato uno standard internazionale.

Per garantire l'interoperabilità tra dispositivi di diverse case produttrici si è creata un'organizzazione internazionale con il compito di standardizzare i messaggi per rappresentare le varie grandezze trasmissibili (temperature, tensioni, stati, ecc.). Questi messaggi vengono chiamati Standard Network Variable Types (SNVT) e rappresentano il dizionario a cui si devono attenere i dispositivi per comunicare (ad esempio per una temperatura ci si aspetta un numero compreso tra 0 e 65535 che rappresenta una temperatura compresa tra -274 e 6279.5 gradi centigradi, con le prime quattro cifre rappresentano la parte intera della temperatura, con i numeri compresi tra 0 e 273 corrispondenti ai valori negativi, mentre l'ultima cifra corrisponde alla parte decimale, ovviamente in questo modo viene consentita la precisione di un solo decimale).

Le principali caratteristiche del protocollo sono:

- comunicazione paritetica tra i nodi che comunicano con gli altri in modo indipendente;
- possibilità di autenticazione dei messaggi;

  - comunicazioni ad eventi delle informazioni che vengono trasmesse solo a seguito di un cambiamento;
  - indipendenza dal livello fisico di trasmissione (mezzo di trasmissione fisico delle informazioni).

- KNX: è il primo protocollo per il building automation standardizzato a livello mondiale.
  Uno dei punti di forza di KNX è che qualsiasi prodotto etichettato con il suo marchio viene testato dai suoi laboratori che non ne provano solo la possibilità di supporto dello standard, ma anche che i dati utili prodotti dal dispositivo siano conformi agli standard di comunicazione definiti.

  Lo standard è progettato per essere indipendente da qualsiasi piattaforma hardware. Un device KNX può essere controllato da un microcontrollore come da un pc e permette la comunicazione anche attraverso reti LAN (Local Area Network).

  I dispositivi appartenenti ad una rete KNX possono essere divisi in tre categorie, ma queste possono cambiare ed adattarsi in qualsiasi momento: i sensori sono dispositivi che catturano ed inviano informazioni sulla rete; gli attuatori sono dispositivi che ricevono informazioni e le convertono in azioni; i system device sono componenti del sistema che possono ricevere informazioni, elaborarle ed infine trasmetterle agli attuatori per eseguire un'azione.

Importazione da file

Il software On.Energy permette anche di importare valori direttamente da file costruiti dall'utente. Il file che il sistema è in grado di leggere sono di tipo "xls" ed "xlsx" e devono essere formattati in base alla tipologia di variabile del sistema che si vuole alimentare.

Inserimento manuale

È possibile inserire o modificare dati manualmente direttamente dal software accedendo ai dati della variabile interessata.

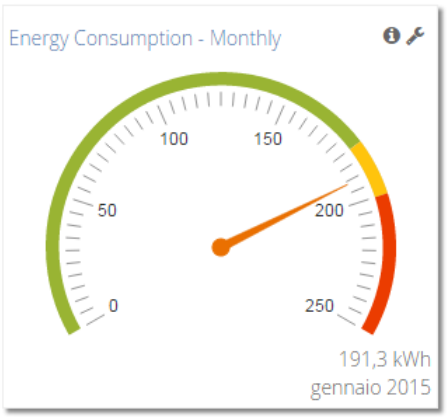
Le possibilità con cui il sistema alimenta le proprie variabili risulta abbastanza ampio e variegato. Purtroppo la maggior parte dei produttori hardware utilizza protocolli proprietari o una personalizzazione dei protocolli standard (questo perchè la maggior parte di essi sviluppano anche software di gestione o semplicemente per fidelizzare la clientela) rendendo inefficace l'utilizzo degli standard internazionali.

3.4.2 Monitoraggio e analisi dei dati

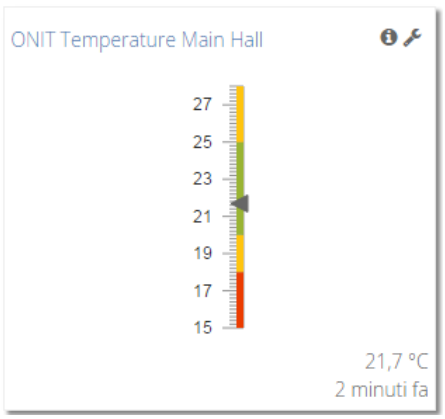
Una delle funzionalità più avanzate messe a disposizione da On.Energy riguarda il monitoraggio e l'analisi dei dati. Per entrambi il software mette a disposizione numerosi strumenti grafici intuitivi e di facile configurazione.
È possibile individuare due modalità di analisi: analisi dei dati in tempo reale (monitoring), in cui l'informazione estrapolata è il valore attuale (o l'ultimo recuperato dal sistema), e analisi di serie storiche (data analysis),

in cui si studia l'andamento di uno stream o indicatore, potendo verificare anche le relazioni tra diverse variabili.

Una prima informazione sul valore attuale di una variabile si ottiene subito dalla modalità con cui stream e indicatori sono rappresentati all'interno dei nodi nella maschera di navigazione (vedi figura 3.2 e figura 3.3). Come si può notare infatti viene presentato l'ultimo valore assunto dalla variabile e il timestamp (ovvero data e ora) in cui questo è stato rilevato.
Un'altra possibilità per ottenere queste informazioni è fornita dai widget. I widget sono componenti grafici posizionabili all'interno dei nodi che permettono, previa configurazione, un'analisi più approfondita delle informazioni, estrapolando così dati che altrimenti non sarebbe possibile ottenere.



(a) widget cruscotto



(b) widget lineare

(c) widget live

Figura 3.5: Esempio widget controllo real-time

Nella figura 3.5 si può vedere un esempio di widget utilizzabili per il monitoring di valori real-time. In particolare per quanto riguarda i widget rappresentati in figura 3.5a e 3.5b è possibile definire anche delle soglie per rappresentare graficamente gli intervalli entro i quali i valori delle variabili dovrebbero rimanere, permettendo così di individuare immediatamente valori anomali.

Un'altra importante funzionalità fornita per il controllo real-time è rappresentata dalla gestione degli allarmi. Come già accennato la gestione degli allarmi è uno dei requisiti più di base per un sistema di controllo automatico

e permettono di avvisare l'utente nel caso in cui una variabile dell'impianto assuma valori al di fuori di intervalli di soglia prefissati. On.Energy visualizza un riepilogo degli allarmi sulla schermata principale fornendo all'utente una rapida identificazione delle anomalie. Inoltre il software fornisce un servizio di notifica via sms e e-mail.

Per quanto riguarda l'analisi di serie storiche On.Energy mette a disposizione un insieme di grafici che permettono l'analisi dell'andamento di una variabile e la comparazione di questa con altre. Il primo e più importante di questi widget è il "widget grafico". Questo permette di visualizzare i dati di una o più serie storiche in due modalità:

- Tabella: i dati vengono presentati così come salvati sulla base di dati. È comunque possibile aggregare i valori, ma questo tipo di visualizzazione non fornisce una grande mole informativa in quanto non è facilmente intuibile l'andamento della serie.

- Grafico: costruisce un grafico a linee, barre o area delle serie prese in considerazione. È possibile selezionare il periodo da considerare e il software presenterà un grafico in cui sull'asse delle ascisse sarà presente la variabile tempo mentre sulle ordinate i valori delle serie nel periodo.

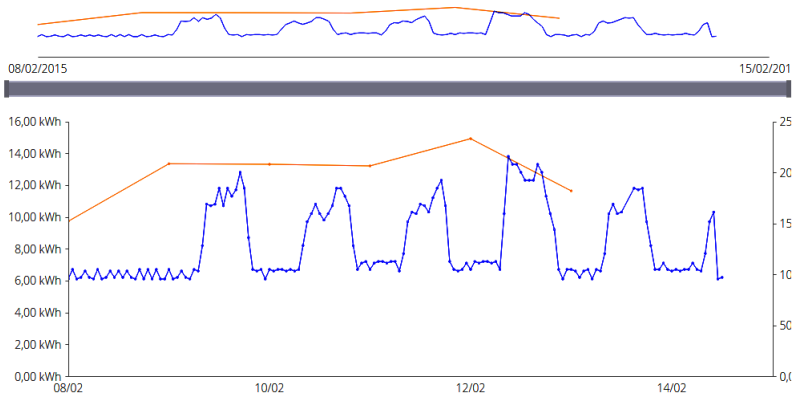
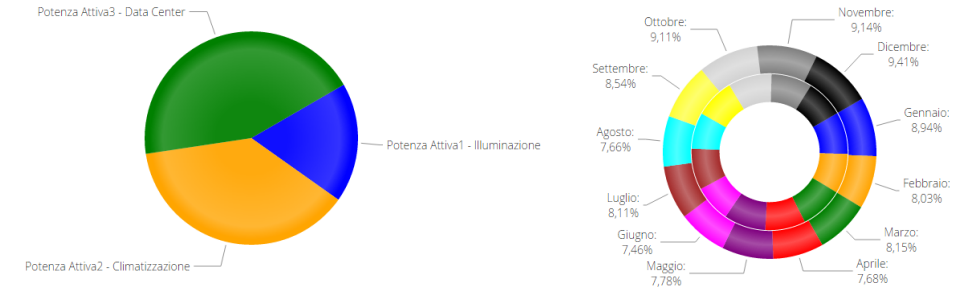


Figura 3.6: Esempio visualizzazione grafico

Un'altro widget utile per l'analisi di serie storiche è il "widget torta". Questo presenta due modalità di visualizzazione: la prima è la classica rappresentazione di un grafico a torta in cui più serie storiche possono essere inserite e confrontate, mentre la seconda è una visualizzazione a ciambella in cui un'unica serie storica può essere inserita ed analizzata periodo su periodo. Nel caso del "widget torta" non è possibile scegliere il periodo di visualizzazione, verrà sempre preso l'ultimo periodo ed i dati saranno aggregati secondo le scelte dell'utente.



