



UNIVERSITÀ DEGLI STUDI DI FIRENZE
SCUOLA DI INGEGNERIA - DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE

Tesi di Laurea Magistrale in Ingegneria Informatica

**PROGETTAZIONE E SVILUPPO DI COMPONENTI
PER LA PIATTAFORMA AIRQINO DEDICATA AL
MONITORAGGIO DELLA QUALITÀ DELL'ARIA**

Candidato
Edoardo D'Angelis

Relatori
Prof. Andrew D. Bagdanov
Prof. Pietro Pala

Correlatori
Dott. Walter Nunziati
Dott.ssa Alice Cavaliere

Anno Accademico 2020/2021

Abstract

Air pollution is currently one of the main issues affecting urbanized areas worldwide. There is concern regarding the health issues caused by long-term exposure to airborne particulate matter (PM) and other harmful gases (such as NO₂, CO₂ and O₃). Measurements at appropriate spatial and temporal scales are essential for understanding and monitoring air pollution, which is required for the development of real-time strategies for exposure control. Conventional approaches to air quality monitoring are based on networks of static and sparse measurement stations, provided by regional or national environmental protection agencies. These stations, however, have limitations due to coarse spatial coverage of the whole municipality, low time-frequency, and high costs. New low-cost and high-portability sensors, intended to complement the existing solutions, are radically changing the conventional approach by allowing real-time information in a high-density form, with new scalable networks (such as AirQino) providing data at fine spatial and temporal scales.

This thesis, developed in collaboration with Magenta srl and the Institute of BioEconomy (IBE) of CNR, focuses on the AirQino platform and has three main objectives: (i) to improve efficiency and scalability of the system; (ii) to investigate and compare different techniques aimed at improving accuracy of the sensors' calibration process; (iii) to develop a web interface to make batch calibration easier.

Sommario

L'inquinamento atmosferico è uno dei principali problemi che interessano le aree urbanizzate. Tra le conseguenze ci sono i problemi di salute causati dall'esposizione a lungo termine al particolato atmosferico (PM) e ad altri gas nocivi (come NO_2 , CO_2 e O_3). Continue misurazioni sono essenziali per comprendere e monitorare l'inquinamento atmosferico e per lo sviluppo di strategie in tempo reale per il controllo dell'esposizione. Gli approcci convenzionali al monitoraggio della qualità dell'aria si basano su reti di stazioni di misurazione fisse, fornite da agenzie di protezione ambientale regionali. Queste stazioni, tuttavia, presentano limitazioni dovute alla copertura spaziale limitata, alla bassa frequenza temporale e ai costi elevati. Nuovi sensori a basso costo e ad alta portabilità, integrabili con le soluzioni esistenti, stanno cambiando radicalmente l'approccio convenzionale, consentendo l'invio di informazioni in tempo reale ad alta densità, con nuove reti scalabili (come AirQino) che forniscono dati ad alta risoluzione temporale e spaziale.

Questa tesi, sviluppata in collaborazione con Magenta srl e con l'istituto di BioEconomia del CNR (IBE-CNR), è incentrata sulla piattaforma AirQino e si prefigge tre obiettivi principali: (i) migliorare l'efficienza e la scalabilità del sistema; (ii) studiare e confrontare diverse tecniche volte a migliorare l'accuratezza del processo di calibrazione dei sensori; (iii) sviluppare un'interfaccia web per facilitare la calibrazione massiva di centraline.

Indice

Abstract	i
Sommario	ii
1 Introduzione	1
1.1 Contesto	3
1.2 Motivazioni	5
1.3 La piattaforma AirQino	6
1.3.1 Architettura e tecnologie	8
1.3.2 Sensori	10
1.3.2.1 MiCS-2714	11
1.3.2.2 SDS011	12
1.3.3 Progetti che includono la piattaforma AirQino	13
1.3.4 Altre piattaforme	15
1.4 Contributi di questa tesi	15
2 Sviluppi tecnologici	17
2.1 Replica del database di produzione	17
2.1.1 Motivazioni	18
2.1.2 Streaming replication	19
2.1.2.1 Preparazione del database primario	22

2.1.2.2	Configurazione della replica	23
2.1.2.3	Automazione con Docker	24
2.2	Ottimizzazione di query temporali	27
2.2.1	Motivazioni	28
2.2.2	Continuous aggregates	31
2.2.3	Risultati ottenuti	32
3	Calibrazione	36
3.1	Introduzione	37
3.2	Dati a disposizione	38
3.2.1	Dataset NO ₂	40
3.2.2	Dataset PM _{2.5} e PM ₁₀	42
3.2.3	Preprocessamento	44
3.2.3.1	Dataset ARPAT NO ₂	44
3.2.3.2	Dataset ARPAT PM _{2.5} e PM ₁₀	46
3.2.3.3	Dataset SMART16	48
3.2.3.4	Unione dei dataset	50
3.3	Regressione	51
3.3.1	Introduzione	51
3.3.2	Correlazione e coefficiente di determinazione	53
3.3.3	Analisi dei residui	57
3.3.3.1	Distribuzione degli errori	57
3.3.3.2	Indipendenza degli errori	58
3.3.3.3	Omogeneità della varianza dei residui	59
3.3.3.4	Influenza degli outliers	60
3.3.4	Modelli di regressione	61
3.3.4.1	Regressione lineare	61
3.3.4.2	Regressione lineare robusta (Huber)	62

3.3.4.3	Regressione lineare avanzata	64
3.3.4.4	Regressione Ridge	66
3.3.4.5	Regressione Lasso	68
3.3.4.6	Regressione polinomiale	68
3.3.4.7	Regressione con Random Forest	70
3.3.4.8	Regressione con Gradient Boosting	71
3.3.4.9	Regressione con SVR	72
3.3.4.10	Regressione con KernelRidge	73
3.4	Esperimenti e risultati ottenuti	73
3.4.1	Risultati NO ₂	75
3.4.2	Risultati PM _{2,5}	80
3.4.3	Risultati PM ₁₀	85
3.5	Validazione	90
3.6	Discussione	93
4	Interfaccia di calibrazione	96
4.1	Motivazioni	96
4.2	Requisiti	97
4.3	Tecnologie utilizzate	99
4.3.1	Backend	99
4.3.2	Frontend	100
4.4	Sviluppo	100
4.4.1	Interfaccia	100
4.4.2	Autenticazione	105
4.4.3	CI/CD e deploy automatico	107
Conclusioni e sviluppi futuri		111
Bibliografia		114

Capitolo 1

Introduzione

Il lavoro di tesi è stato realizzato in collaborazione con Magenta srl [1] e con l'Istituto per la BioEconomia del Consiglio Nazionale delle Ricerche (IBE CNR) [2].

L'oggetto di studio è la piattaforma **AirQino** [3] per il monitoraggio ambientale ad alta precisione in ambito urbano. Gli obiettivi sono stati molteplici:

- Sviluppi tecnologici alla piattaforma, rivolti a migliorare in un caso l'affidabilità dei dati inviati dai sensori, e nell'altro la gestione del problema della quantità di dati in aumento costante (capitolo 2);
- Studio e analisi del processo di calibrazione delle centraline AirQino, con un confronto quantitativo sulle diverse tecniche utilizzate per la rilevazione di relazioni tra il segnale dei sensori e la stazione di riferimento (capitolo 3);
- Realizzazione di un'interfaccia che permetta la calibrazione di più centraline contemporaneamente (capitolo 4).



Figura 1.1: Magenta srl

Fonte: <https://magentalab.it>



Figura 1.2: CNR - Istituto per la BioEconomia (IBE)

Fonte: <https://www.ibe.cnr.it>



Figura 1.3: La piattaforma AirQino

Fonte: <https://airqino.it>

1.1 Contesto

Il monitoraggio della qualità dell'aria è una delle attività più importanti per la tutela della salute pubblica. La qualità dell'aria può essere influenzata da molte sorgenti di emissione di origine naturale e antropiche, tra cui le automobili, le centrali elettriche, gli impianti di riscaldamento e le fabbriche. Gli effetti dell'inquinamento atmosferico sulla salute sono molteplici e possono essere a breve o a lungo termine. In questo contesto, i dati raccolti dal monitoraggio della qualità dell'aria possono essere utilizzati per valutare l'impatto dell'inquinamento atmosferico sulla salute umana e sull'ambiente, e per identificare le principali fonti di inquinamento ambientale.

I principali inquinanti atmosferici sono l'ossido di zolfo (SO_x), gli ossidi di azoto (NO_x), il monossido di carbonio (CO), l'ozono (O_3) e il particolato atmosferico (PM o polveri sottili). Per il particolato si parla di PM_{10} se le particelle hanno un diametro inferiore o uguale ai $10\mu\text{m}$, e $\text{PM}_{2.5}$ se inferiore o uguale ai $2.5\mu\text{m}$.

Il monitoraggio della qualità dell'aria è disciplinato in Italia dal DLgs. n. 155 del 13/08/2010, che recepisce la direttiva europea 2008/50/CE [4]. Il decreto assegna alle varie ARPA¹ regionali il compito istituzionale di effettuare il monitoraggio della qualità dell'aria e ne prescrive in dettaglio le modalità. Il monitoraggio viene effettuato attraverso stazioni ubicate in siti di campionamento sia fissi che mobili, in base a precisi criteri di esposizione: si distinguono quindi stazioni di fondo (urbane o suburbane), di traffico urbano e industriali. Ad esempio, nella città di Firenze la rete regionale di rilevamento gestita da ARPAT [5] è costituita da 6 stazioni: 2 urbane-traffico, 3 urbane-fondo e una suburbana-fondo. Avendo valore normativo, le misure

¹Agenzia regionale per la protezione ambientale

effettuate devono inoltre rispondere a precisi criteri di accuratezza ed essere sottoposte a precisi processi di validazione. [6]

Queste reti di monitoraggio sono quindi costituite da stazioni dotate di costose strumentazioni, oltretutto con elevati costi di manutenzione; un altro loro limite intrinseco è la rappresentatività strettamente puntuale delle misure (possono fornire misurazioni dettagliate e altamente accurate [7], ma soltanto in posizioni precise e sparse per una determinata città) [8]. Questo rende difficile raccogliere informazioni rappresentative e affidabili per un'intera area urbana e, quindi, formare un quadro più chiaro sull'andamento dell'inquinamento. [8]

In questo scenario si inseriscono i nuovi sensori *low-cost* per il monitoraggio dell'inquinamento atmosferico integrativo, soprattutto in situazioni in cui i sistemi di monitoraggio tradizionali risultano poco pratici [9]. Nell'acquisizione delle misure, infatti, i sensori di nuova generazione prevedono, a fronte di una minore accuratezza di misura, una serie di vantaggi quali: il basso costo di installazione, l'altra risoluzione temporale del dato acquisito e l'alta rappresentatività spaziale data dalla presenza capillare di più punti di acquisizione. [6]. Oltre a fornire una rete più diffusa in grado di catturare efficacemente la variabilità dell'inquinamento atmosferico, il dispiegamento di sensori a basso costo in numero significativo può anche aiutare a valutare l'esposizione in tempo reale in modo da progettare strategie di mitigazione. [10]

La stessa direttiva europea 2008/50/CE [4] ha legittimato l'utilizzo di tali sensori, riconoscendo e regolamentando l'acquisizione di misure aggiuntive, anche a minor costo e minor precisione, ma che permettano di avere un quadro più completo (sia nello spazio che nel tempo) della qualità dell'aria. [6]

1.2 Motivazioni

Ci sono alcuni inconvenienti da tenere conto per quanto riguarda l'utilizzo di sensori a basso costo. Questi includono:

- Una minore accuratezza nelle misurazioni rispetto alle tradizionali stazioni di monitoraggio; [11]
- Sensibilità a fattori ambientali come temperatura, umidità e pressione;
- Derive del sensore, dovute all'usura dei componenti hardware, che richiedono frequenti ricalibrazioni [12];

La necessità di effettuare calibrazioni frequenti dei sensori implica la definizione di un **processo di calibrazione** che risulti il più possibile accurato ed efficiente. Inoltre alcuni tipi di sensori non forniscono output in unità ingegneristica, per cui la fase di calibrazione risulta obbligatoria.

Allo stesso tempo, un rete di monitoraggio ambientale deve fare i conti con la raccolta e l'analisi di grandi quantità di dati provenienti dai sensori in tempo reale. Su questo aspetto, uno degli obiettivi fondamentali per un sistema di questo tipo riguarda la necessità di garantire sempre **affidabilità** e **disponibilità** dei dati. Inoltre, dal punto di vista tecnologico si deve prevedere l'utilizzo di soluzioni che permettano al sistema di **scalare** facilmente in caso di crescita del numero di stazioni e di sensori. In generale si tratta di problemi difficili da affrontare, sia per quantità di dati che per affidabilità.

Il lavoro svolto si colloca proprio in questo contesto, con il duplice obiettivo di (i) studiare, analizzare e confrontare soluzioni diverse per il processo di calibrazione di centraline e (ii) aumentare sia la scalabilità del sistema che l'affidabilità e la disponibilità dei dati inviati dai sensori.

1.3 La piattaforma AirQino

AirQino è una piattaforma di monitoraggio ambientale ad alta precisione, realizzata dal Consiglio Nazionale delle Ricerche (CNR) in collaborazione con TEA Group e Quanta Srl. [13] Il progetto nasce dall'esigenza di realizzare una rete di stazioni mobile per un monitoraggio più completo della qualità dell'aria in ambito urbano, in linea con la direttiva europea 2008/50/EC [4], che riconosce e regolamenta l'importanza di misure aggiuntive rispetto a quelle delle stazioni fisse.