(a) widget torta (b) widget ciambella

Figura 3.7: Esempio widget torta e ciambella

Per tutti i widget è possibile effettuare sia aggregazioni su base temporale (giornaliera, mensile, annuale, ecc.) sia in base alle etichette assegnate al nodo su cui si è scelto di generare il widget, permettendo così di aggiungere informazioni significative all'analisi.

Infine come ultima risorsa di analisi dei dati e delle serie storiche, On.Energy, permette di definire una nuova base di dati per eseguire sessioni di analisi basate su informazioni multidimensionali. Il tutto si concretizza nella creazione di un data warehouse sulle informazioni di interesse che attraverso una tabella permette all'utente la consultazione, la navigazione e l'aggregazione in stile OLAP, fornendo cosi un potente strumento decisionale e di data mining.

Capitolo 4

Analisi

In questo capitolo si analizzeranno i limiti della soluzione On.Energy che hanno portato l'azienda Onit S.R.L a decidere di svuluppare un modulo dedicato all'analisi di stima oggetto di questo lavoro di tesi.
Il modulo da sviluppare dovrà essere estremamente configurabile e permettere all'utente di eseguire i calcoli necessari in pochi e semplici passi, alla conclusione del progetto il modulo sarà inserito nella versione HTML5 di On.Energy.

Il risultato dell'analisi dei requisiti e del problema è un documento a cui ci si deve attenere rigorosamente durante la fase di sviluppo e deve eliminare tutti i fattori causa di ambiguità che potrebbero portare a difficoltà aggiuntive durante l'ultima fase del progetto.

4.1 Analisi dei requisiti

Durante un'attenta revisione del software On.Energy, l'azienda Onit srl si è accorta della mancanza della possibilità di poter produrre una stima di una variabile basata sulla dipendenza di questa da altre variabili; o meglio questo era fattibile, ma solo dopo una complicata procedura di costruzione di diversi indicatori che permettessero di eseguire i calcoli necessari.
Questo non poteva essere accettabile in quanto la stima dell'andamento di una variabile per sistemi di tipo Energy Management è considerata una funzionalità fondamentale per capire se ci sono stati sprechi o consumi inaspettati, inoltre, per gli utenti poco esperti in questo tipo di analisi, le operazioni richieste per produrre una stima risultavano particolarmente difficoltose e quindi praticamente mai eseguite. Per questo lo sviluppo di un modulo che automatizzasse tutti i passi richiesti per produrre una formula di stima teorica è diventato un requisito prioritario da inserire nella nuova versione software On.Energy.

Il primo punto dei requisiti riguarda la scelta delle modalità di produzione di un modello di stima per una variabile messa in relazione con altre n variabili.
In letteratura esistono vari metodi di stima che dovranno essere presi in esame per verificarne l'efficacia nell'ambito in esame.

Il secondo requisito riguarda la selezione delle variabili; per la buona riuscita dell'analisi è infatti necessario che la scelta delle variabili da in-

cludere nel modello sia ponderata, e non si utilizzino stream o indicatori completamente slegati tra loro. È quindi nata la necessità di guidare l'utente in questa scelta e dargli un'indicazione su quali variabili mantenere per la costruzione del modello e quali escludere in quanto non significative. È comunque importante che l'utente sia consapevole di ciò che sta facendo e abbia un'idea del risultato che vuole raggiungere.

Il terzo punto richiesto riguarda la customizzazione dell'intero processo; i vari passi che l'utente dovrà eseguire dovranno essere il più personalizzabili possibile, ad esempio l'utente dovrà essere in grado di scegliere se aggregare i valori delle variabili (mensilmente, annualmente, ecc.) e quale funzione di aggregazione utilizzare (sommatoria, media, massimo, minimo, ecc.); inoltre deve essere data la possibilità di scegliere il metodo di risoluzione più adeguato (si veda la sezione 2.2), oltre a definire le modalità per l'individuazione e lo scarto di valori anomali (outliers); il tutto dovrà essere semplice ed intuitivo. Questo è un requisito fondamentale in quanto utenti più esperti potrebbero voler customizzare il più possibile il metodo di calcolo, ma non si devono appesantire troppo i passaggi per permettere anche ad utenti meno esperti di eseguire l'analisi, senza dover studiare approfonditamente l'argomento.
Durante la definizione di questo requisito si è posta particolare attenzione alla scelta del periodo di selezione dei dati su cui poi si dovrà calcolare il modello. L'utente deve essere in grado di scegliere il periodo a lui più congeniale, inoltre deve poter scegliere se utilizzare categorie ed etichette per suddividere ulteriormente i dati. Il modulo deve essere in grado di filtrare i

dati e produrre un modello differente per ogni etichetta scelta dall'utente.

Il quarto punto definito nei requisiti riguarda la visualizzazione del risultato; deve essere chiaro fin da subito quando un modello può essere considerato rappresentativo della variabile dipendente e quando invece non lo è.

Infine il quinto ed ultimo punto dedotto dall'analisi dei requisiti riguarda il salvataggio dei risultati; è chiaro, che dopo aver costruito il modello, il risultato sarà una formula in funzione delle variabili indipendenti. L'unica possibilità che ha il software On.Energy per elaborare una formula di questo tipo è tramite la costruzione di un indicatore. Perciò al termine della costruzione del modello dovrà essere possibile creare un nuovo indicatore all'interno del sistema, impostando le ultime informazioni necessarie (come ad esempio l'unità di misura o il periodo di validità della formula), e procedere al salvataggio e alla computazione.

4.2 Analisi del problema

Dopo aver individuato i requisiti ed i vincoli da rispettare per concludere con successo lo sviluppo del modulo, è possibile iniziare a pensare come procedere ed individuare le eventuali criticità. In questa sezione verranno presi in considerazione i punti individuati precedentemente e proposta una possibile soluzione, che fungerà da guida durante la fase implementativa.

Il primo requisito riguarda la scelta di una metodologia di stima. Come si è scritto nel Capitolo 2, l'analisi di regressione rispecchia alla perfezione le richieste definite per il modulo da implementare; è comunque da considerare uno studio iniziale dei diversi metodi proposti in letteratura in modo da catturarne gli aspetti positivi e negativi e capire i vincoli a cui sono soggetti. Ci si dovrà concentrare in particolar modo sull'aspetto informatico, quindi cercando l'esistenza di eventuali librerie in grado di fornire supporto durante l'implementazione delle procedure di calcolo.

Una volta analizzate nel dettaglio le varie metodologie ed individuati gli strumenti necessari allo sviluppo, si potrà procedere con la progettazione del nuovo modulo per il sistema On.Energy. Innanzitutto si richiede di guidare l'utente nella selezione delle variabili; è quindi necessario dare un'indicazione di come e quanto due variabili siano legate (si vedano le sezioni dedicate alla correlazione e alla tipologia di relazione nel Capitolo 2.1.2), in modo che l'utente sia in grado di capire la significatività di una variabile nel modello e possa scegliere la relazione che più si adatta all'andamento delle variabili prese in esame (ad esempio una variabile indipendente potrebbe avere una relazione lineare rispetto alla variabile dipendente, mentre un'altra potrebbe essere legata da una relazione parabolica con la stessa variabile dipendente).

Per quanto riguarda la customizzazione del processo si è deciso di permettere all'utente di scegliere tutti i parametri necessari al calcolo del modello (ad esempio per la scelta del metodo di risoluzione, della soglia di scarto degli outliers o del numero di iterazioni), fornendo anche un insie-

me di valori predefiniti, calcolati in base a formule illustrate nel capitolo 2; inoltre si dovrà dare la possibilità di aggregare i dati delle serie temporali e filtrarli in base alle date di rilevazione (sia in per un periodo selezionato che per etichette precedentemente configurate).

La parte di recupero e pulizia dei dati da dare in input all'algoritmo presenta una serie di criticità che è utile analizzare. Ogni tipologia di serie serie storica (puntuale, aggregata o accumulatore), presente sul software On.Energy, possiede una propria modalità di salvataggio dei dati, in particolare per quanto riguarda l'istante di rilevazione (ad esempio una serie puntuale individua un unico istante, mentre una serie aggregata mantiene i due istanti di inizio e fine del periodo di aggregazione); inoltre la possibilità di avere più variabili con periodi di campionamento differenti complica ulteriormente le cose. Ogni algoritmo di risoluzione considerato accetta in input variabili che possiedano la stessa numerosità di osservazioni; si rende perciò necessario individuare un metodo che permetta di trovare la corrispondenza temporale tra i valori delle variabili e scartare i dati senza corrispondenza, il tutto rispettando la logica già presente nel software per non creare inconsistenze logiche e interpretative.
Inoltre si è considerata la possibilità che una variabile possa dipendere da se stessa o da altre variabili ad istanti precedenti (ad esempio $t - 1$), si è quindi introdotto anche questo concetto, permettendo il time shifting dei valori in base al periodo di campionamento delle variabili o all'aggregazione scelta dall'utente.

Il risultato della stima deve essere fornita sotto forma di espressione in cui compariranno i coefficienti calcolati e le variabili indipendenti; inoltre per ogni espressione trovata (possono essere più di una se si utilizzano etichette) deve essere visualizzata e spiegata la bontà del modello, in modo da chiarire se il risultato è accettabile oppure no. Quando l'utente deciderà di salvare il risultato si dovrà presentare una mascherà che fornirà un riepilogo delle scelte fatte e permetterà di salvare l'*indicatore* corrispondente.

4.3 Tecnologie utilizzate

Dopo che nelle sezioni precedenti si sono analizzati i requisiti e le criticità ad essi legate ora si può procedere ad un'analisi della struttura che si intende dare al progetto e delle tecnologie utilizzate in fase di sviluppo (a livello di linguaggi di programmazione).
In questa sezione verranno quindi presi in esame sia i pattern architetturali che i diversi linguaggi scelti per implementare il modulo di On.Energy da realizzare.

4.3.1 Pattern Model-View-Controller

MVC è un pattern architetturale molto utilizzato (soprattutto per applicazioni web-based), che permette di separare le parti di logica applicativa (prese in carico da Model e Controller) dalle parti di visualizzazione (prese in carico dalla View) presenti in un software.

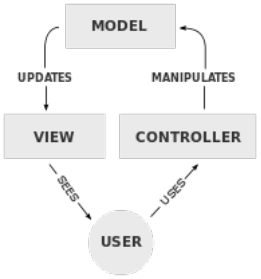


Figura 4.1: Struttura di un'applicazione MVC [16]

Come si può notare in figura 4.1, in pratica il software viene suddiviso in tre parti (Model, View e Controller) indipendenti tra loro ma in grado di comunicare per eseguire le funzionalità richieste dall'utente.
Per meglio comprendere questo pattern è necessario dare una breve spiegazione di questi tre componenti:

- Model: Cattura il comportamento dell'applicazione in termini di dominio del problema. La parte di model contiene tutti i dati ed i metodi di accesso ad essi, che possono essere utilizzati da View e Controller per definire il loro comportamento. Il Model non fornisce soltanto l'accesso fisico ai dati, ma si preoccupa anche di creare il necessario livello di astrazione tra il formato in cui i dati sono memorizzati alla fonte e il formato in cui i livelli di Controller e View si aspettano di riceverli.

- View: Visualizza i dati messi a disposizione dal Model nel formato scelto. Possono esistere più tipologie di rappresentazioni per uno stesso dominio di informazioni (ad esempio dati rappresentati in tabella o in un diagramma a torta), è perciò possibile che più View utilizzino

uno stesso Model per rappresentare le informazioni in esso contenute in modalità differenti.

- Controller: Riceve i comandi dall'utente e li attua modificando lo stato del Model. In pratica rimane in attesa che un utente esegua un'operazione attraverso la View, a quel punto la elabora e salva le informazioni generate, reindirizzando l'utente ad un'altra View se necessario.

La struttura dell'applicazione in questo modo offre diversi vantaggi. Infatti data l'indipendenza tra i componenti è possibile effettuare modifiche su ognuno di essi senza dover riprogettare il comportamento dell'intero dominio applicativo; inoltre ogni componente viene semplificato e risulta di più facile gestione, purtroppo di contro si complica molto la struttura applicativa.

ASP.NET MVC

ASP.NET MVC è un framework sviluppato da Microsoft che fornisce un insieme di classi e funzionalità per aiutare a strutturare un'applicazione web utilizzando il pattern Model-View-Controller. Essendo ASP.NET MVC il framework utilizzato all'interno del progetto On.Energy è importante definirne alcune peculiarità.

Struttura delle cartelle Il framework genera un insieme di cartelle predefinite (Model, View, Controller, Script e Content) in cui posizionare i vari elementi del progetto. Nella cartella Script prima andranno inseriti tutti gli script Javascript che si intende utilizzare, nella cartella Content i fogli di stile e i contenuti grafici delle pagine.

Le ultime tre cartelle, come suggeriscono i loro nomi, conterranno i file destinati ad interpretare i ruoli di Model View e Controller. La suddetta suddivisione è una semplice convenzione, che può essere ignorata previa configurazione dell'applicazione.

Attributi All'interno di un Controller è possibile definire degli attributi per ogni metodo (o action), che permettono di definirne il comportamento. Un esempio di attribuiti sono [HttpPost] o [HttpGet] che indicano il tipo di richiesta HTTP che un metodo è in grado di gestire, oppure [Authorize] che permette l'accesso ad un metodo solo se l'utente è autorizzato e quindi ha eseguito l'accesso all'applicazione, infine [ValidateAntiForgeryToken] che permette di verificare l'integrità delle informazioni contenute in una richiesta. Per alcuni attributi è possibile specificare anche informazioni aggiuntive che ne caratterizzano ulteriormente il comportamento (es. [Authorize(Roles="administrators, users")]).

Helper Il framework ASP.NET MVC fornisce un insieme di classi e metodi con lo scopo di agevolare la creazione di codice html all'interno delle View. Utilizzando questi helper è quindi possibile costruire intere form oppure porzioni di html all'interno di un documento, sarà poi il server ad associare ad ogni chiamata la stringa in linguaggio HTML vero e proprio (es. using (Html.BeginForm("Url", FormMethod.Post, new role = "form", name = "Name")) permette di creare il tag form in cui inserire i vari input di cui si vuole eseguire il submit all'url specificato).
Oltre ad helper dedicati alla costruzione di codice HTML esistono metodi

utili anche per la generazione di Url a partire dal nome di un metodo del Controller e per la costruzione di codice Ajax.

Validazione Il framework permette anche di gestire la validazione di campi di input modificabili dall'utente. Attraverso la classe "ModelState" è possibile verificare se un modello risulta valido ed inserire messaggi di errore che potranno essere visualizzati nella View per avvertire l'utente.

Partial View e Layout Entrambi danno la possibilità di riutilizzare codice all'interno di più pagine. La differenza tra le due è che le partial view rappresentano una porzione di una pagina e possono essere utilizzate più volte all'interno di una stessa View o da View diverse, mentre i layout costituiscono il template per la visualizzazione di pagine diverse. All'interno di un layout sarà poi inserito dall'applicativo il contenuto delle pagine che ereditano il layout stesso.

4.3.2 Razor View Engine

Un altro importante concetto nel paradigma MVC è rappresentato dal *View Engine*. Il View Engine è quel componente che si occupa della ricerca, esecuzione e visualizzazione della vista selezionata al termine delle operazioni di un Controller.
ASP.NET MVC possiede un View Engine di default che prende origine dal classico ASP.NET, ma è possibile definire un proprio "motore" configurando l'applicazione per il suo utilizzo.
Questo, inoltre, permette allo sviluppatore di inserire codice eseguito lato

server all'interno delle View, creando così pagine dinamiche. Queste porzioni di codice saranno riconosciute ed interpretate dal View Engine prima di restituire la pagina al client.

Nel progetto On.Energy e nel modulo oggetto del lavoro di tesi si è scelto di utilizzare *Razor View Engine*, risulta quindi doveroso darne una breve introduzione.
Razor è in grado di interpretare codice C# all'interno di pagine HTML a patto che questo sia preceduto dal simbolo '@' e, in caso di più istruzioni consecutive, racchiuso da parentesi graffe. Questo ha facilitato molto la programmazione con il paradigma ASP.NET MVC in quanto è possibile utilizzare un unico linguaggio di programmazione sia nei Controller e Model che nelle View per la creazione di pagine dinamiche. Inoltre Razor è stato progettato per richiedere il minor numero di caratteri, è perciò possibile inserire codice C# senza dover indicare dove questo si interrompa (si indica l'inizio con il simbolo '@' ma non la fine).
Un semplice esempio di HTML con Razor:

```
@helper ProductListing(List<Product> products) {
    <ul id="products">
        @foreach(var p in products) {
            <li>@p.Name ($@p.Price)</li>
        }
    </ul>
}
```

Figura 4.2: Esempio di html con Razor [28]

Nell'immagine 4.2 viene creata una lista HTML di prodotti in modo dinamico da un oggetto di tipo "List" C#, inserendo delle istruzioni all'interno dell'albero HTML. Ogni riga di codice C# possiede un tag che ne indica l'inizio ma nessuno che ne indica la chiusura (codice di semplice interpretazione e pulito).

4.3.3 Entity Framework

L'*Entity Framework* è un componente di .NET Framework che permette agli sviluppatori di separare il dominio dei dati dell'applicativo dalla logica di accesso alle sorgenti, quindi dalla gestione della memorizzazione fisica e del recupero dei dati stessi. In pratica fornisce un livello più alto di astrazione permettendo di definire e manipolare le entità di cui abbiamo bisogno senza entrare nel merito della rappresentazione fisica dei dati.