Alcune delle caratteristiche di AirQino sono:

- Sistema di rilevamento **polifunzionale**: il sistema offre la possibilità di rilevare gli agenti inquinanti presenti in atmosfera;
- Sistema **versatile** e dispiegabile in più punti per creare una rete di monitoraggio capillare, flessibile ed economica;
- Alte **prestazioni** dei sensori installati, garantite da una rigorosa calibrazione e validazione degli apparati da parte dei laboratori del CNR;
- Tutte le stazioni sono **configurabili** con un'ampia gamma di sensori aggiuntivi a seconda delle proprie esigenze. [3]

La rete di sensori AirQino consente rilevare le principali sostanze inquinanti riscontrabili nell'aria (NO_2 , O_3 , CO , $\text{PM}_{2.5}$, PM_{10}) ma anche ulteriori parametri ambientali, come la temperatura, l'umidità relativa dell'aria e il principale gas climalterante, la CO_2 . Inoltre la centralina è **estendibile**, e permette di aggiungere ulteriori sensori ausiliari in base alle necessità.



Figura 1.4: Una centralina AirQino

Fonte: <https://airqino.it>

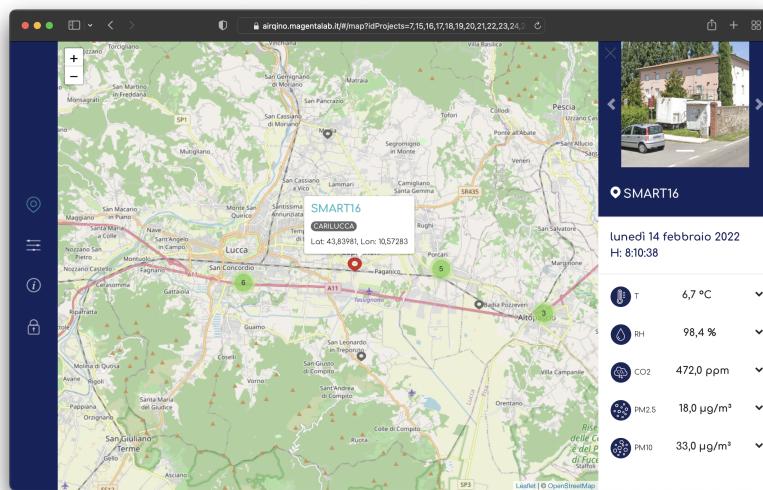


Figura 1.5: La piattaforma web di AirQino

Fonte: <https://airqino.magentaLab.it>

La piattaforma AirQino mette inoltre a disposizione un portale web per la consultazione dei dati, all’indirizzo <https://airqino.magenta-lab.it>.

La piattaforma consiste in una mappa interattiva che visualizza tutte le reti di centraline. Selezionata una stazione di interesse, vengono mostrati dettagli e foto della stazione stessa (figura 1.5). Per ciascun sensore della centralina selezionata è possibile visualizzare grafici di andamento medio settimanali, insieme al dato istantaneo relativo all’ultima misurazione, nell’unità di misura come da normativa. [3]

1.3.1 Architettura e tecnologie

L’architettura del sistema AirQino è organizzata nei seguenti elementi principali (1.6):

- **Gateway:** server che espone servizi compatibili con le centraline, e normalizza i dati trasmessi verso il server di raccolta. Questa applicazione, realizzata con Java e framework **Spring** [14], fornisce un *endpoint* con lo stesso indirizzo a cui le centraline comunicano, in modo da garantire continuità di servizio. Il gateway ha anche il compito di segnalare eventuali interruzioni di funzionamento, secondo regole ben definite. Il gateway produce anche un output su protocollo *MQTT2*, utilizzando un broker open source per pubblicare i dati verso il backend;
- **Backend:** applicazione server che si interfaccia con il broker *MQTT* e scrive sul database i dati, esponendo inoltre servizi web di tipo REST utilizzati dal frontend web. Il backend è realizzato con Java e framework **Spring** [14] e utilizza **Timescale** [15], un database relazionale open source per la gestione efficienti di dati temporali;

- **Frontend:** applicazione web che permette la visualizzazione di mappa e grafici dei dati raccolti dalle centraline, utilizzando i servizi esposti dal backend. L’interfaccia è basata su tecnologia **Angular** [16]. Il frontend ha anche una sezione *admin*, protetta da autenticazione, che permette la gestione del sistema. Le funzionalità di gestione previste sono:
 - Gestione anagrafica centraline (nome, posizione, progetti a cui afferisce);
 - Configurazione dei parametri di funzionamento, tra cui i parametri di calibrazione (per la trasformazione da dati grezzi a valori leggibili dall’utente) e le soglie di rilevamento allarmi;
 - Gestione utenti;
 - Scaricamento di dati *raw* oppure calibrati. [3]

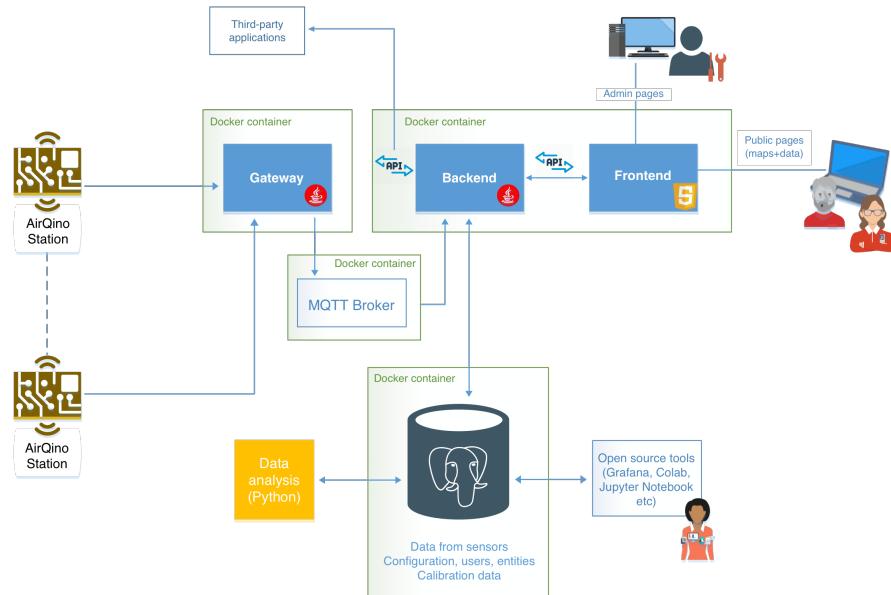


Figura 1.6: Schema architetturale della piattaforma AirQino

1.3.2 Sensori

La tabella 1.1 mostra le varie tipologie di sensori che sono in dotazione con le centraline AirQino, con il rispettivo principio di funzionamento, range e precisione.

Sensori	Tipologia	Unità	Range	Prec.
NO ₂	Sensori di gas MOS	µg/m ³	0-5000	15%
O ₃	Semiconduttore	µg/m ³	0-1000	15%
CO	Sensori di gas MOS	µg/m ³	0-30	15%
VOC totali	Sensori di gas MOS	µg/m ³	0-1000	15%
CO ₂	NDIR	ppm	0-2000	10%
PM _{2.5}	Contatore di particelle ottico	µg/m ³	0-1000	10%
PM ₁₀	Contatore di particelle ottico	µg/m ³	0-1000	10%
Umidità relativa	Stato solido	%	0-100	5%
Temp. dell'aria	Stato solido	°C	-40 - 80	5%
Temp. interna	Stato solido	°C	-40 - 80	5%

Tabella 1.1: Tipologie di sensori in dotazione con le centraline AirQino (configurazione base, estendibile). Fonte: <https://airqino.it>

Nello specifico i sensori utilizzati sono:

- **SenseAir S8** per acquisizione CO₂;
- **DHT22** per umidità relativa e temperatura;
- **MiCS-2614** per O₃;
- **MiCS-2714** per NO₂;
- **TGS-2600** per VOC;
- **SDS011** per PM_{2.5} e PM₁₀. [6]

Oggetto di questo studio sono stati due sensori con principi di funzionamento diversi: MiCS-2714 (1.3.2.1) per NO_2 e SDS011 (1.3.2.2) per $\text{PM}_{2.5}$ e PM_{10} .

1.3.2.1 MiCS-2714

Il MiCS-2714 è un sensore di tipo MOS, ovvero costituito da un *film* depositato su una piastra di elementi riscaldanti la cui temperatura operativa è generalmente compresa tra 300°C e 500°C . [6]

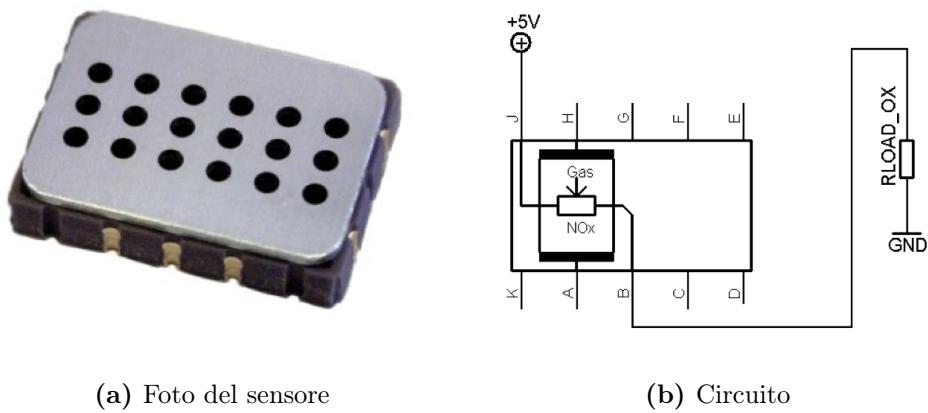


Figura 1.7: Il sensore MiCS-2714 per il rilevamento di NO_2

Fonte: <https://mouser.it>

Di solito il materiale funzionale del *film* più adatto per la rilevazione di NO_2 è l'ossido di ferro e lantanio (LaFeO_3), che oltre ad avere una buona sensibilità agli ossidi di azoto ha una bassa sensibilità al monossido di carbonio. Questo tipo di materiale funzionale risulta molto sensibile ai gas ossidanti, come O_3 e NO_2 . Qualsiasi sia il materiale funzionale, il principio di funzionamento per tutti i MOS nella rilevazione di gas è quello di interagire con il gas presente all'interno dell'atmosfera tramite reazioni di ossidoriduzione, portando a un cambiamento di conduttività, che viene rilevato da un circuito

apposito. Le variazioni della conduttività dei sensori è fortemente influenzata dalle variazioni di umidità e temperatura [6]. Di seguito le specifiche del sensore MiCS-2714 come riportate dal *datasheet*:

Parametro	Specifiche
Range di misura	0.05 – 10ppm
Tensione di alimentazione	4.9-5.1V
Ciclo di vita	>2 anni
Dimensioni	12×16mm
Peso	4g

Tabella 1.2: Specifiche tecniche del sensore MiCS-2714

Fonte: <https://mouser.it>

1.3.2.2 SDS011

Il sensore SDS011 basa il suo funzionamento sul principio della diffusione ottica (detta anche dispersione o *scattering*). In particolare, un raggio luminoso che procede in una certa direzione, quando colpisce una particella, viene diffuso in maniera disordinata in tutte le direzioni. [17]

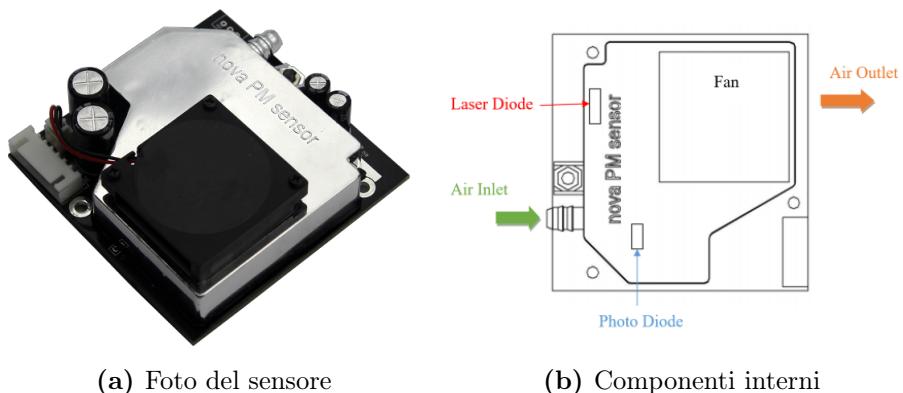


Figura 1.8: Il sensore SDS011 per il rilevamento di $\text{PM}_{2.5}$ e PM_{10}

Fonte: <https://circuitdigest.com>

Il sensore è costituito da una ventola che aspira l'aria esterna per convogliarla in una camera di misura stagna. All'interno della camera di misura è contenuto un diodo laser e un fotodiodo rilevatore (figura 1.8 (b)). Appena l'aria viene immessa nella camera, il laser viene acceso: un fascio di luce laser procede “dritto” senza diffondersi. Quando il laser colpisce il particolato, la sua luce viene diffusa e l'intensità luminosa che si disperde nella camera può essere rilevata dal fotodiodo. Il sensore riesce a distinguere tra il PM₁₀ e il PM_{2.5} perché la forma d'onda della luce rilevata dal fotodiodo è correlata con il numero e le dimensioni delle particelle: per questo motivo a bordo del sensore c'è un microcontrollore ad 8 bit che esegue l'analisi [17]. Di seguito le specifiche del sensore SDS011 come riportate dal *datasheet*:

Parametro	Specifiche
Range di misura	0-1000 µg/m ³
Tensione di alimentazione	4.7-5.3V
Ciclo di vita	11 mesi
Dimensioni	71×70mm
Peso	6g

Tabella 1.3: Specifiche tecniche del sensore SDS011

Fonte: <https://reichelt.com>

1.3.3 Progetti che includono la piattaforma AirQino

Di seguito sono elencati alcuni progetti che fanno uso delle centraline AirQino:

- **Prato Urban Jungle** [18] mira a promuovere la progettazione urbana creativa e visionaria per ri-naturalizzare i quartieri di Prato in modo sostenibile e socialmente inclusivo;

- **SMART Treedom** [19] è il frutto dalla collaborazione tra Treedom e l’Istituto di Biometeorologia del Consiglio Nazionale delle Ricerche. La finalità del progetto è stata quella di prototipare un sistema integrato che possa essere modulato con diversi sensori in base al tipo di grandezza fisica che si vuole misurare e una tecnologia laser per la misura delle polveri sottili;
- **Trafair** [20] è un progetto europeo biennale co-finanziato dal programma europeo Connecting Europe Facility (CEF) nel settore delle telecomunicazioni con lo scopo di sviluppare un servizio di previsione della qualità dell’aria urbana basata su previsioni meteo e flussi di traffico in sei città europee di dimensioni diverse: Zaragoza, Firenze, Modena, Livorno, Santiago de Compostela e Pisa;
- **Smart Garda Lake** [21] nasce nel 2017 con lo scopo di creare una rete di monitoraggio ambientale per rilevamenti in campo meteorologico, inquinamento acustico, stato delle acque superficiali e qualità dell’aria all’insedia degli obiettivi di sostenibilità dell’Agenda 2030 dell’ONU;
- **PlanetWatch** [22] è una piattaforma decentralizzata che consente di monitorare e proteggere il pianeta attraverso la condivisione di informazioni. Gli utenti possono condividere informazioni sull’ambiente, la sostenibilità e la responsabilità sociale;
- **Brenner LEC** [23] si colloca nel contesto di un’area sensibile come le Alpi e si pone l’obiettivo di creare un “corridoio a emissioni ridotte” (LEC – Lower Emission Corridor) lungo l’asse autostradale del Brennero al fine di ottenere un chiaro beneficio ambientale nei settori della tutela dell’aria e della protezione del clima, nonché una riduzione dell’inquinamento acustico.

1.3.4 Altre piattaforme

Esistono anche altre piattaforme con obiettivi simili ad AirQino:

- **Airly** [24] è una piattaforma che consente di condividere informazioni ambientali in tempo reale, grazie alla quale è possibile monitorare la qualità dell'aria e i livelli di inquinamento;
- **Aqicn** [25] è un progetto open source lanciato nel 2010 che consente di monitorare l'inquinamento atmosferico in tempo reale;
- **IQAir** [26] è una società svizzera che produce e vende purificatori d'aria per uso residenziale e commerciale. La loro applicazione fornisce un rapporto in tempo reale sulla qualità dell'aria e previsione dell'inquinamento atmosferico;
- **Decentlab** [27] è un'azienda svizzera che fornisce dispositivi e servizi di sensori wireless per soluzioni di monitoraggio distribuite ed economiche;
- **HackAIR** [28] è una piattaforma open source che consente ai cittadini di monitorare la qualità dell'aria nei propri quartieri. Gli utenti possono interagire con la piattaforma per segnalare e visualizzare i dati relativi alla qualità dell'aria, e condividere informazioni e dati con altri utenti.

1.4 Contributi di questa tesi

Questo lavoro di tesi ha contribuito ai seguenti sviluppi e risultati scientifici:

- Creazione di una *replica* del database di produzione AirQino tramite la funzionalità di **streaming replication**, con lo scopo di migliorare

l'affidabilità dei dati provenienti dai sensori e alleggerire il carico dal database primario in caso di query particolarmente onerose o test di performance;

- Riduzione significativa dei tempi di risposta del database di AirQino, tramite l'implementazione della funzionalità di **continuous aggregates**, per alcuni query specifiche finalizzate a ricavare la media oraria delle misurazioni dei sensori nell'ultima settimana, con conseguente miglioramento nelle prestazioni della piattaforma;
- Analisi e confronto tra le performance di diversi modelli di **regressione** (sia lineare che non lineare) applicati alle misurazioni dei sensori AirQino di NO₂, PM_{2.5} e PM₁₀ confrontati con la stazione ARPAT di riferimento, con individuazione del modello più adatto per la calibrazione del segnale e per il miglioramento della predizione su dati futuri;
- Realizzazione di un'**interfaccia web** autenticata con lo scopo di semplificare la calibrazione di più centraline AirQino contemporaneamente.

Nel prossimo capitolo vengono introdotti gli sviluppi tecnologici effettuati sulla piattaforma AirQino. Nel capitolo 3 viene presentato il lavoro svolto sulla calibrazione delle centraline, i risultati scientifici ottenuti dai modelli di regressione e la validazione sul campo del modello più performante. Infine, nel capitolo 4 viene presentato il processo di sviluppo e realizzazione dell'interfaccia web per la calibrazione multipla di centraline AirQino.

Capitolo 2

Sviluppi tecnologici

Questo capitolo riguarda gli sviluppi realizzati dal punto di vista tecnologico che sono andati direttamente ad impattare la piattaforma AirQino, migliorandone in un caso l'affidabilità dei dati e nell'altro i tempi di risposta del database per query particolarmente onerose.

2.1 Replica del database di produzione

Spesso fare analisi mediamente complesse sui dati contenuti in un database può comportare rallentamenti nei tempi di risposta. Se questi carichi risultano frequenti, il sistema può arrivare a bloccarsi e provocare interruzioni del servizio. Una soluzione per risolvere questo problema è la creazione di una (o più) **repliche** del database primario. Nella replica, i dati e gli oggetti del database vengono copiati e distribuiti su un altro spazio fisico. Le operazioni onerose a questo punto possono essere fatte direttamente sulla replica che agisce come nodo secondario: in questo modo, il carico viene distribuito e non si intaccano le performance del database principale.

Il concetto di *replica* è diverso dal *mirroring*, in cui vengono create una o più copie di un database su diverse istanze del server, e funzionano come copie di riserva (e si attivano soltanto nel caso di guasto del nodo principale).

Un sistema di replica correttamente implementato può offrire diversi vantaggi, tra cui **riduzione del carico** (perché i dati replicati possono essere distribuiti su più server), **efficienza** (i server offrono prestazioni migliori perché meno gravati da query pesanti) e **ridondanza** (i dati sono raggiungibili da più indirizzi).

Di contro, questa tecnica comporta la necessità di mantenimento dei nodi secondari, spesso collocati su server diversi (con i costi a questi associati). Inoltre, repliche errate o non implementate in maniera corretta possono causare la mancata sincronizzazione tra i nodi, portando ad una perdita o incoerenza dei dati.

2.1.1 Motivazioni

L'affidabilità dei dati rappresenta uno dei punti critici per un sistema di ingestione di grosse quantità di dati. Questo è vero anche caso di AirQino, dove il database di produzione conta oltre 100 milioni di misurazioni rilevate, in continuo aumento, con una media di 300 inserimenti al minuto. In questo scenario la replicazione dei dati può portare diversi vantaggi, principalmente legati alle prestazioni, disponibilità e sicurezza dei dati:

1. **Maggiore affidabilità:** tramite la replica del database viene garantita la disponibilità dei dati anche nel caso in cui una delle macchine presenti un guasto hardware;
2. **Miglioramento delle prestazioni:** essendo i dati distribuiti su diverse istanze, accessi multipli non saturano i server. Questo aspetto risulta

particolarmente importante per applicazioni con una grande quantità di richieste simultanee;

3. **Maggiore sicurezza dei dati:** Mentre in un sistema tradizionale i backup di un database (se effettuati) sono archiviati sullo stesso disco, con la replica del database possono essere distribuiti su più server, aumentandone di fatto l'affidabilità e la ridondanza.

Esistono diverse tecniche di replicazione del database, che dipendono sia dalla tecnologia utilizzata (MySQL [29], Postgres [30]) che dalla natura del database stesso (relazionale o non relazionale). Il database di AirQino fa uso di **Timescale** [15], un database relazionale open source per la gestione efficienti di dati temporali basato su Postgres; una caratteristica di Postgres è proprio la possibilità di replicazione tramite l'utilizzo della tecnologia di **streaming replication**, descritta di seguito.

2.1.2 Streaming replication

La *streaming replication* è una funzionalità che consente di replicare i dati in tempo reale da una istanza di database Postgres a un'altra. Questo significa che se si modificano i dati in una delle istanze, questi saranno immediatamente replicati anche nelle altre. Questa replica di lettura (*standby* in termini Postgres) è di fatto una copia fisica creata in modo asincrono dell'istanza del database primario.

Postgres fa uso di un ruolo speciale detto di replica (*replication role*) per eseguire la replica in streaming. Questo presenta dei privilegi ma non può essere utilizzato per modificare i dati: la replica infatti è di **sola lettura**.

La gestione della replicazione si basa sulle transazioni **WAL** (*Write Ahead Log*) e utilizza il protocollo TCP per garantire una connessione sicura tra i

server, e inviare in modo asincrono le modifiche al database via via che vengono effettuate. Postgres salva le informazioni aggiornate del server primario in registro delle transazioni, noto come registro *write-ahead*, utile anche in fase di preparazione per il ripristino dopo un'interruzione o per un *rollback*. La replica in streaming funziona proprio trasferendo e applicando il registro al server di replica in tempo reale (figura 2.1).

È possibile anche promuovere una replica di lettura Postgres a una nuova istanza database di origine. Una volta promossa, la replica smette di ricevere comunicazioni WAL e non è più un'istanza di sola lettura, ma agisce di fatto come nodo primario.

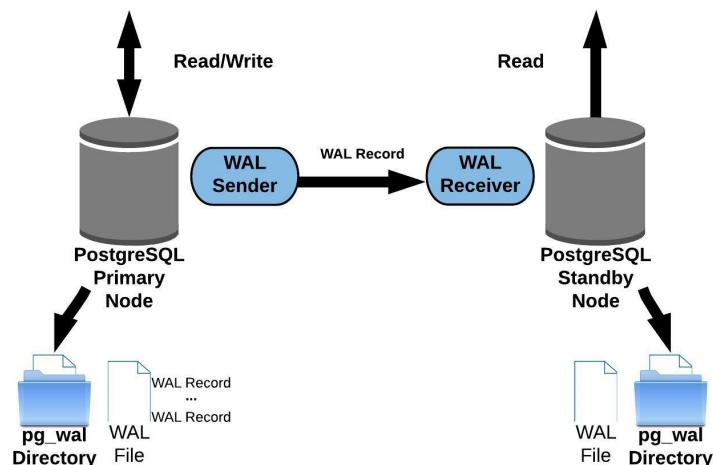


Figura 2.1: Streaming Replication di Postgres

Fonte: <https://severalnines.com>

La replica in streaming può essere costruita in una configurazione 1:N, con un solo server primario, ma è possibile anche aggiungere più server di replica (configurazione *multi-standby*). È anche possibile realizzare una configurazione *a cascata*, in cui un server di replica si connette ad un altro server di replica.

Per la replica in streaming, è possibile scegliere se effettuare una modalità *sincrona* oppure *asincrona*. La differenza tra replica sincrona e replica asincrona sta nell'attesa o meno della risposta dalla replica prima di completare l'elaborazione sul server primario. In particolare:

- **Replica sincrona:** il server primario attende una risposta dal server di replica prima di completare un processo. In questo caso il tempo di risposta complessivo include anche il tempo di spedizione del registro WAL;
- **Replica asincrona** (impostazione predefinita): il server primario completa un processo senza attendere una risposta dal server di standby. Pertanto, il tempo di risposta complessivo è praticamente lo stesso di quando non viene utilizzata la replica in streaming; in questa modalità il risultato aggiornato sul server primario potrebbe non essere immediatamente disponibile sul server di replica. [31]

Ci sono tuttavia alcune limitazioni nell'utilizzo di repliche con Postgres:

- Ogni replica è di sola lettura. Non è possibile creare una replica che sia anche scrivibile;
- Non è possibile creare una replica di lettura da un'altra replica di lettura a cascata;
- Gli utenti e i ruoli di accesso vengono rispecchiati dall'istanza primaria, il che significa che non è possibile utilizzare credenziali diverse o aggiungere utenti soltanto alla replica;
- Non è possibile creare tabelle e viste nell'istanza di replica; in pratica si possono solo eseguire query di tipo *SELECT*.

La sezioni seguenti spiegano il lavoro svolto per configurare la replica del database di produzione della piattaforma AirQino.

2.1.2.1 Preparazione del database primario

1. Per avviare la procedura, è stato aggiunto sul database un utente PostgreSQL, con un ruolo adatto ad avviare la *streaming replication*, con i comandi SQL:

```
1 SET password_encryption = 'scram-sha-256';
2 CREATE ROLE repuser WITH REPLICATION PASSWORD 'SOME_SECURE_PASSWORD' LOGIN;
```

Dove *repuser* è il nome dell’utente di replicazione e *SOME_SECURE_PASSWORD* è la password di replicazione;

2. Sono stati aggiunti i seguenti parametri di configurazione al file */var/lib/postgresql/data/postgresql.conf*:

```
1 listen_addresses='*'
2 wal_level = replica
3 max_wal_senders = 2
4 max_replication_slots = 2
5 synchronous_commit = off
```

Questi sono i parametri adatti ad una configurazione a replica singola: in caso di più repliche sarà necessario aumentare il valore di *max_wal_senders* e *max_replication_slots* [32];

3. Sono stati aggiunti i seguenti parametri al file */var/lib/postgresql/data/pg_hba.conf* per configurare l’autenticazione basata su host, in modo da accettare connessioni dalla replica:

```
1 host replication repuser <REPLICA_IP>/32 scram-sha-256
```

Dove *repuser* è l'utente creato al passo 1 e *REPLICA_IP* è l'IP della macchina in cui si trova la replica [32];

4. È stato riavviato il database per applicare i cambiamenti;
5. Infine è stato creato uno *slot di replicazione* sul database con il comando:

```
1 SELECT * FROM pg_create_physical_replication_slot('replica_1_slot');
```

Questo assicura che il server master conservi i registri WAL necessari per le replicate anche quando queste sono disconnesse dal master.