L'Entity Framework fornisce un insieme di classi ognuna corrispondente ad una tabella presente nel database applicativo. Ogni classe fornirà un insieme di campi (corrispondenti alle colonne della tabella presa in considerazione) e metodi di accesso ad essi. Quando si vuole accedere in lettura o modifica a dati presenti nel DB lo si potrà fare tramite un'istanza di una di queste classi, sarà poi l'Entity Framework a tradurre le richieste utilizzando il metodo più consono per interrogare la base di dati. Per fare ciò il framework utilizza un insieme di file in formato XML che descrivono la modalità di memorizzazione fisica dei dati e le relazioni presenti tra le varie entità rappresentate.

Code First

La modalità *Code First* dell'Entity Framework permette di creare e modificare classi che saranno poi tradotte dal framework in nuove tabelle o in modifiche di tabelle già presenti. Quindi si può partire nel definire le classi appartenenti al dominio dei dati dell'applicativo, senza che la base di dati sia presente o possegga tutte le entità necessarie, in quanto queste verranno generate dinamicamente.

Per poter utilizzare questa modalità è necessario creare una classe di tipo *DbContext* in cui verranno definiti i mapping tra le classi appartenenti al modello dei dati e le tabelle nel DB, oltre ad un insieme di regole utili per la gestione. Ad ogni modifica di una delle classi corrispondente ad una tabella del DB è necessario creare una *migration* (che verrà generata automaticamente con il comando Add-Migration <name> da linea di comando), ogni migration specifica le operazioni SQL da eseguire per aggiornare il database.

Ogni classe definita può essere "decorata" con annotazioni che forniscono informazioni aggiuntive sulla tabella o su un singolo campo presente in essa (come ad esempio il vincolo di chiave primaria, univocità, ecc.).

```
public class Blog
{
    public int BlogId { get; set; }
    public string Name { get; set; }

    public virtual List<Post> Posts { get; set; }
}

public class Post
{
    public int PostId { get; set; }
    public string Title { get; set; }
    public string Content { get; set; }

    public int BlogId { get; set; }
    public virtual Blog Blog { get; set; }
}

public class User
{
    [Key] //indica che il campo username e' chiave
    public string Username { get; set; }
    public string DisplayName { get; set; }
}

public class BloggingContext : DbContext
{
    public DbSet<Blog> Blogs { get; set; }
    public DbSet<Post> Posts { get; set; }
    public DbSet<User> Users { get; set; }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        //Definisce la lunghezza del campo Title della tabella post
        modelBuilder.Entity<Post>().Property(p => p.Title).HasMaxLength(100);
    }
}
```

Nell'esempio sopra esposto si definiscono tre classi (Blog, Post e User) che saranno poi tradotte dall'Entity Framework nelle tabelle corrispondenti.

```
using (var db = new BloggingContext())
{
    // Creazione e salvataggio di un nuovo blog
    var name = Console.ReadLine();

    var blog = new Blog { Name = "Blog1" };
    db.Blogs.Add(blog);
    db.SaveChanges();
}
```

Infine in questo ultimo esempio viene mostrato come inserire una nuova riga in una tabella; come si può notare viene creato un nuovo oggetto di tipo Blog ed utilizzato questo per valorizzare i campi da inserire nel DB.

Language Integrated Query

Language Integrated Query (o LINQ) è un componente .NET che aggiunge la possibilità di effettuare interrogazioni su oggetti utilizzando una sintassi simile a SQL; non è quindi più necessario imparare un linguaggio di query diverso per ogni tipo di origine dati (database SQL, documenti XML, ecc.); inoltre, grazie all'IDE di Visual Studio, viene fornito l'aiuto dell'IntelliSense nella scrittura di queste query (mentre prima dovevano essere scritte tramite stringa).
LINQ definisce un insieme di operatori che possono essere usati per interrogare, proiettare e filtrare dati rappresentati come oggetti.

```
var results =
    SomeCollection
        .Where(c => c.SomeProperty < 10)
        .Select(c => new {c.SomeProperty, c.OtherProperty});
```

Nell'esempio sopra esposto viene filtrata una collezione di oggetti in base al valore di una proprietà, ed infine vengono selezionati due campi degli

elementi restanti. Come si può notare si utilizza una notazione tipica di SQL che permette di risparmiare codice ed una più immediata comprensione di quanto scritto.

4.3.4 HTML5

HTML5 è la quinta versione del linguaggio di markup HTML. La stesura delle sue specifiche iniziò nel 2004 ad opera del consorzio Web Hypertext Application Technology Working Group composto da alcune delle più importanti aziende operanti nel settore, a cui successivamente si unì il World Wide Web Consortium (W3C) con cui si concluse la standardizzazione della versione.
A differenza delle precedenti versioni, HTML5 è stato progettato specificatamente per poter creare applicazioni desktop-like sul web; fornisce quindi un insieme di funzionalità per facilitare la progettazione di complesse interfacce grafiche interattive e per supportare l'esecuzione di complicate elaborazioni. Le nuove funzionalità fornite da HTML5 possono essere qui brevemente elencate:

- indipendenza dal device utente;
- accedere all'applicazione anche in assenza di una connessione Web;
- salvare informazioni sul device dell'utente;
- usare tipologie di interazione tipiche di applicazioni desktop, come il drag and drop;

- comunicare in modo bidirezionale sia con il server sia con altre applicazioni;
- introduzione di nuovi tag come <header>, <footer>, <nav> e <article> per supportare la strutturazione della pagina. È importante sottolineare che ognuno dei tag prevede una semantica ben definita all'interno della pagina;
- introduzione di nuovi tag come <audio> e <video> per il supporto di contenuti multimediali senza l'utilizzo di plugin.

Una interessante caratteristica di HTML5 è che tutte le strutture di HTML4 rimangono valide e funzionanti. Questo perchè buona parte del web è ancora rappresentato da siti statici o document-oriented e renderle obsolete sarebbe stato distruttivo.

Bootstrap

Bootstrap è un framework front-end per HTML e CSS utile per uno sviluppo semplice e veloce di interfacce web. Il suo sviluppo è iniziato nel 2011 da parte dei creatori di Twitter e oggi è alla sua versione 3.0.
Bootstrap fornisce un insieme di classi CSS e funzioni Javascript che permettono di strutturare agilmente una pagina web sia in ambiente desktop che mobile, oltre ad una serie di template per diversi componenti grafici (come button, form, table, ecc.).

Una pagina creata con Bootstrap assume la struttura di una griglia (grid view) di dimensione variabile a seconda della grandezza della finestra su cui

viene visualizzata (responsive web). Inoltre è compatibile con tutti i maggiori browser, permettendo quindi di non dover adattare la visualizzazione rispetto allo user-agent.

4.3.5 Javascript

Javascript è un linguaggio di scripting orientato agli oggetti e agli eventi utilizzato per la creazione di pagine dinamiche e per permettere di spostare parte della computazione lato client. Javascript infatti viene interamente preso in carico dal client e le sue istruzioni vengono eseguite dal web browser. Per questo con il passare del tempo è diventato sempre più importante avere degli strumenti performanti in grado di interpretare questo linguaggio: ne è un esempio il V8 JavaScript sviluppato da Google per Chrome.

Nonostante tutto Javascript non risultava uno strumento molto potente per lo sviluppo di applicazioni web interamente basate su questo linguaggio, soprattutto per la mancanza della possibilità di comunicare con il lato server. Con l'avvento di *Ajax* nei primi anni 2000, anche questa lacuna è stata colmata e Javascript è diventato uno strumento imprescindibile per lo sviluppo di applicazioni web. Grazie ad Ajax, infatti, è possibile effettuare richieste asincrone al server ed utilizzare i dati ritornati per generare informazioni real-time e aggiornare la pagina senza che questa debba essere ricaricata.
Sull'onda di queste nuove potenzialità si ebbe la nascita di nuove librerie in grado di semplificare lo sviluppo di applicazioni Javascript fornendo anche

supporto cross-browser. *JQuery* in particolare è la più famosa di queste librerie e offre un insieme di funzioni in grado di supportare lo sviluppo Javascript; ad esempio grazie alla funzione $() è possibile fare riferimento ad ogni singolo elemento presente nell'albero HTML e utilizzare su di essi un insieme di funzioni in grado di manipolarne l'aspetto, gestirne eventi, ecc.

Con l'arrivo di HTML5, inoltre, Javascript è stato arricchito con i concetti di *WebSocket* e *WebWorker*, il primo permette di aprire un canale di comunicazione bidirezionale tra client e server, mentre il secondo aggiunge il concetto di esecuzione asincrona, ovvero è possibile eseguire operazioni complesse in background non bloccando quindi l'esecuzione dell'intera pagina (thread).

Javascript è un linguaggio a tipizzazione debole, quindi permette di cambiare il tipo di una variabile runtime, questo però porta anche ad una serie di inconvenienti. Infatti la libertà di definire tipi runtime può causare inconsistenze logiche oltre che incomprensioni nel codice. Inoltre la mancanza di concetti come classi, interfacce e namespaces fa si che le applicazioni Javascript siano un insieme di event-handlers e funzioni non strutturate, rendendo la progettazione e lo sviluppo in larga scala particolarmente complesso, oltre a rendere caratteristiche come estendibilità e riutilizzo difficilmente applicabili.