2.1.2.2 Configurazione della replica

Di seguito sono elencati i passi eseguiti per configurare e attivare la replica su un server secondario:

1. È stata interrotta l'istanza Postgres (se attiva) con il comando:

```
1 pg_ctl -D $PGDATA -m fast -w stop
```

2. Sono stati cancellati i contenuti della cartella *PGDATA*:

```
1 rm -rf $PGDATA/*
```

3. È stato avviato il backup del database primario:

```
1 pg_basebackup -h <PRIMARY_HOST> -p <PRIMARY_PORT> -D
$PGDATA -U repuser -vP -R -W
```

Dove *PRIMARY_HOST* è l'IP della macchina in cui si trova il database primario, *PRIMARY_PORT* è la porta del database primario e *repuser* è l'utente di replicazione creato al passo 1.

Da notare che con la *flag* *-W* viene chiesta in maniera interattiva la password di replicazione, anch'essa impostata in precedenza al passo 1;

4. Infine è stata riavviata l'istanza Postgres:

```
1 pg_ctl -D $PGDATA -w start
```

A questo punto la replica è attiva e sincronizzata 1:1 in tempo reale con il database primario. Per il database di produzione AirQino, questa procedura ha richiesto circa 5 minuti.

2.1.2.3 Automazione con Docker

L'intero setup è stato poi automatizzato con Docker [33] e docker-compose, in modo da avviare la sincronizzazione direttamente all'avvio del container, rendendo la procedura molto più semplice da gestire.

Poichè la creazione della replica richiede l'interruzione dell'istanza Postgres, questa procedura non si può eseguire all'interno del container Docker stesso, perché si andrebbe a interrompere tutto il container. Per questo è stato necessario aggiungere i comandi per la replica in uno script di *entrypoint*, che viene eseguito subito prima di avviare il container [34]:

1. È stato creato un **Dockerfile** partendo dall'immagine Timescale ufficiale (**timescaledb:latest**), con l'aggiunta di uno script *entrypoint*, in questo modo:

```
1 FROM timescale/timescaledb:latest-pg13
2 ADD replica.sh /docker-entrypoint-initdb.d/
```

Dove `docker-entrypoint-initdb.d` è una cartella speciale messa che serve proprio a garantire l'esecuzione di qualsiasi file si trovi al suo interno al momento dell'avvio del database;

2. È stato creato il file `replica.sh`, eseguito tutte le volte che si avvia il database, con il seguente contenuto:

```
1 echo "Stopping Postgres instance..."  
2 pg_ctl -D ${PGDATA} -m fast -w stop  
3  
4 echo "Clearing PGDATA folder..."  
5 rm -rf ${PGDATA}  
6  
7 echo "Creating base backup..."  
8 PGPASSWORD=${REPLICATION_PASSWORD} pg_basebackup -h ${  
         REPLICATION_HOST} -p ${REPLICATION_PORT} -D ${PGDATA}  
         } -U ${REPLICATION_USER} -vP -R -w  
9  
10 echo "Restarting Postgres instance..."  
11 pg_ctl -D ${PGDATA} -w start
```

Il file riproduce semplicemente i passi descritti in 2.1.2.2.

Come già accennato, la flag `-W` di `pg_basebackup` chiede la password in maniera interattiva, ma questo ovviamente non è adatto a script automatizzati che invece non richiedono interazione. Come alternativa è stata utilizzata la flag `-w` (minuscola) e passata la password come una variabile di ambiente (`PGPASSWORD`);

3. È stato creato il file `docker-compose.yml` così strutturato (ridotto per semplicità):

```
1 services:
```

```
2    replica:
3
4    build:
5        context: .
6        dockerfile: Dockerfile
7
8    environment:
9        PGDATA: /var/lib/postgresql/data/pgdata
10
11       # Parametri di replicazione
12       REPLICA_USER: repuser # Utente di replicazione impostato al punto 1
13       REPLICATION_HOST: x.x.x.x # IP del db primario
14       REPLICATION_PORT: x # Porta del db primario
15       REPLICATION_PASSWORD: SOME_SECURE_PASSWORD #
16           Password di replicazione impostata al punto 1
17
18    ports:
19        - 45432:5432
20
21    volumes:
22        - /var/replica-pg13-timescale/:/var/lib/postgresql/data
```

Questo file semplicemente costruisce il container leggendo le istruzioni dal **Dockerfile** del passo 1, e applicando le varabili di ambiente specificate nel campo **environment**. Ulteriori parametri possono essere specificati, come ad esempio la porta sulla quale viene esposto il database (campo **ports**) e la cartella dell'host che viene mappata come volume di dati nel container (campo **volumes**);

4. Infine, è stato avviato il container con **docker-compose up**.

In questo modo la replica viene sincronizzata con il database primario automaticamente all'avvio del container Docker (figura 2.2).

```
streaming-replication-docker-timescale-primary-1 streaming-replication-docker-timescale-primary-1 streaming-replication-docker-timescale-primary-1 streaming-replication-docker-timescale-primary-1 by gcc (Alpine 10.3.1_l1_g12021042) 10.3.1_j20210420_12:00:00+0000
streaming-replication-docker-timescale-primary-1 PostgreSQL init process complete; ready for start up.
2021-10-26 15:01:08.856 UTC [1] LOG:  starting PostgreSQL 13.4 on aarch64-unknown-linux-musl, compiled by gcc (Alpine 10.3.1_l1_g12021042) 10.3.1_j20210420_12:00:00+0000
2021-10-26 15:01:08.856 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
2021-10-26 15:01:08.856 UTC [1] LOG:  listening on IPv6 address "::", port 5432
2021-10-26 15:01:08.856 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2021-10-26 15:01:08.856 UTC [1] LOG:  database system is ready to accept connections
2021-10-26 15:01:08.856 UTC [1] LOG:  TimescaledB background worker launcher connected to shared catalog
logs

Retrying backup...
pg_basebackup: initializing base backup, waiting for checkpoint to complete
pg_basebackup: checkpoint completed
pg_basebackup: waiting for log start point: 0/2000028 on timeline 1
pg_basebackup: starting streaming WAL receiver
pg_basebackup: created temporary replication slot "pg_basebackup_108"
pg_basebackup: 0/26899 KB (0%), 0/1 tablespace (.../pgsql/data/pgbasebackup_label)
pg_basebackup: waiting for log end point: 0/2000100
2A895/26899 KB (19%), 1/1 tablespace
pg_basebackup: write-ahead log end point: 0/2000100
pg_basebackup: waiting for background process to finish streaming ...
pg_basebackup: 0/26899 KB (19%), 1/1 tablespace
pg_basebackup: remaining backup manifest.tmp to backup_manifest
pg_basebackup: base backup completed
waiting for server to start... 2021-10-26 15:01:10.298 UTC [97] LOG:  starting PostgreSQL 13.4 on aarch64-unknown-linux-musl
streaming-replication-docker-timescale-primary-1
2021-10-26 15:01:10.298 UTC [97] LOG:  listening on IPv4 address "0.0.0.0", port 5432
streaming-replication-docker-timescale-primary-1
2021-10-26 15:01:10.299 UTC [97] LOG:  listening on IPv6 address "::", port 5432
streaming-replication-docker-timescale-primary-1
2021-10-26 15:01:10.299 UTC [97] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2021-10-26 15:01:10.299 UTC [97] LOG:  database system was unpaused; last known up at 2021-10-26 15:01:08 UTC
streaming-replication-docker-timescale-primary-1
2021-10-26 15:01:10.415 UTC [98] LOG:  entering standby mode
streaming-replication-docker-timescale-primary-1
2021-10-26 15:01:10.415 UTC [98] LOG:  waiting for recovery to complete
streaming-replication-docker-timescale-primary-1
2021-10-26 15:01:10.416 UTC [98] LOG:  consistent recovery state reached at 0/2000100
streaming-replication-docker-timescale-primary-1
2021-10-26 15:01:10.416 UTC [98] LOG:  database system is ready to accept read only connections
streaming-replication-docker-timescale-primary-1
2021-10-26 15:01:10.422 UTC [102] LOG:  started streaming WAL from primary at 0/3000000 on timeline 1
streaming-replication-docker-timescale-primary-1
server started
```

Figura 2.2: Output del comando `docker-compose up` con sincronizzazione automatica della replica del database primario

2.2 Ottimizzazione di query temporali

Esistono molti vantaggi nell'aggregazione di grandi quantità di dati:

- **Flessibilità:** aggregare dati in tempo reale in base a qualsiasi criterio desiderato;
 - **Risparmio di tempo:** non è necessario eseguire query aggiuntive per ottenere altre informazioni;
 - **Risparmio di spazio:** i dati aggregati in tempo reale occupano meno spazio rispetto ai dati non aggregati.

I dati delle serie temporali (es. misurazioni sensori) tendono a crescere molto rapidamente, e questo può influire sull'esecuzione di query con lo scopo di aggregare i dati (ad esempio per generare report o riepiloghi sull'andamento o sui trend di crescita/decrescita). Per garantire efficienza e scalabilità con questa tipologia di dati, è necessario che le query di aggregazione su dati

temporali abbiano un tempo di risposta costante, indipendentemente dalla quantità di dati e senza gravare in maniera eccessiva sul database.

2.2.1 Motivazioni

La piattaforma AirQino (vedi 1.3) raccoglie dati emessi ogni minuto da decine di centraline in diverse località. Per memorizzare questi dati viene utilizzato un database Postgres [30] con estensione Timescale [15], che offre delle funzionalità specifiche proprio per dati di tipo temporale.

Una delle funzionalità della piattaforma AirQino consente di mostrare un grafico dell'andamento (su base media oraria) dell'ultima settimana per diverse grandezze (temperatura, umidità, CO₂, NO₂, PM_{2.5}, PM₁₀ etc...), come mostrato in figura 2.3.

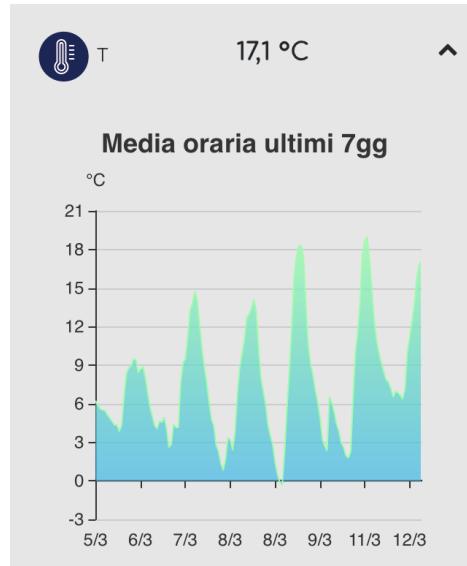


Figura 2.3: Grafico dell'andamento della temperatura nell'ultima settimana per una centralina AirQino.

Fonte: <https://airqino.magenta-lab.it>

La piattaforma prevede inoltre altri casi d'uso per i dati a media oraria:

- Visualizzazione sulla home page dell'applicazione le medie orarie degli ultimi 7 giorni;
- Esportazione dataset periodici, da mettere a disposizione degli utenti esterni (ad esempio pubbliche amministrazioni che hanno installato le centraline);
- Esportazione dataset *on demand* da parte di utenti esterni, con indicazione di inizio/fine del periodo desiderato.

In tutti questi casi il requisito è di ottenere **medie orarie del dato calibrato**.

Per calcolare la media oraria nell'ultima settimana, ad ogni richiesta sul database viene eseguito una query SQL di questo tipo (ridotta per semplicità):

```
1 SELECT time_bucket('1_hour', sd.data_acquired) as bucket, avg(sd.float_value)
2 FROM station_data sd
3 WHERE sd.data_acquired > NOW() - INTERVAL '7_days'
4 AND sd.sensor_id = 29510691 /* id centralina */
5 ORDER BY bucket DESC;
```

Dove *station_data* è la tabella contenente i dati dei sensori, la colonna *data_acquired* rappresenta data e ora di misurazione e la colonna *float_value* indica il valore grezzo della misurazione. I dati vengono limitati solo all'ultima settimana tramite una comparazione su clausola *WHERE*.

La query inoltre fa uso della funzione *time_bucket* di Timescale, che consente di partizionare la tabella con i dati dei sensori minuto per minuto direttamente in fasce orarie; l'utilizzo combinato con la funzione *avg()* consente poi di estrarre la media del valore su tutta l'ora.

Questo però significa che se si desidera eseguire questa query più di una volta, il database deve eseguire la scansione dell'intera tabella e ricalcolare la media ogni volta. Questo risulta poco efficiente, perché nella maggior parte dei casi i dati nella tabella non sono cambiati in modo significativo, quindi non è necessario eseguirne nuovamente la scansione.

La figura 2.4 mostra i tempi di risposta della query per estrarre la media oraria di NO₂ per tutte le centraline AirQino, prima di aver applicato l'ottimizzazione.