Kendo UI

Kendo UI è un framework basato su Javascript e JQuery che fornisce un insieme di widget (come grafici, tabelle, datepicker, ecc.) in grado di snel-

lire l'impegno necessario allo sviluppatore per costruire componenti grafici complessi e permettere la compatibilità con tutti i maggiori browser.
Il framework è composto di tre pacchetti, ognuno utile per lo sviluppo di applicazioni web-based moderne:

- Kendo UI Web, per la realizzazione di interfacce Web evolute;
- Kendo UI DataViz, un insieme di widget per la visualizzazione grafica dei dati;
- Kendo UI Mobile, una libreria per la realizzazione di applicazioni web per il mobile.

Kendo UI è il principale strumento che è stato usato in On.Energy per la costruzione dei vari widget forniti dal software.

Capitolo 5

Sviluppo del modulo On.Energy

In quest'ultimo capitolo viene inizialmente presentata la struttura con cui è stato realizzato il nuovo modulo On.Energy obiettivo di questo lavoro di tesi, e successivamente saranno descritte le interfacce necessarie all'utente per l'utilizzo della nuova funzionalità, in modo da fornire al lettore una più completa visione su ciò che è stato fatto.

5.1 Panoramica sulla struttura del modulo

In questa sezione verrà presa in analisi la struttura che si è deciso di dare al modulo implementato prestando particolare attenzione ai casi d'uso e all'interazione tra i componenti. Questo è importante per capire il comportamento del modulo nella sua interezza e per verificare come i requisiti esposti nel capitolo precedente siano stati soddisfatti.

In prima battuta è utile analizzare il diagramma delle classi per verificare come si è deciso di impostare la struttura secondo il paradigma Model-View-Controller.

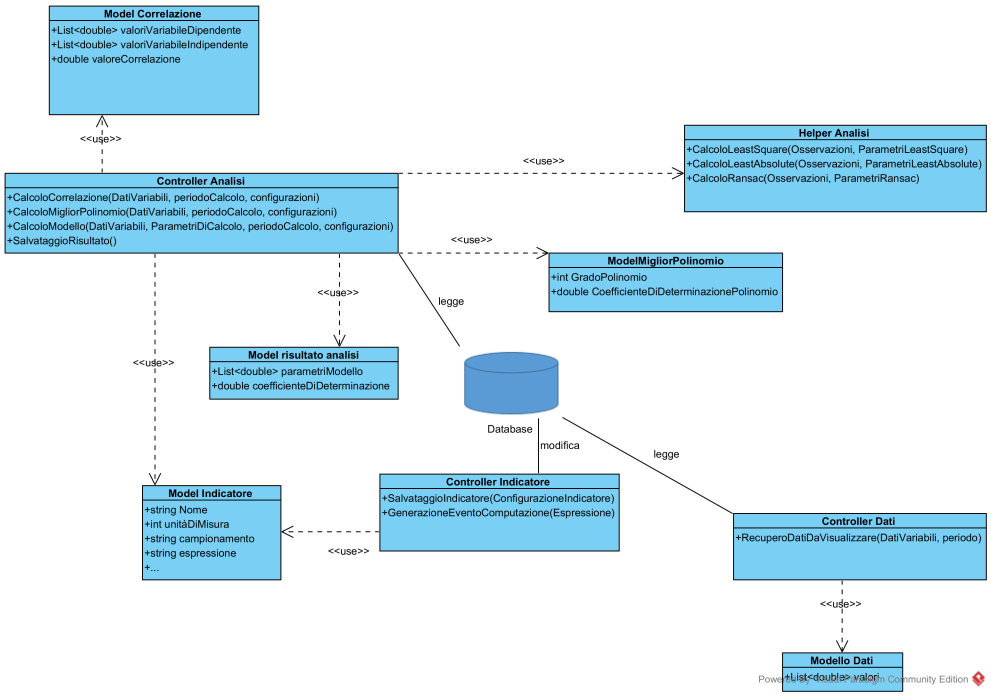


Figura 5.1: Diagramma delle classi

Come si può vedere dalla figura 5.1 il modulo è composto da una classe controller principale (*Controller Analisi*) che si occupa di ricevere tutte le richieste inviate dall'utente, effettuare i calcoli necessari ed infine costruire e inizializzare un oggetto model, che permetta alla view di mostrare i risultati. In particolare sono quattro le richieste (o action) che il Controller di Analisi può gestire:

1. Calcolo correlazione: questa richiesta verrà inviata dall'utente nel momento in cui ha scelto le variabili da inserire nel modello e il periodo su cui vuole costruire il modello stesso (oltre all'aggregazione temporale dei dati). Il controller riceverà queste informazioni e potrà procedere con il calcolo dell'indice di correlazione tra ogni coppia composta dalla variabile dipendente e ognuna delle variabili indipendenti (si è scelto di utilizzare l'indice di Spearman in quanto non si conosce la relazione tra la coppia di variabili). I valori delle variabili relativi al periodo selezionato verranno recuperate direttamente dalla base di dati con il supporto dell'Entity Framework, e dovrà poi essere il controller stesso ad occuparsi dell'aggregazione dei dati e a scartare i valori non coincidenti.

2. Calcolo miglior polinomio: questa funzionalità esposta dal controller permette di trovare il miglior polinomio che lega ogni coppia di variabili (dipendente, indipendente) utilizzando l'algoritmo presentato nel capitolo 2.

3. Calcolo modello: una delle funzionalità più importanti del Controller Analisi riguarda il calcolo e costruzione del modello di stima. Questa utilizzerà i metodi messi a disposizione dell'*Helper Analisi* per calcolare il miglior modello di stima per la variabile dipendente, considerando i parametri di configurazione (soglia di scarto outliers, numero di iterazioni, etc.) dati in input dall'utente.

4. Salvataggio risultato: infine l'ultima funzionalità disponibile riguarda il salvataggio dei risultati. Questa si dovrà appoggiare al controller

dedicato agli indicatori per poter salvare sul database la formula generata, e dare inizio alla computazione che verrà aggiornata ad ogni nuova rilevazione delle variabili indipendenti.

Un altro componente che si può notare dalla figura 5.1 è il *controller dati*. Questo si occupa principalmente del recupero delle informazioni (valori compresi) relative alle variabili per permetterne l'analisi dell'andamento temporale. Questa funzionalità è disponibile per ogni variabile presente nel sistema (stream o indicatore che sia), perciò essendo il modello di stima mappato su un normale indicatore anch'esso da la possibilità di effettuare un'analisi storica. Per gli indicatori generati in questo modo è però piú significativo un confronto tra la variabile stimata ed i valori effettivamente rilevati, si è quindi cercato di facilitare l'utente in questo senso presentando fin da subito entrambe le serie temporali nella maschera di analisi che verrà successivamente mostrata.

Casi d'uso

Dopo aver analizzato la struttura delle classi in stile MVC che si è deciso di dare al modulo di stima, è ora possibile individuare i casi d'uso che interesseranno l'utente durante l'utilizzo della nuova funzionalità.

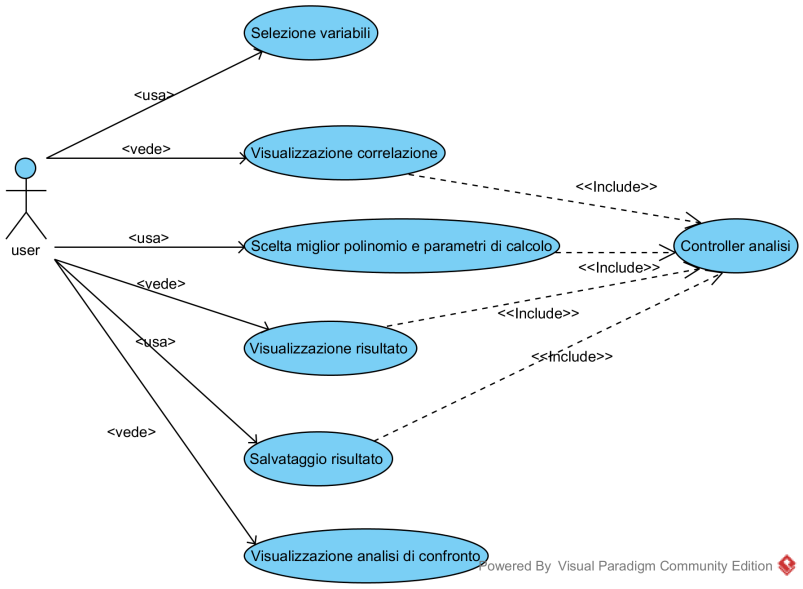


Figura 5.2: Casi d'uso

Come si può notare da figura 5.2 l'insieme dei casi d'uso è divisibile in due macro-gruppi: visualizzazione e configurazione.
La parte di visualizzazione comprende principalmente la comunicazione all'utente dei risultati derivati da calcoli (correlazione e creazione del modello); come già detto più volte, oltre a mostrare un mero risultato numerico, che potrebbe essere poco significativo per l'utente, è importante dare un'indicazione visiva e chiara su come interpretare questi valori. La parte di configurazione, invece, permette all'utente di effettuare le scelte necessarie alla personalizzazione dell'analisi da eseguire (scelta delle variabili, aggregazioni, metodo di risoluzione, etc.).
Ognuno dei casi d'uso individuati permetterà all'utente di utilizzare una delle funzionalità messe a disposizione dal *Controller Analisi* sopra esposte.