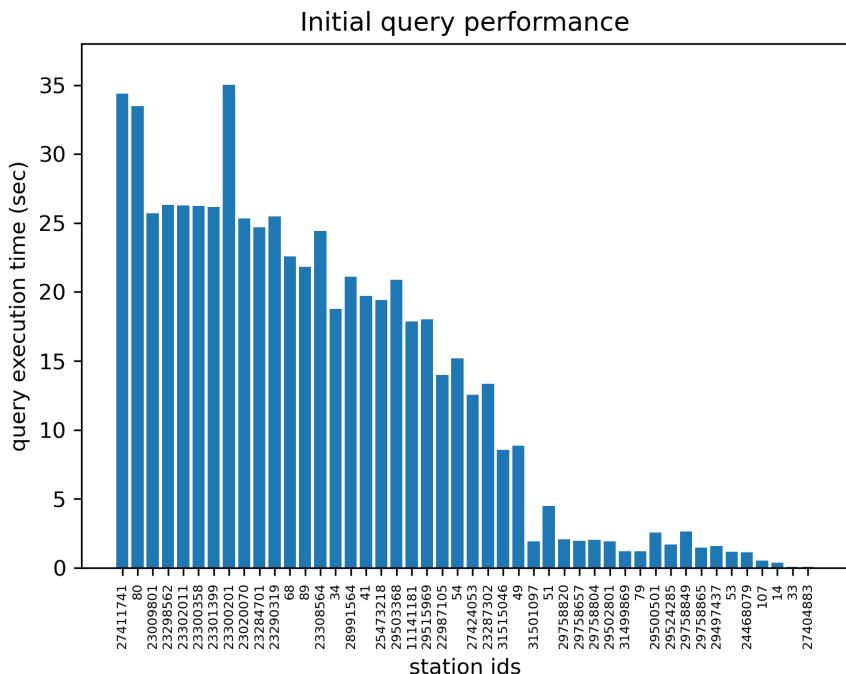


Figura 2.4: Tempi di risposta della query per estrarre la media oraria di NO₂ dell’ultima settimana per tutte le centraline AirQino, prima dell’ottimizzazione (media su 10 iterazioni)

Come si può vedere, la query per recuperare la media oraria impiega fino a 35 secondi per le centraline più attive, un tempo molto elevato.

Per affrontare questo tipo di problema e rendere più veloce l'aggregazione dei dati, Timescale mette a disposizione una funzionalità chiamata **continuous aggregates** (*aggregati continui*).

2.2.2 Continuous aggregates

I continuous aggregates sono una funzionalità integrata in Timescale che consente di aggregare i dati in tempo reale, senza la necessità di eseguire query aggiuntive [35]. Il funzionamento è simile alle **viste materializzate** di Postgres, con la differenza che i continuous aggregates non necessitano di essere aggiornati manualmente ogni volta che un nuovo dato viene inserito nella tabella. Infatti, la riaggredazione viene eseguita automaticamente in background tramite una **refresh policy** definita dall'utente in fase di creazione della vista.

L'utilizzo dei countinuous aggregates offre diversi vantaggi:

- Miglioramento delle **performance**, infatti non è più necessario scansioneare tutte le volte la tabella con i dati raw ma è sufficiente leggere questi risultati precalcolati;
- Funzionalità avanzate, come la possibilità di salvare i dati raw solo per un periodo di tempo limitato, continuando però a mantenere i dati aggregati. Così facendo si hanno dei dati riassuntivi per eventi che sono molto indietro nel tempo, mentre per gli eventi più recenti si continua ad avere tutti i dati raccolti, portando and un grosso **risparmio di spazio occupato**. [36]

Di contro, lo svantaggio di questa soluzione è che i dati non sono aggiornati ad ogni INSERT effettuata, ma solo ad un intervallo di tempo specificato dall'utente. Questo significa che in fase di lettura i dati più recenti non sono

presenti nelle viste materializzate. Per limitare questo problema si potrebbe pensare di diminuire l'intervallo di tempo necessario per il refresh dei dati, ma questo porterebbe ad un aumento del carico computazionale. C'è quindi un compromesso tra la velocità di lettura e l'aggiornamento dei dati. [36]

Timescale inoltre introduce il concetto di *hypertable*, normali tabelle SQL ma con partizionamento automatico su base temporale.

2.2.3 Risultati ottenuti

Di seguito sono riportati i passi eseguiti sul database di produzione AirQino per l'attivazione del meccanismo di continuous aggregates sulla tabella con i dati dei sensori (*station_data*):

1. È stata creata la *hypertable* a partire dalla tabella originale, avendo cura di migrare tutti i dati (questo è necessario perché i continuous aggregates si applicano solo alle *hypertable*):

```
1  SELECT create_hypertable('station_data', 'data_acquired', migrate_data =>
    true);
```

Dove *data_acquired* è la colonna temporale della tabella, in cui vengono salvati data e ora della misurazione. Questa operazione potrebbe impiegare molto tempo in base alla quantità di dati nella tabella; sul database di produzione AirQino ha richiesto circa 36 minuti;

2. È stata creata la vista materializzata per il calcolo delle medie orarie con continuous aggregates:

```
1  CREATE MATERIALIZED VIEW station_data_hourly_avg
2  WITH (timescaledb.continuous) AS
3  SELECT time_bucket('1_hour', sd.data_acquired) AS bucket,
```

```

4   station_id,
5   sensor_id,
6   avg(sd.float_value)
7 FROM station_data sd
8 GROUP by bucket, station_id, sensor_id;

```

Dove il nome dell'aggregato è *station_data_hourly_avg*, mentre la funzionalità di aggregazione continua viene specificata nella clausola *WITH*. Nella *SELECT* invece sono stati specificati la tabella di partenza (*station_data*), l'intervallo di aggregazione (*1 hour*) e l'operazione da eseguire per aggregare i dati (*avg()*, ovvero la media). Questa operazione sul database di produzione AirQino ha richiesto circa 5 minuti;

3. Infine, è stata creata una *refresh policy* adeguata con il comando:

```

1 SELECT add_continuous_aggregate_policy('station_data_hourly_avg',
2   start_offset => INTERVAL '7_days',
3   end_offset => INTERVAL '1_hour',
4   schedule_interval => INTERVAL '1_hour');

```

Dove *start_offset* indica quanto indietro nel tempo si va a calcolare l'aggregato, *end_offset* indica fino a quando fermarsi e *schedule_interval* indica l'intervallo di aggiornamento della lista. In questo caso è stato scelto di aggiornare, ogni ora, i dati della settimana passata, fino a un'ora prima del tempo di esecuzione. Per ragioni di performance conviene sempre escludere l'ultimo *bucket* (in questo caso l'ultima ora di dati arrivati). [37]

Una volta creata la vista, è possibile riscrivere la *SELECT* originale per la media degli ultimi 7 giorni, ma stavolta prendendo i dati da questa nuova

tabella con aggregati continui (*station_data_hourly_avg*):

```

1 SELECT sd.avg
2 FROM station_data_hourly_avg sd
3 WHERE bucket > NOW() - INTERVAL '7 days'
4 AND sd.sensor_id = 29510691 /* id centralina */
5 ORDER BY bucket DESC;

```

Questa query è risultata molto più performante rispetto alla precedente, essendo le medie orarie di fatto già precalcolate e sempre aggiornate in background. I nuovi risultati sulle centraline AirQino sono riportati di seguito in figura 2.5.

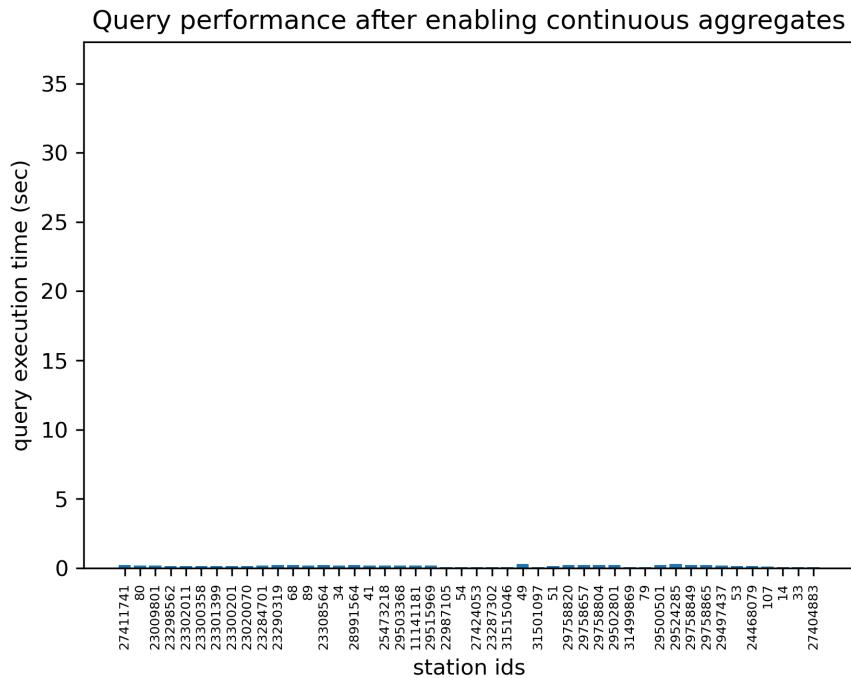


Figura 2.5: Tempi di risposta della query per estrarre la media oraria di NO₂ dell'ultima settimana per tutte le centraline AirQino, dopo l'ottimizzazione (media su 10 iterazioni)

Da cui si può notare che con la nuova query, ottimizzata con continuous aggregates, i tempi di risposta risultano costanti per tutte le centraline, indipendentemente dalla quantità di dati. Questo miglioramento ha portato ad una riduzione significativa dei tempi di risposta (anche 175 volte meno per alcune centraline), garantendo maggiore **scalabilità** ed **efficienza** al sistema.

Capitolo 3

Calibrazione

Questo capitolo riguarda la parte di tesi incentrata sulla calibrazione delle centraline AirQino (1.3). Nella sezione 3.1 viene introdotto l'attuale processo di calibrazione e le motivazioni; in 3.2 vengono presentati i dataset a disposizione, la loro struttura e il lavoro di preprocessamento fatto.

La sezione 3.3 racchiude una breve panoramica teorica sui concetti di regressione, correlazione, metriche (coefficiente di correlazione, coefficiente di determinazione, scarto quadratico medio), analisi dei residui e modelli di regressione (sia lineare che non lineare).

La sezione 3.4 elenca gli esperimenti svolti per ciascun inquinante (NO_2 , $\text{PM}_{2.5}$ e PM_{10}) e presenta i risultati ottenuti (con analisi dei residui e confrontando i vari modelli, sia su tutto l'anno che con cadenza mensile), mentre in 3.5 viene presentato il lavoro svolto per la validazione sul campo del modello selezionato.

Infine, la sezione 3.6 riassume tutto il lavoro svolto e fornisce un'analisi qualitativa dei risultati ottenuti.

3.1 Introduzione

Al fine di verificare il corretto funzionamento dei sensori della piattaforma è necessario implementare una procedura di **calibrazione**. Lo scopo di questa procedura è quello di catturare la risposta del sensore sotto differenti concentrazioni di gas nel *target range* di funzionamento del sensore. Nel caso di AirQino, questa procedura di calibrazione è composta da due fasi:

- **Test indoor:** nel laboratorio di Elettronica di IBE CNR viene allestito un banco di calibrazione. Per testare il funzionamento dei sensori di NO₂, CO, CO₂ e CH₄ vengono utilizzati dei campioni di gas a concentrazione nota, mentre per il sensore di O₃ viene utilizzato un generatore di ozono. Per studiare il comportamento del sensore di polveri (PM) è invece necessario fare il confronto con le misure di un *Particle Counter*, in particolare viene usato il **DustTrak DRX modello 8533** (TSI Inc., Shoreview, MN, USA) [38]. Il protocollo impiegato per la calibrazione prevede l'apporto di un flusso costante di gas. La scheda viene inserita in una cameretta dotata di due prese di aria, una collegata alla bombola di calibrazione in cui è presente un flussimetro e una per l'uscita. La risposta del sensore viene convertita da analogico a digitale con un convertitore (con uscita a 10 bit, ovvero un range di 0-1024 *counts*) presente nella scheda di acquisizione. Se la variazione del segnale del sensore supera i limiti del convertitore (1024 counts), viene effettuata una modifica del valore delle resistenze sul segnale di uscita, in modo tale da ottenere una risposta simile a quella del convertitore. Tenendo anche conto dei tempi di risposta dichiarati dalle specifiche dei sensori, il tempo massimo di ogni test è di circa 30 minuti, mentre il tempo minimo per completare una misura è di circa 15 minuti; [6]

- **Test outdoor:** la seconda fase della calibrazione riguarda lo studio della risposta del sensore sotto differenti condizioni ambientali. Si rende quindi necessario verificare i valori osservati per le concentrazioni di gas e polveri tramite la comparazione verso sensori fissi di riferimento. Per fare questo vengono individuate delle stazioni ARPAT di monitoraggio fisso sul territorio, in prossimità delle quali vengono posizionate le centraline; [6]
- **Test indoor vs reference:** come per il test outdoor, le centraline vengono messe in calibrazione assieme a degli strumenti di reference (**Horiba Ambient Air Pollution AP SERIES analyzers** [39]). Per un periodo di tempo di circa una settimana sia gli strumenti di reference che le centraline collezionano misure di concentrazione della stessa aria campionata. Dai dati raccolti in questo periodo vengono quindi valutate le performance dei sensori low cost, ed effettuate delle operazioni di ricalibrazione ulteriore. [6]

3.2 Dati a disposizione

I dataset a disposizione sono due:

- Dataset delle misurazioni di concentrazione di NO₂ nell'aria relative alla centralina SMART16 AirQino (43°50'23.4"N 10°34'22.2"E);
- Dataset delle misurazioni di concentrazione di PM_{2.5} e PM₁₀ nell'aria relative alla centralina SMART16 AirQino (43°50'23.4"N 10°34'22.2"E).

I dati sono stati collezionati nel periodo dal 01/01/2020 al 20/11/2021 di co-locazione di SMART16 con la stazione ARPAT di Capannori (Lucca). Il comune rurale di Capannori è stato scelto in quanto si trova all'interno di

un'area interessata sia da una varietà di fonti di emissione che da condizioni meteorologiche tali da evitare la dispersione degli inquinanti.