Interazioni

Per concludere con l'analisi della struttura del modulo verranno considerate le interazioni presenti tra i vari componenti, per capire quali siano le informazioni scambiate ed in quale ordine le richieste dovranno essere pervenute al Controller.

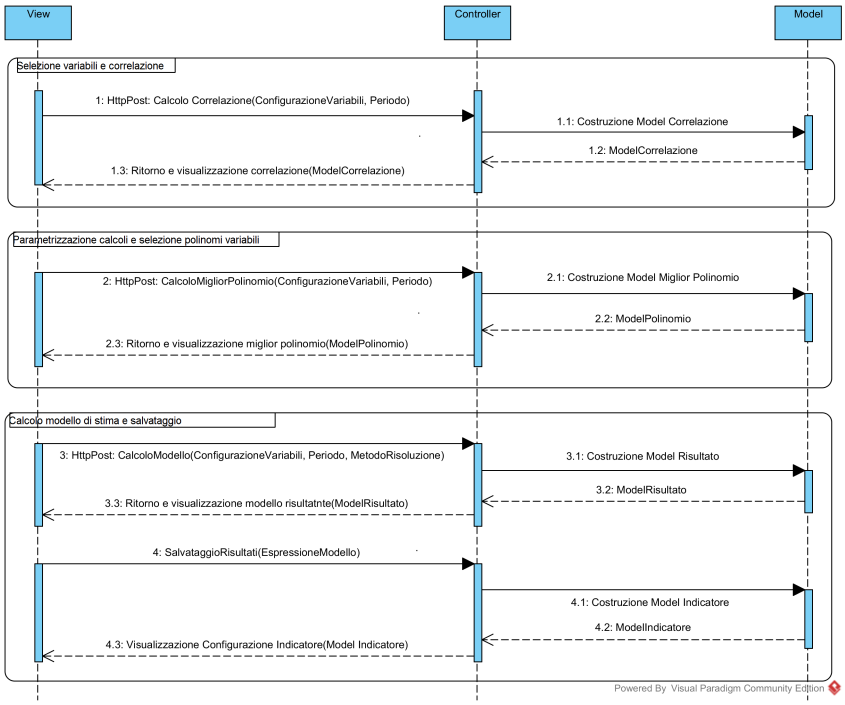


Figura 5.3: Diagramma delle interazioni

In figura 5.3 sono rappresentati i principali messaggi scambiati tra Model, View e Controller. Come si può notare le iterazioni avvengono utilizzando il protocollo HTTP e richieste di tipo POST. Si è scelto questo tipo di interazione in quanto la quantità di informazioni scambiate tra server e client può essere elevata (si possono includere molte variabili, e per ognuna

è necessario trasmettere le singole configurazioni scelte dall'utente) e il protocollo pone delle limitazioni riguardanti la dimensione del payload di altre tipologie di messaggi (ad esempio per HTTP GET).

L'intero processo ha inizio con l'invio delle richieste necessarie per il calcolo della correlazione e l'individuazione del grado del miglior polinomio di relazione tra le variabili. Per entrambe il server avrà necessità di conoscere le informazioni relative alle serie storiche e al periodo. Il passo successivo vede la generazione del modello di stima, inviando al server (Controller) una richiesta contenente oltre alle informazioni precedentemente viste, anche l'indicazione relativa al metodo di risoluzione scelto ed i parametri necessari al calcolo. Il processo si può considerare concluso con il salvataggio della stima in un indicatore e quindi con l'ultima richiesta di salvataggio che trasferisce al Controller le informazioni necessarie alla memorizzazione su database.

5.2 Interfacce generazione modello

La funzionalità di generazione del modello di stima è divisa in step in modo da guidare l'utente nei vari passi di configurazione e dare una suddivisione logica alle operazioni richieste.
In questa sezione verranno presentate e spiegate le maschere che un utente incontrerà durante i vari passi dell'analisi, in cui si potranno scegliere le diverse configurazioni disponibili e alla fine procedere con il salvataggio e l'analisi della serie storica generata.

Il sistema preso in esame nella descrizione dei vari step rappresenta la stima dell'energia elettrica consumata nel periodo 2013-2014 dagli uffici dell'azienda Onit stessa.

5.2.1 Scelta delle variabili

Il primo passo ha il compito di richiedere all'utente le informazioni generali riguardanti le variabili ed il periodo da prendere in esame e di calcolare gli indici di correlazione relativi.

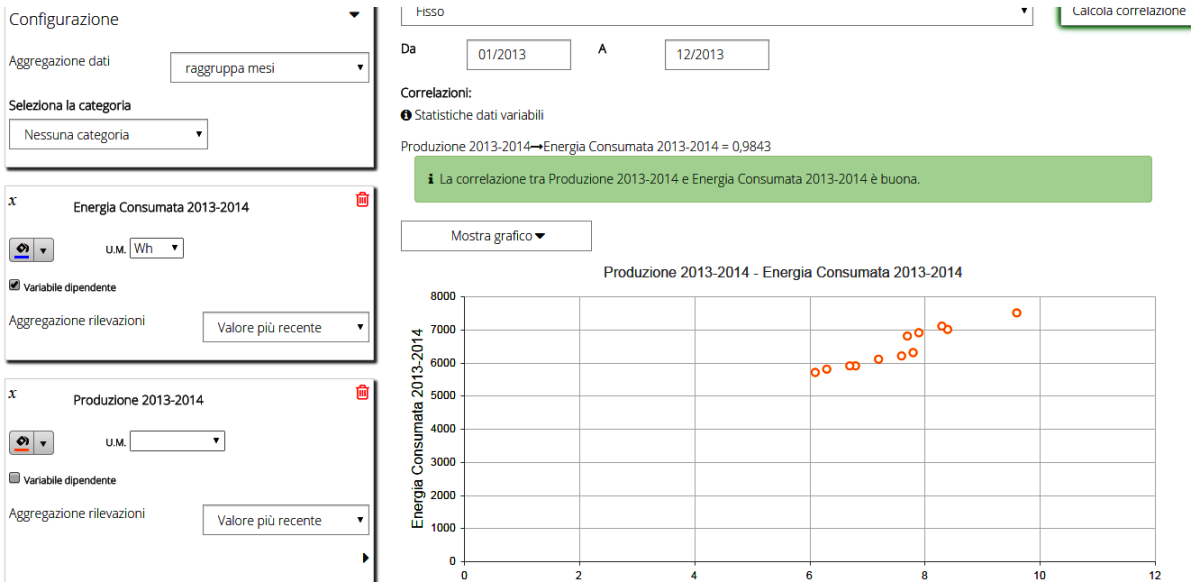


Figura 5.4: Passo 1: Scelta delle variabili e correlazione

Come si può vedere dalla figura 5.4 le informazioni che è richiesto specificare sono:

- la scelta del periodo di selezione dei valori delle variabili;
- l'unità di misura per ognuna delle varibili scelte;
- la tipologia delle variabili (dipendente o indipendente);
- un'eventuale aggregazione dei dati (oraria, giornaliera, mensile e annuale) e il corrispondente comportamento a seguito dell'aggregazione (si può prendere il valore più recente, meno recente, massimo, minimo oppure aggregare i valori nel periodo per somma o media).
- le eventuali etichette che permettono di suddividere ulteriormente i valori in base alla data di rilevazione.

Nell'esempio sopra riportato in figura 5.4 si è scelto di utilizzare i valori dell'anno 2013 aggregandoli mensilmente e prendendo il valore più recente per ogni mese.

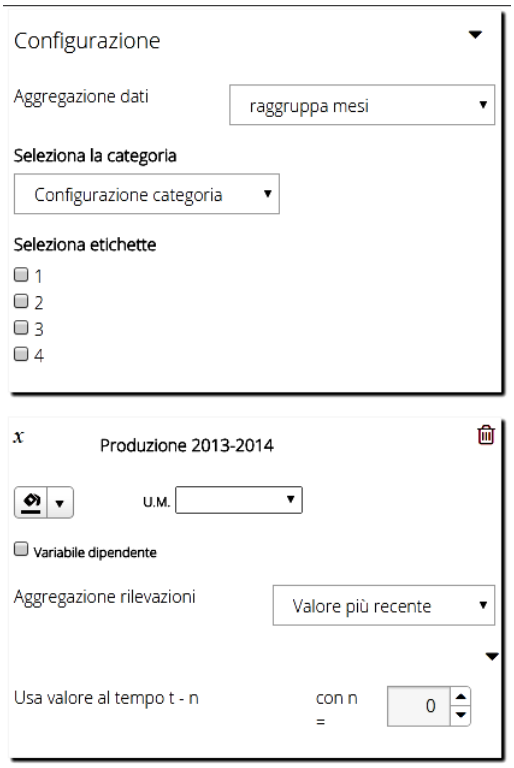


Figura 5.5: Passo 1: Scelta dello spostamento temporale

Un'altra importante informazione che l'utente può specificare riguarda lo spostamento temporale di una variabile (vedi figura 5.5), grazie a questo l'utente può considerare nel modello anche eventi precedenti che possono aver influenzato i valori della variabile dipendente.