Figura 3.1: Una centralina AirQino

Fonte: <https://airqino.magenta-lab.it>



Figura 3.2: Centralina ARPAT in sede Capannori (provincia di Lucca)

Fonte: <http://arpat.toscana.it>

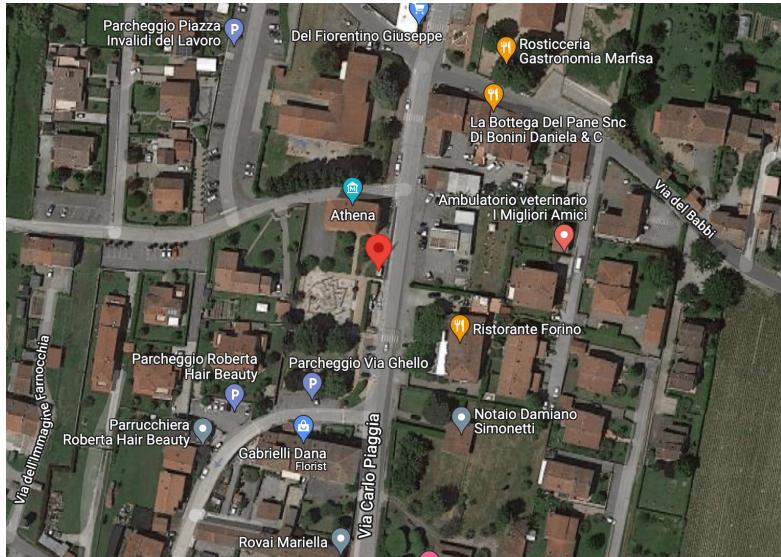


Figura 3.3: Posizione della centralina SMART16 (AirQino) e ARPAT (Capannori) a Lucca

3.2.1 Dataset NO₂

Il dataset di misurazioni NO₂ di AirQino è stato ottenuto con delle query specifiche sul database della piattaforma. I dati ARPAT (Capannori) per il confronto invece vengono messi a disposizione sul sito ufficiale [5] e sono accessibili tramite API, a seguito di un periodo di validazione che può durare anche diversi mesi. Ci sono delle differenze sostanziali tra i due set di dati:

	Periodo	Unità	Frequenza dati
SMART16	dal 01/01/2020 al 31/12/2020	counts	circa due minuti
ARPAT	dal 01/01/2020 al 31/12/2020	µg/m ³	medie giornaliere

Tabella 3.1: Differenze tra i dati di SMART16 e ARPAT (per NO₂)

Le centraline AirQino misurano la concentrazione di NO₂ con il sensore **MiCS-2714** (come già accennato in 1.3.2) che fornisce output in *counts*

(unità di misura del segnale che viene passato attraverso un convertitore analogico digitale, con uscita a 10 bit). Questo significa che sarà compito della fase di calibrazione convertire l'output direttamente in unità ingegneristica (in questo caso $\mu\text{g}/\text{m}^3$).

Nella figura 3.4 viene riportato l'andamento della concentrazione di NO_2 nell'ambiente (in *counts*) misurata dalla centralina SMART16 nel periodo di interesse (dal 01/01/2020 al 31/12/2020).

La figura 3.5 invece riporta l'andamento della concentrazione di NO_2 nell'aria (in $\mu\text{g}/\text{m}^3$) misurato dalla stazione ARPAT di Capannori nello stesso periodo.

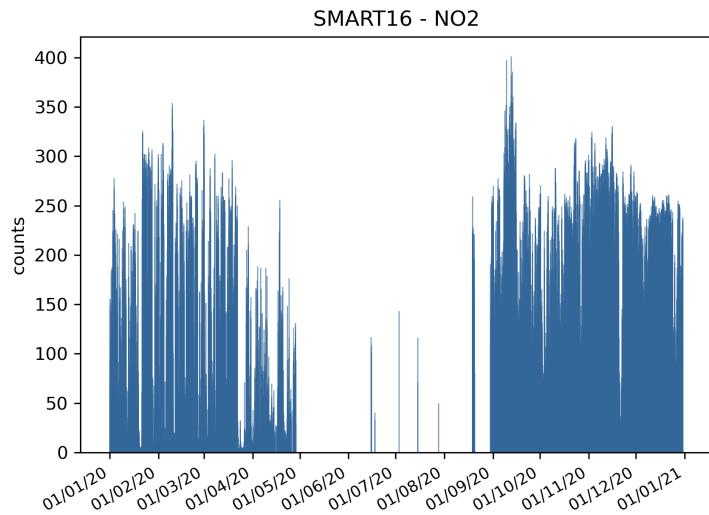


Figura 3.4: Andamento (in *counts*) di NO_2 (SMART16)
nel periodo dal 01/01/2020 al 31/12/2020

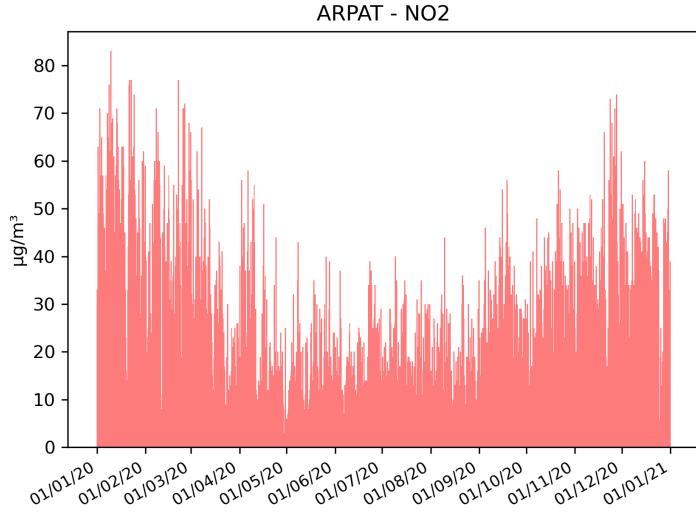


Figura 3.5: Andamento NO_2 (in $\mu\text{g}/\text{m}^3$) misurato dalla stazione ARPAT di Capannori nel periodo dal 01/01/2020 al 31/12/2020

3.2.2 Dataset $\text{PM}_{2.5}$ e PM_{10}

Come per NO_2 , i dati ARPAT (Capannori) invece sono stati collezionati tramite API da sito ufficiale, e poi confrontati con i dati di AirQino. Anche in questo caso ci sono delle differenze tra i due set di dati:

	Periodo	Unità	Frequenza dati
SMART16	dal 01/09/2020 al 31/08/2021	$\mu\text{g}/\text{m}^3$	circa due minuti
ARPAT	dal 01/09/2020 al 31/08/2021	$\mu\text{g}/\text{m}^3$	medie ogni otto ore

Tabella 3.2: Differenze tra i dati di SMART16 e ARPAT (per PM)

Il sensore utilizzato dalle centraline SMART è il **SDS011** (vedi 1.3.2) che fornisce l'uscita direttamente in unità ingegneristica ($\mu\text{g}/\text{m}^3$, con una calibrazione di fabbrica indicata nel *datasheet*), quindi in questo caso non c'è bisogno di convertire l'unità di misura nella fase di calibrazione (a differenza

di quanto accade con NO_2). Nella figura 3.6 sono riportati gli andamenti della concentrazione di $\text{PM}_{2.5}$ e PM_{10} nell'aria (in $\mu\text{g}/\text{m}^3$) misurati dalla centralina SMART16 nel periodo di interesse (dal 01/09/2020 al 31/08/2021). La figura 3.7 invece riporta gli andamenti della concentrazione di $\text{PM}_{2.5}$ e PM_{10} nell'aria (in $\mu\text{g}/\text{m}^3$) misurato dalla stazione ARPAT di Capannori.

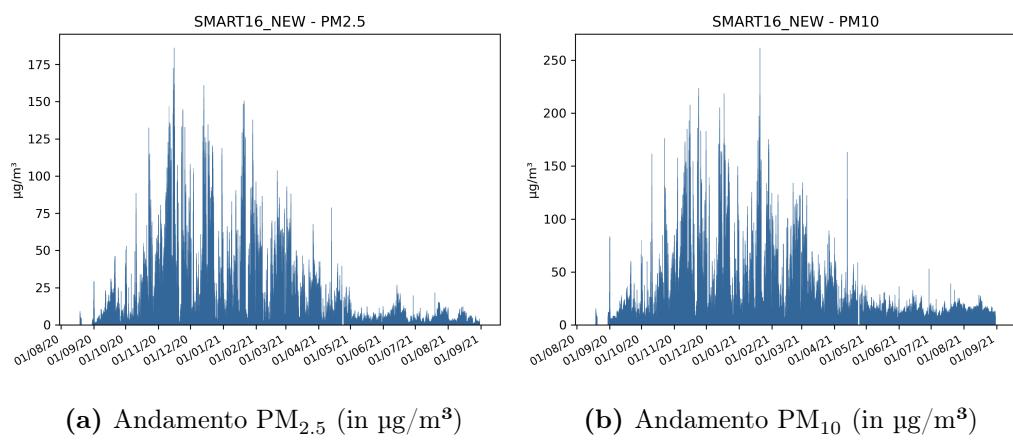


Figura 3.6: Andamento PM_{2.5} e PM₁₀ (SMART16)

nel periodo dal 01/09/2020 al 31/08/2021

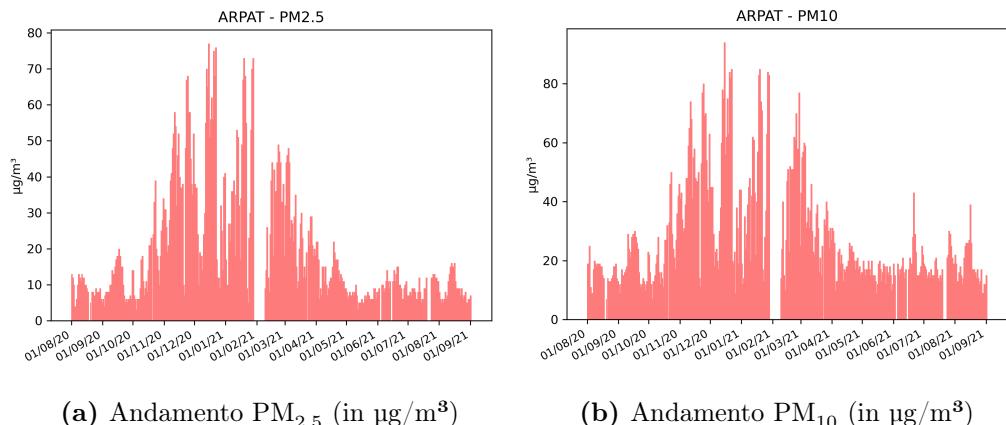


Figura 3.7: Andamento PM_{2.5} e PM₁₀ (ARPAT)

nel periodo dal 01/09/2020 al 31/08/2021

3.2.3 Preprocessamento

Per facilitare il caricamento e l'elaborazione, si è resa necessaria una fase iniziale di preprocessamento dei dati, descritta di seguito.

3.2.3.1 Dataset ARPAT NO₂

Il dataset originale NO₂, fornito da ARPAT, consiste in un file csv da 8785 misurazioni, con encoding ISO-8859-1, valori separati da punto e virgola (;), e strutturato come in figura 3.8. In particolare:

- La data è in formato AAAAMMGG (colonna "DATA");
- L'ora risulta in formato intero (colonna "ORA FINE MISURA", con valori da 1 a 24, dove 24 indica le 00:00);
- Data e ora sono da considerarsi con fuso orario locale;
- La colonna "VALIDITÀ" indica se la misurazione è da considerare valida per la calibrazione;
- La media oraria è riportata in $\mu\text{g}/\text{m}^3$ (come da normativa [40]).

LU-CAPANNORI_NO2_2020						
STAZIONE	PARAMETRO	DATA (formato AAAAMMGG)	ORA FINE MISURA (solare)	MEDIA ORARIA IN $\mu\text{g}/\text{m}^3$ A 20 °C	VALIDITÀ	
LU-CAPANNORI	NO2	20200101	1	33	1	
LU-CAPANNORI	NO2	20200101	2	NaN		0
LU-CAPANNORI	NO2	20200101	3	28	1	
LU-CAPANNORI	NO2	20200101	4	25	1	
LU-CAPANNORI	NO2	20200101	5	24	1	
LU-CAPANNORI	NO2	20200101	6	22	1	
LU-CAPANNORI	NO2	20200101	7	21	1	
LU-CAPANNORI	NO2	20200101	8	20	1	
LU-CAPANNORI	NO2	20200101	9	19	1	
LU-CAPANNORI	NO2	20200101	10	27	1	
LU-CAPANNORI	NO2	20200101	11	31	1	
LU-CAPANNORI	NO2	20200101	12	32	1	
LU-CAPANNORI	NO2	20200101	13	31	1	
LU-CAPANNORI	NO2	20200101	14	26	1	
LU-CAPANNORI	NO2	20200101	15	25	1	
LU-CAPANNORI	NO2	20200101	16	25	1	
LU-CAPANNORI	NO2	20200101	17	49	1	
LU-CAPANNORI	NO2	20200101	18	63	1	
LU-CAPANNORI	NO2	20200101	19	52	1	

Figura 3.8: Struttura del dataset originale NO₂ fornito da ARPAT

In questa fase sono state effettuate le seguenti modifiche:

- Data e ora sono state unite in una singola colonna;
- Data e ora sono state convertite in formato standard UTC¹ per conformarsi al dataset di AirQino;
- I dati non validi sono stati scartati;
- Le colonne sono state rinominate per semplicità ("data" per la data e "avg" per il valore di NO₂).