Confermando i parametri scelti si procede al calcolo della correlazione. I risultati sono arricchiti con un messaggio che cambia a seconda del valore di correlazione risultante, e indica all'utente se può procedere con tranquillità oppure è opportuno cambiare le variabili in gioco.
Dopo il calcolo della correlazione è possibile procedere con il passo succes-

sivo.

5.2.2 Configurazione modello e metodo di calcolo

Prima di poter passare al calcolo del modello è necessario inserire le informazioni relative ai parametri di calcolo.
Il secondo passo prevede infatti la selezione del polinomio da utilizzare per ogni variabile e il metodo di calcolo del modello, quest'ultimo offre la scelta tra tre possibilità: Least Square, Least Absolute e RanSaC.
Per ognuno dei metodi sopra citati è possibile impostare i valori dei parametri di calcolo, scegliendo tra valori predefiniti calcolati dal sistema e valori personalizzati che può scegliere direttamente l'utente. Inoltre vi sono delle configurazioni specifiche per ogni metodo di calcolo che sono visibili solo se questo viene selezionato; ad esempio per il metodo Lest Square è possibile scegliere se scartare i valori anomali o utilizzare calcoli pesati, per far si che gli outliers influenzino meno il modello generato, mentre per il metodo RanSaC si può scegliere il numero di tentativi, in modo che, seppur utilizzando estrazioni casuali, la probabilità di trovare il miglior modello aumenti.

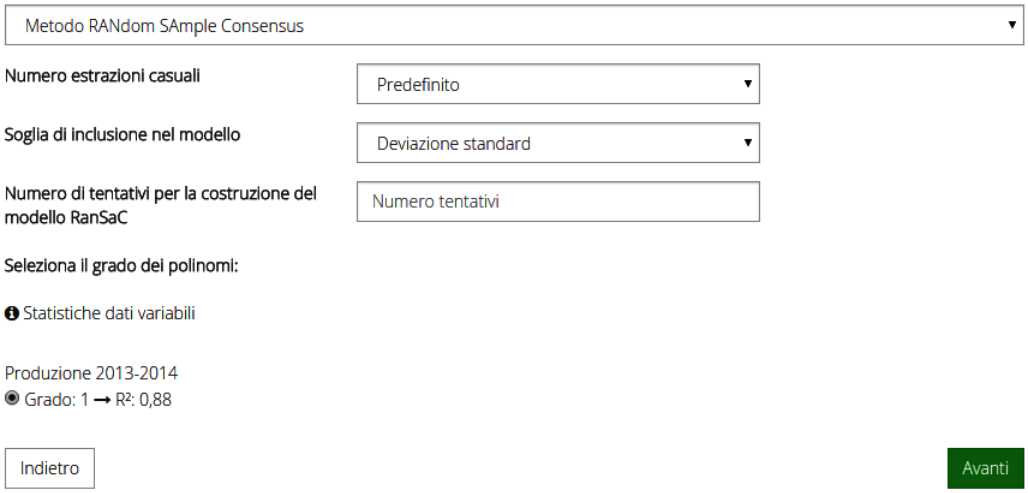


Figura 5.6: Passo 2: Configurazione modello e metodo di calcolo

A questo punto il sistema possiede tutte le informazioni necessarie e si può procedere al calcolo del modello di stima.

5.2.3 Visualizzazione risultati

Il terzo step applica tutte le configurazioni scelte dall'utente e calcola il modello di regressione, mostrando i risultati ed un messaggio riguardante la bontà del modello trovato.
In questa maschera l'utente conoscerà il risultato delle scelte da lui precedentemente fatte e potrà scegliere se procedere al salvataggio o tornare indietro e modificare le configurazioni per provare ad avere un risultato migliore.

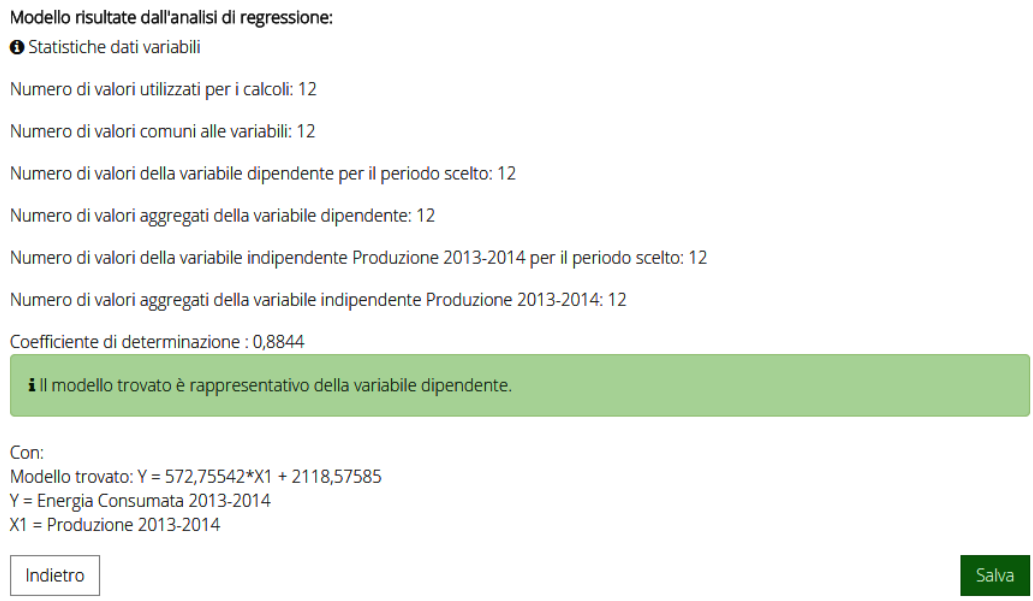


Figura 5.7: Passo 3: Visualizzazione risultati

Come si può vedere dalla figura 5.7 oltre alle informazioni sul modello risultate vengono mostrati anche i dati riepilogativi riguardanti il numero di valori utilizzati per la costruzione del modello ed i valori scartati per ogni variabile.

5.2.4 Salvataggio indicatore

Quando l'utente deciderà di salvare i risultati ottenuti dall'analisi verrà reindirizzato alla mascherà mostrata in figura 5.8, che gli permetterà di inserire gli ultimi dati necessari al salvataggio dell'indicatore.

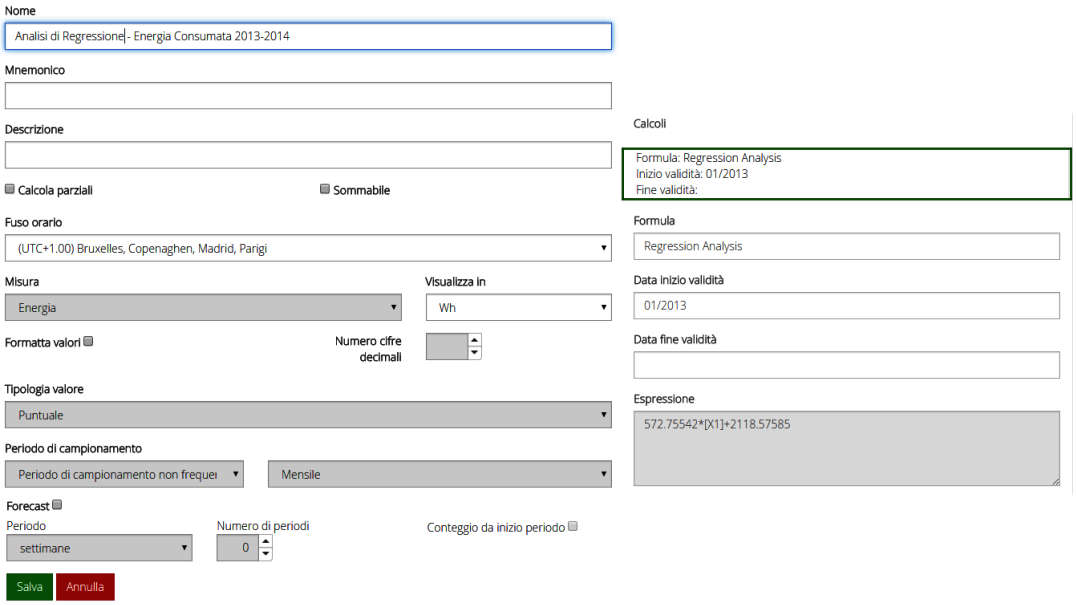


Figura 5.8: Passo 4: Salvataggio e configurazione indicatore

Come si può vedere molte delle informazioni presenti vengono già valorizzate dalle scelte fatte in fase di costruzione del modello (ad esempio il periodo di campionamento viene recuperato direttamente dalla variabile dipendente o dalle scelte di aggregazione fatte), ma è comunque necessario inserire le ultime informazioni non disponibili, come il nome dell'indicatore ed il periodo di validità dei calcoli.
Questa maschera funge anche da riepilogo, infatti presenta tutte le informazioni e le scelte generate dal modulo di analisi.