Il risultato è un file csv che si presenta come riportato in figura 3.9:

data	avg
2020-01-01 00:00:00+00:00	33.0
2020-01-01 02:00:00+00:00	28.0
2020-01-01 03:00:00+00:00	25.0
2020-01-01 04:00:00+00:00	24.0
2020-01-01 05:00:00+00:00	22.0
2020-01-01 06:00:00+00:00	21.0

Figura 3.9: Struttura del dataset ARPAT NO₂ processato

¹Il tempo coordinato universale o tempo civile, abbreviato con la sigla UTC, è il fuso orario scelto come riferimento globale, a partire dal quale sono calcolati tutti i fusi orari del mondo.

3.2.3.2 Dataset ARPAT PM_{2.5} e PM₁₀

Il dataset originale PM_{2.5} e PM₁₀, fornito da ARPAT, consiste in un singolo file csv da 33.674 misurazioni con encoding UTF-8, valori separati da virgola, e strutturato come in figura 3.10. In particolare:

- È stato considerato il periodo dal 01/09/2020 al 31/08/2021;
- La data è in formato gg/mm/aaaa (colonna "DATA");
- L'ora è in formato intero (colonna "ORA" con valori da 1 a 24, dove 24 indica le 00:00);
- Data e ora sono da considerarsi con fuso orario locale;
- I valori di PM_{2.5} e PM₁₀ sono riportati rispettivamente nelle colonne "PM2.5_LU-CAPANNORI" e "PM10_LU-CAPANNORI";
- I dati sono riportati come medie orarie, ma di fatto rappresentano medie ogni otto ore.

LU-CAPANNORI_PM_Dati_Orari			
DATA	ORA	PM10_LU-CAPANNORI	PM2.5_LU-CAPANNORI
18/1/2018	1	34	24
18/1/2018	2	34	24
18/1/2018	3	34	24
18/1/2018	4	34	24
18/1/2018	5	34	24
18/1/2018	6	34	24
18/1/2018	7	34	24
18/1/2018	8	34	24
18/1/2018	9	34	24
18/1/2018	10	34	24
	.	.	.

Figura 3.10: Struttura del dataset originale ARPAT PM_{2.5} e PM₁₀

Per questo dataset sono state effettuate le seguenti modifiche:

- Data e ora sono state unite in una singola colonna;
- Data e ora sono state convertite in formato standard UTC per conformati al dataset di AirQino;
- I dati non validi sono stati scartati;
- Le colonne sono state rinominate per semplicità ("data" per la data, "pm2.5" per i valori di PM_{2.5} e "pm10" per i valori di PM₁₀);
- I dati sono stati ricampionati e salvati come medie ogni otto ore.

Il risultato è un file csv di 4211 misurazioni che si presenta come riportato in figura 3.11:

data	pm10	pm2.5
2018-01-17 21:00:00+00:00	34.0	24.0
2018-01-18 05:00:00+00:00	34.0	24.0
2018-01-18 13:00:00+00:00	34.0	24.0
2018-01-18 21:00:00+00:00	35.25	26.5
2018-01-19 05:00:00+00:00	36.0	28.0
2018-01-19 13:00:00+00:00	36.0	28.0
2018-01-19 21:00:00+00:00	37.875	29.25

Figura 3.11: Struttura del dataset ARPAT PM_{2.5} e PM₁₀ processato con ricampionamento a otto ore

3.2.3.3 Dataset SMART16

Il dataset originale per la centralina SMART16 di AirQino consiste in due file csv (uno di 201.279 misurazioni per NO₂, e l'altro di 324.431 misurazioni per PM_{2.5} e PM₁₀), strutturati rispettivamente come in figura 3.12 e 3.13.

In particolare:

- Nel primo ci sono dati dal 01/01/2020 al 31/12/2020, nel secondo invece dal 18/08/2020 al 30/08/2021;
- Data e ora sono già in formato standard UTC;
- I valori di NO₂ sono riportati nella colonna "no2";
- I valori di PM_{2.5} e PM₁₀ sono riportati rispettivamente nelle colonne "pm2_5" e "pm10";
- In entrambi i file sono riportate anche le coordinate GPS inviate dalla centralina al momento della misurazione (colonne "long" e "lat");
- Le colonne "tair", "rad", "co2", "o3", "co", "voc", "ds18" si riferiscono rispettivamente a temperatura ambientale, umidità, CO₂, O₂, CO, VOC e temperatura interna;
- In entrambi i file i dati sono riportati con frequenza di circa due minuti.

SMART16														
	long	lat	data		tair	rad	co2	pm2_5	pm10	o3	no2	co	voc	ds18
0	10.577585	43.80190666666667	2020-01-01	00:00:02	2.8	97.6	458	92	71.0	352	212	164	444	15.3
1	10.577585	43.80190666666667	2020-01-01	00:01:36	2.8	97.9	458	101	78.0	354	212	169	449	15.29
2	10.577585	43.80190666666667	2020-01-01	00:03:10	2.9	98.1	457	111	82.0	356	208	165	443	15.28
3	10.577585	43.80190666666667	2020-01-01	00:04:44	2.9	98.2	456	111	81.0	352	199	163	438	15.25
4	10.577585	43.80190666666667	2020-01-01	00:06:18	3.0	98.2	456	102	80.0	349	191	162	436	15.26
5	10.577585	43.80190666666667	2020-01-01	00:07:52	3.0	98.0	456	113	83.0	349	195	164	438	15.25
6	10.577585	43.80190666666667	2020-01-01	00:09:26	3.0	97.6	456	105	79.0	349	199	164	439	15.26
7	10.577585	43.80190666666667	2020-01-01	00:11:00	3.0	97.3	456	96	74.0	346	191	162	434	15.27
8	10.577585	43.80190666666667	2020-01-01	00:12:34	3.0	97.0	452	89	66.0	340	174	161	430	15.26
9	10.577585	43.80190666666667	2020-01-01	00:14:08	3.0	96.7	452	95	69.0	337	166	160	428	15.25
10	10.577585	43.80190666666667	2020-01-01	00:15:42	3.0	96.4	452	93	69.0	336	170	160	430	15.25

Figura 3.12: Struttura del dataset originale SMART16 per NO₂

SMART16_new													
data	long	lat	tair	rad	co2	pm10	pm2_5	o3	no2	ds18			
2020-08-18 21:28:20	10.5728716666667	43.83988	21.2	98.4	501	17	8.0	352	310	27.67			
2020-08-18 21:29:54	10.5728716666667	43.83988	21.3	98.4	476	14	9.0	328	223	29.57			
2020-08-18 21:31:26	10.5728716666667	43.83988	21.3	98.4	471	14	9.0	315	208	30.52			
2020-08-18 21:33:00	10.5728716666667	43.83988	21.3	98.4	470	14	9.0	308	207	31.05			
2020-08-18 21:34:34	10.5728716666667	43.83988	21.3	98.4	468	15	9.0	302	205	31.4			
2020-08-18 21:36:08	10.5728716666667	43.83988	21.4	98.4	468	19	10.0	301	211	31.7			
2020-08-18 21:37:42	10.5728716666667	43.83988	21.4	98.4	468	17	10.0	300	213	31.74			
2020-08-18 21:39:16	10.5728716666667	43.83988	21.4	98.4	472	19	11.0	300	219	31.81			
2020-08-18 21:40:50	10.5728716666667	43.83988	21.5	98.4	472	18	10.0	299	217	31.91			
2020-08-18 21:42:24	10.5728716666667	43.83988	21.5	98.4	476	18	12.0	299	221	32.04			

Figura 3.13: Struttura del dataset originale SMART16 per PM_{2.5} e PM₁₀

Per questi due dataset sono state effettuate le seguenti modifiche:

- I dati del dataset NO₂ sono stati ricampionati a medie orarie;
- I dati del dataset PM_{2.5} e PM₁₀ sono stati ricampionati a otto ore per allinearsi ai dati ARPAT;
- I dati non validi sono stati scartati.

Di seguito sono riportati i risultati del preprocessamento dei due dataset:

data	no2	pm2_5	pm10
2020-01-01 00:00:00+00:00	155.556		
2020-01-01 01:00:00+00:00	92.967		
2020-01-01 02:00:00+00:00	77.057		
2020-01-01 03:00:00+00:00	52.618		
2020-01-01 04:00:00+00:00	68.706		
2020-01-01 05:00:00+00:00	68.576		
2020-01-01 06:00:00+00:00	90.818		

data	pm2_5	pm10
2020-08-30 05:00:00+00:00	1.54	7.92
2020-08-30 13:00:00+00:00	2.03	9.434
2020-08-30 21:00:00+00:00	3.269	11.192
2020-08-31 05:00:00+00:00	2.551	8.919
2020-08-31 13:00:00+00:00	2.607	6.227
2020-08-31 21:00:00+00:00	20.698	48.196
2020-09-01 05:00:00+00:00	9.533	17.854
2020-09-01 13:00:00+00:00	1.076	5.686
2020-09-01 21:00:00+00:00	905	4.102

(a) Dataset NO₂ processato

(b) Dataset PM processato

Figura 3.14: Struttura dei dataset SMART16 processati con ricampionamento a una e otto ore

3.2.3.4 Unione dei dataset

In seguito, per facilitare l'elaborazione dei dati e l'applicazione delle tecniche di regressione (3.3), i dataset SMART e ARPAT (sia NO₂ che PM_{2.5} e PM₁₀) sono stati uniti in un unico dataset in base all'indice in comune (colonna 'data'). I risultati di questa unione sono riportati di seguito (figure 3.15 e 3.16) e rappresentano i dataset finali utilizzati nella fase di sperimentazione (3.4), rispettivamente di 5500 e 1038 misurazioni totali.

data	airqino_no2	arpat_no2
2020-01-01 00:00:00+00:00	155.556	33.0
2020-01-01 02:00:00+00:00	77.057	28.0
2020-01-01 03:00:00+00:00	52.618	25.0
2020-01-01 04:00:00+00:00	68.706	24.0
2020-01-01 05:00:00+00:00	68.576	22.0
2020-01-01 06:00:00+00:00	90.818	21.0
2020-01-01 07:00:00+00:00	105.182	20.0
2020-01-01 08:00:00+00:00	148.182	19.0
2020-01-01 09:00:00+00:00	113.419	27.0

Figura 3.15: Struttura del dataset finale per NO₂ (SMART16 vs ARPAT)

data	airqino_pm2.5	airqino_pm10	arpat_pm2.5	arpat_pm10
2020-08-19 21:00:00+00:00	3.595	6.619	5.0	7.0
2020-08-30 05:00:00+00:00	1.54	7.92	6.0	14.0
2020-08-30 13:00:00+00:00	2.03	9.434	6.0	14.0
2020-08-30 21:00:00+00:00	3.269	11.192	5.25	13.25
2020-08-31 05:00:00+00:00	2.551	8.919	5.0	13.0
2020-08-31 13:00:00+00:00	2.607	6.227	5.0	13.0
2020-08-31 21:00:00+00:00	20.698	48.196	5.75	12.25
2020-09-01 05:00:00+00:00	9.533	17.854	6.0	12.0
2020-09-01 13:00:00+00:00	1.076	5.686	6.0	12.0
2020-09-01 21:00:00+00:00	905	4.102	4.5	9.75
2020-09-02 05:00:00+00:00	728	4.781	4.0	9.0
2020-09-02 13:00:00+00:00	713	4.976	4.0	9.0
2020-09-02 21:00:00+00:00	1.692	5.268	6.25	11.25
2020-09-03 05:00:00+00:00	1.051	5.068	7.0	12.0

Figura 3.16: Struttura del dataset finale per PM (SMART16 vs ARPAT)

3.3 Regressione

Nella statistica applicata si osserva l'esistenza di relazioni fra due o più grandezze. Sorge allora il problema di determinare una funzione che, in base ai dati ricavati mediante esperimenti o rilevazioni statistiche, rappresenti questi relazioni permettendo di analizzare meglio i fenomeni osservati. Con il termine *regressione* si intende una tecnica statistica che serve a stimare la relazione esistente tra due o più variabili.

3.3.1 Introduzione

Limitando lo studio a problemi che stabiliscono relazioni fra due sole variabili, si tratta, partendo dalle coppie (x_i, y_i) di dati rilevati, di determinare una funzione $y = f(x)$ che rappresenti la relazione.

Per fare questo si può procedere in due modi:

- determinare una funzione che assuma esattamente i valori (x_i, y_i) rilevati;
- determinare una funzione che si accosti il più possibile ai punti (x_i, y_i) .

Osservando la relazione tra le variabili si sceglie il tipo di funzione interpolante: lineare, quadratica, esponenziale, ecc. e quindi si procede alla determinazione dei parametri, ossia delle costanti che compaiono nella funzione scelta in modo che sia soddisfatta una condizione di accostamento prefissata.

Per conseguire questo scopo il metodo più utilizzato è il metodo dei **minimi quadrati**, che costituisce un'applicazione della ricerca del minimo di una funzione di più variabili mediante gli strumenti dell'analisi infinitesimale.

Considerate due variabili X e Y sulle quali vengono effettuate n rilevazioni:

$$(x_1, y_1), (x_2, y_2), \dots, (x_i, y_i), \dots, (x_n, y_n)$$

Sia $y = f(x; a, b, c, \dots, k)$ la funzione interpolante scelta. Siano inoltre \hat{y}_i valori predetti sulla curva corrispondenti ai valori x_i rilevati.