5.2.5 Analisi di confronto

L'ultima maschera riguarda l'analisi di confronto tra due serie storiche, questa maschera non è legata alla generazione del modello di stima, ma è co-

munque utile analizzarla. La visualizzazione in esame era già presente nel software On.Energy, ma è stata modificata per permettere una più rapida analisi degli indicatori di stima da parte dell'utente. Quando infatti si decide di visualizzare l'andamento di una stima, la maschera proporrà anche la serie corrispondente alla variabile dipendente per cui è stata generata la stima stessa, in questo modo un utente potrà sempre tenere sottocchio il legame tra le due variabili.

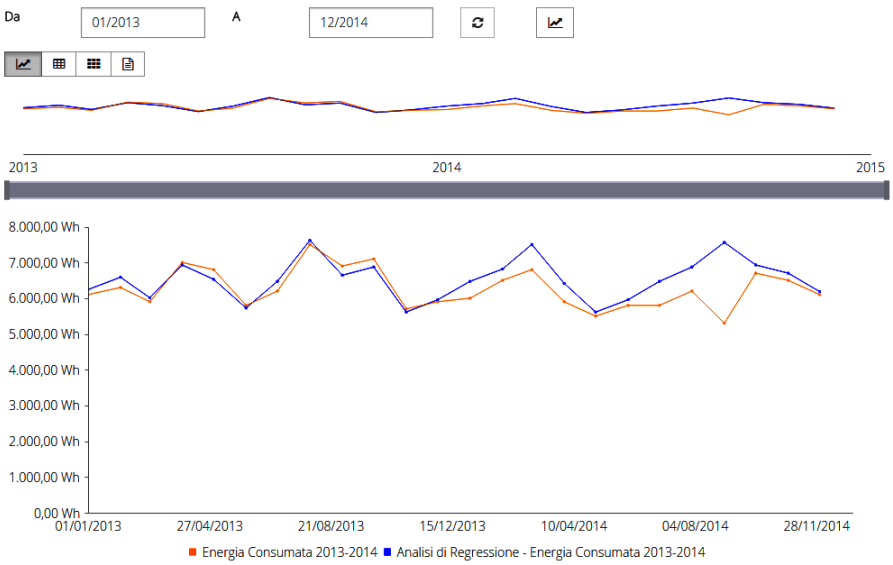


Figura 5.9: Analisi di confronto

Come si può notare dalla figura 5.9 il modello generato per il caso preso ad esempio è abbastanza descrittivo della variabile dipendente, fatta eccezione per un picco nell'intorno di agosto-settembre 2014 in cui il modello ha sovrastimato molto il consumo, probabilmente dovuto al periodo di vacanze estive dell'azienda.

Conclusioni e sviluppi futuri

La progettazione del modulo oggetto di questo lavoro di tesi ha permesso innanzitutto di approfondire le conoscenze dei sistemi SCADA/EMS evidenziando le caratteristiche e le funzionalità fondamentali al loro funzionamento. Questo ha permesso di far emergere alcune lacune del software On.Energy Management System che hanno giustificato l'utilità di un modulo per la creazione di modelli previsionali di variabili energetiche.
L'analisi previsionale avviene per mezzo della regressione lineare, cioè un metodo statistico che mette in relazione diverse variabili per generare un modello. Successivamente viene inserita una nuova variabile nel sistema On.Energy, che consentirà all'utente di eseguire tutte le metodologie di analisi disponibili e verificare dove e quando i consumi si distaccano dal modello teorico. Questa tipologia di analisi fornisce ai manager aziendali dei suggerimenti sugli interventi da eseguire per efficientare l'impianto energetico in esame in base a fattori relativi al contesto.
Per validare l'efficacia dei metodi di modellazione utilizzati si è cercato di effettuare test su valori reali notando che, se le variabili utilizzate sono legate da un buon livello di correlazione, il modello rispecchia in modo corretto i valori utilizzati per costruirlo e quindi può essere considerato un buon mo-

dello previsionale.

Ciò che effettivamente manca al momento per rendere completa l'offerta legata ai modelli previsionali fornita da On.Energy è la possibilità di un'analisi approfondita dei residui, ovvero delle differenze presenti tra il modello ed i dati rilevati, è infatti possibile verificare graficamente e tramite rappresentazione tabulare l'andamento delle due serie mettendole anche a confronto, ma non esiste uno strumento in grado di calcolare in modo automatico queste differenze e fornirne un calcolo sempre aggiornato su cui estrapolare informazioni aggiuntive.

Uno dei possibili sviluppi che potranno sicuramente essere affrontati nel prossimo futuro, e che porterebbero al modulo oggetto della tesi un valore aggiunto, riguarda l'analisi dei residui utilizzando il metodo delle somme cumulate (CUSUM). Questo potenzierebbe le possibilità di tenere sotto controllo i consumi rappresentati dalla variabile stimata, verificando quando le sue differenze con lo stimatore iniziano a divergere e superano una soglia definita, possibilmente generando un allarme che avvisi l'utente della necessità di un intervento.

Bibliografia

[1] `Accord.Net Documentazione`.
https://github.com/accord-net/framework/wiki/Regression.

[2] `ASP.NET MVC Tutorial`.
http://www.w3schools.com/aspnet/mvc_intro.asp.

[3] `ASP.NET Razor - Markup`.
http://www.w3schools.com/aspnet/razor_intro.asp.

[4] `Bootstrap 3 Tutorial`.
http://www.w3schools.com/bootstrap/.

[5] `Code First per un nuovo database`.
https://msdn.microsoft.com/it-it/data/jj193542.aspx.

[6] `Coefficiente di correlazione per ranghi di Spearman`.
http://it.wikipedia.org/wiki/Coefficiente_di_correlazione_per_ranghi_di_Spearman.

[7] `Correlazione (statistica)`.
http://it.wikipedia.org/wiki/Correlazione_(statistica).

[8] `Entity Framework`.
http://en.wikipedia.org/wiki/Entity_Framework.

[9] `Indice di correlazione di Pearson`.
http://it.wikipedia.org/wiki/Indice_di_correlazione_di_Pearson.

[10] `Javascript`.
http://it.wikipedia.org/wiki/JavaScript.

[11] `JQuery UI vs Kendo UI`.
http://jqueryuivskendoui.com/.

[12] `KNX (standard)`.
http://en.wikipedia.org/wiki/KNX_(standard).

[13] `LonWorks`.
http://it.wikipedia.org/wiki/LonWorks.

[14] `Math.Net Documentazione`.
http://numerics.mathdotnet.com/Matrix.html.

[15] `Modbus`.
http://en.wikipedia.org/wiki/Modbus.

[16] `Model-View-Controller`.
http://en.wikipedia.org/wiki/Model-view-controller.

[17] `Panoramica di ASP.NET MVC`.
https://msdn.microsoft.com/it-it/library/dd381412(v=vs.108).aspx.

[18] `RANSAC`.
http://en.wikipedia.org/wiki/RANSAC.

[19] `Regression analysis`.
http://en.wikipedia.org/wiki/Regression_analysis.

[20] `Regressione lineare`.
http://it.wikipedia.org/wiki/Regressione_lineare.

[21] `Relazioni statistiche: regressione e correlazione`.
http://www.sp.units.it/Docenti

[22] `SCADA`.
http://en.wikipedia.org/wiki/SCADA, 2008.

[23] `ISO 50001`.
http://en.wikipedia.org/wiki/ISO_50001, 2014.

[24] W. Salter A. Daneels. `What is SCADA?` *International Conference on Accelerator and Large Experimental Physics Control Systems*, 1999.

[25] Dale Barr and Peter M. Fonash. ***Supervisory Control and Data Acquisition (SCADA) Systems***. Chantilly, Virginia, 2004.

[26] Tao Hong e David A. Dickey. *Electric load forecasting: fundamentals and best practices*. 2010.

[27] Scott Guthrie. Code-First Development with Entity Framework 4. 2010.

[28] Scott Guthrie. Introducing Razor–a new view engine for ASP.NET. 2010.

[29] The Hong Kong Electronic Industries Association (HKEIA). *Guidebook for ISO 50001 Energy Management System*. 2013.

[30] Marco Mamei. L'architettura MVC (Model-View-Controller) - Introduzione.

[31] Giuseppe Marchi. Introduzione a LINQ.

[32] Simone Moretti. Introduzione a Entity Framework.

[33] Australian Government Department of Industry and Science. Establishing an Energy Management System. 2014.

[34] Sandro Paganotti, Gianluca Guarini, Simone Bonati, and Carlo Daniele. Guida HTML5.
http://www.html.it/guide/guida-html5/.

[35] Michele Pellerey. Il concetto di correlazione.

[36] Rich. Preparing for the Internet of Things. *B-SCADA*, 2014.

[37] Margaret Rouse. LINQ (Language Integrated Query). 2008.

[38] Alberto Sabbion. Stima del parametro ''h'' di una carta di controllo CUSUM mediante approssimazione stocastica. 2005.

[39] Onit Group s.r.l. On.Energy.
http://www.onit.it/onenergy/it-it/home.aspx.

[40] Onit Group s.r.l. Onit Home Page.
http://www.onit.it/.

[41] Marco Zuliani. RANSAC for Dummies. 2014.