La condizione del metodo dei minimi quadrati è quella di determinare i valori dei parametri in modo che sia minima la somma dei quadrati delle differenze fra i valori osservati y_i e i valori predetti \hat{y}_i (figura 3.17), ovvero:

$$\varphi(a, b, c, \dots, k) = \sum_{i=1}^n [y_i - f(x_i; a, b, c, \dots, k)]^2$$

dove i valori x_i e y_i sono noti, mentre sono incogniti i parametri a, b, c, \dots, k della funzione. [41]

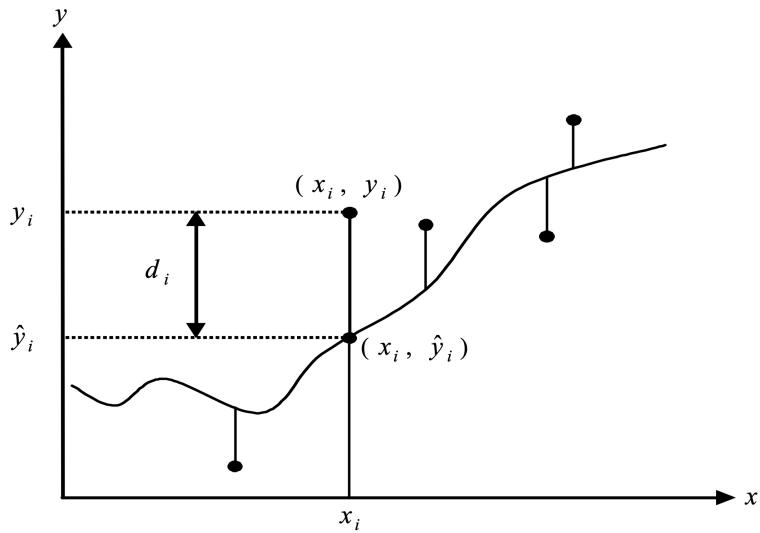


Figura 3.17: Condizione dei *minimi quadrati* [41]

3.3.2 Correlazione e coefficiente di determinazione

Quando la dipendenza tra le due variabili è lineare, si parla di correlazione lineare, e può essere valutata mediante il coefficiente di correlazione lineare (r):

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

dove il termine al numeratore rappresenta la *covarianza* di X e Y , cioè la variabilità congiunta delle coppie (x_i, y_i) di valori corrispondenti rispetto al proprio valor medio; il denominatore invece rappresenta il prodotto delle deviazioni standard di X ed Y .

Il coefficiente di correlazione lineare gode di importanti proprietà:

- $-1 \leq r \leq 1$;

- si ha $r = 1$ quando tutti i dati sono allineati lungo una retta crescente (figura 3.18);

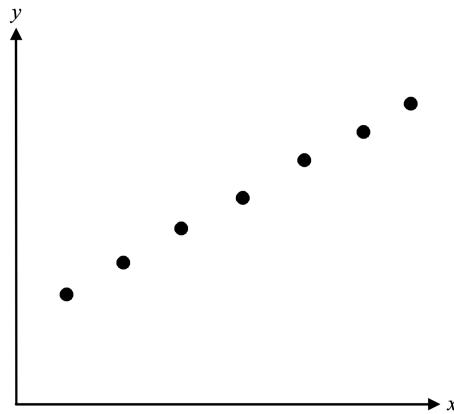


Figura 3.18: Correlazione lineare positiva

- si ha $r = -1$ quando tutti i dati sono allineati lungo una retta decrescente (figura 3.19);

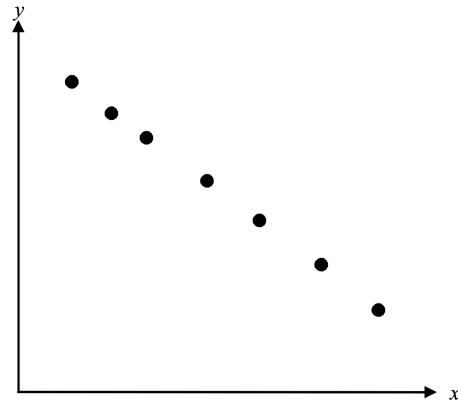


Figura 3.19: Correlazione lineare negativa

- si ha $r = 0$ quando non esiste una relazione lineare tra i dati (figura 3.20).

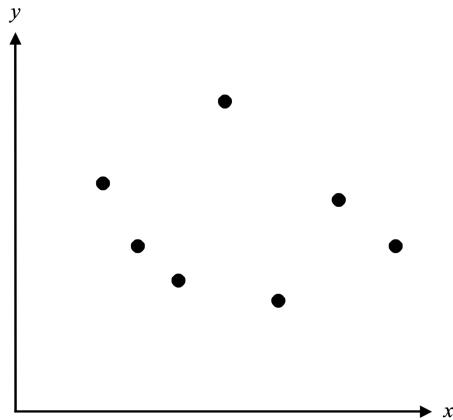


Figura 3.20: Nessuna correlazione

La varianza della variabile Y (σ_y^2) si può scomporre in una parte ($\sigma_{\hat{y}}^2$), detta *varianza spiegata*, in cui la variabilità della Y è dovuta alla dipendenza di Y dalla variabile X , e in una parte (σ_e^2), detta *varianza non spiegata* in cui la variabilità della Y non dipende dalla variabile X , ma da altri fattori. Si può quindi introdurre un secondo indicatore, dato dal rapporto tra la varianza spiegata e la varianza totale, chiamato **coefficiente di determinazione**:

$$r^2 = \frac{\sigma_{\hat{y}}^2}{\sigma_y^2}$$

che indica quale frazione della variazione della variabile Y può essere ricondotta e spiegata dalle variazioni della variabile X .

Sapendo che:

$$\sigma_y^2 = \sigma_{\hat{y}}^2 + \sigma_e^2$$

allora:

$$r^2 = \frac{\sigma_y^2}{\sigma_{\hat{y}}^2 + \sigma_e^2}$$

è evidente, quindi, che se la variabilità non spiegata è trascurabile, σ_e^2 tende ad annullarsi ed r^2 avrà un valore prossimo ad 1, mentre diventerà via via minore di 1 al diminuire dell'accordo tra la funzione calcolata e le osservazioni sperimentali.

Minore è la somma residua rispetto alla somma totale dei quadrati, maggiore sarà il valore del coefficiente di determinazione, r^2 , il quale è un indicatore del livello di precisione con cui l'equazione ottenuta dall'analisi di regressione spiega la relazione tra le variabili. [42]

Un'altra metrica utile in ambito delle regressioni è l'**errore quadratico medio** (in inglese *Mean Squared Error*, MSE) che indica la discrepanza quadratica media fra i valori dei dati osservati ed i valori dei dati stimati:

$$MSE = \frac{\sum_{i=1}^n (x_i - \hat{x}_i)^2}{n}$$

La sua radice quadrata fornisce un ulteriore indice statistico, la cosiddetta radice dell'errore quadratico medio (in inglese *root-mean-square error*, RMSE). L'RMSE può essere anche calcolato come deviazione standard degli scarti. Da notare che l'MSE ed RMSE non sono quantità adimensionali, ma assumono l'unità di misura della grandezza considerata (RMSE) ed il suo quadrato (MSE).

3.3.3 Analisi dei residui

Esistono metodi utili per diagnosticare le violazioni delle ipotesi di regressione di base: questi si basano principalmente sullo studio dei residui del modello. Spesso infatti la retta di regressione è una semplificazione della realtà e non coglie tutta la variabilità presente in un insieme di dati. [43]

Si definiscono i residui come:

$$e_i = y_i - \hat{y}_i, \quad i = 1, 2, \dots, n$$

dove y_i è il valore osservato e \hat{y}_i è il valore predetto.

Poiché un residuo può essere visto come la deviazione tra i dati e l'adattamento, è anche una misura della variabilità nella variabile di risposta, non spiegata dal modello di regressione. [44]

Eventuali scostamenti dalle ipotesi sugli errori dovrebbero quindi manifestarsi nei residui. L'analisi grafica dei residui è una tecnica efficace per verificare la linearità della relazione tra le variabili e scoprire diversi tipi di inadeguatezze del modello, tra cui:

- se i residui hanno distribuzione normale (3.3.3.1);
- se gli errori non sono indipendenti rispetto ai valori di X (3.3.3.2);
- se la varianza dei residui è omogenea (3.3.3.3);
- se ci sono degli outliers che influenzano la pendenza della retta (3.3.3.4).

3.3.3.1 Distribuzione degli errori

La distribuzione normale degli errori può essere verificata attraverso un grafico dei quantili, detto anche **q-q plot**. In questa tipologia di grafico,

i quantili teorici di una distribuzione Normale sono riportati sull'asse orizzontale. I quantili dei residui standardizzati sono invece riportati sull'asse verticale. L'idea è che se i residui hanno una distribuzione normale, i loro quantili dovrebbero coincidere con quelli della distribuzione normale. A livello grafico, questo significa che i punti dovrebbero disporsi lungo la *bisettrice*, indicata dalla retta presente nel grafico (figura 3.21).

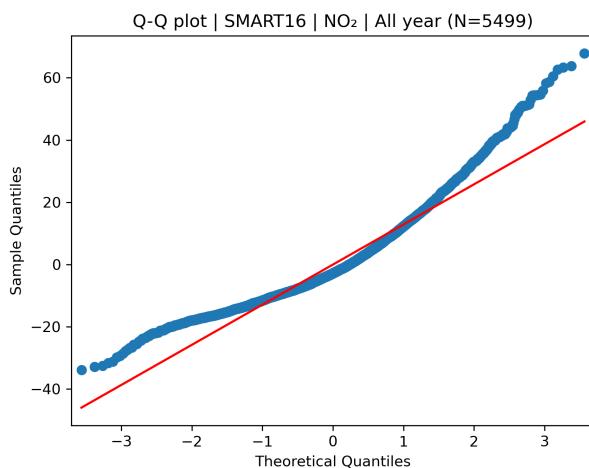


Figura 3.21: Esempio di grafico distribuzione degli errori (q-q plot)

3.3.3.2 Indipendenza degli errori

Se una variabile indipendente (X) risulta correlata con il termine d'errore, è possibile utilizzare questa variabile per predire quale sarà l'errore del modello di regressione. Questo in generale non è un buon segno, perché la componente di errore di un modello di previsione deve essere sempre imprevedibile in modo da evitare *bias* dovuto a misurazioni precedenti.

Per verificare l'indipendenza tra la variabile indipendente e i residui è utile osservare un grafico come quello riportato in figura 3.22, dove sull'asse

orizzontale si riportano i valori della x nell'ordine in cui sono stati raccolti, mentre sull'asse verticale i valori dei residui.

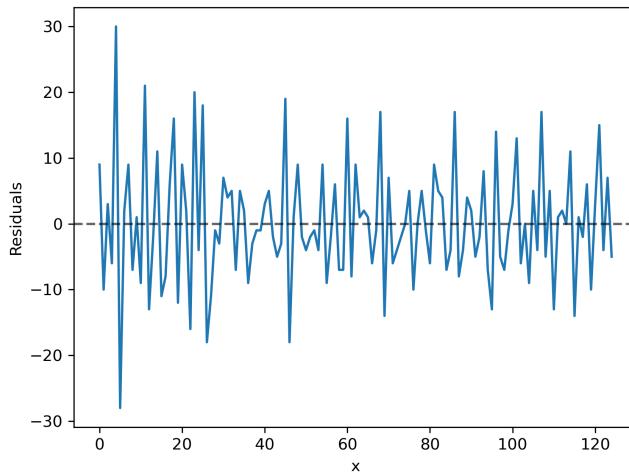


Figura 3.22: Esempio di plot dei residui

L'ipotesi è confermata se non è individuabile nessuna relazione tra le due variabili (ovvero se non compaiono trend o strutture cicliche nel tempo).

3.3.3.3 Omogeneità della varianza dei residui

Per verificare l'ipotesi di omogeneità delle varianze dei residui, in modo da valutare la bontà dell'adattamento, è necessario creare un grafico a dispersione (**scatterplot**). I valori predetti della y si riportano sull'asse orizzontale delle x . Sull'asse verticale delle y invece si indicano i valori dei residui (figura 3.23). [43]

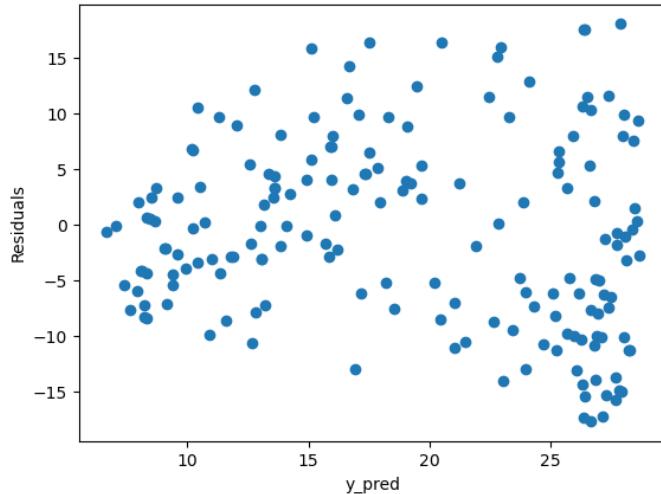


Figura 3.23: Esempio di grafico distribuzione dei residui

Se c’è omogeneità della varianza dei residui, i punti saranno dispersi in modo simile sia nella parte sinistra che in quella destra del grafico. Questa proprietà, se verificata, prende il nome di **omoschedasticità**. Al contrario, si parla di **eteroschedasticità** se i residui del modello non sono statisticamente indipendenti da tutte le variabili esplicative, e cioè quando la varianza degli errori non è la stessa in tutte le osservazioni.

3.3.3.4 Influenza degli outliers

Il grafico a dispersione tra valori predetti e residui permette di individuare anche i possibili **outliers**, ovvero i punti isolati nel grafico (quelli con residui maggiori). Tuttavia, per verificare se ci sono outliers in un modello di regressione, spesso si utilizzano altre tecniche (ad esempio eliminando i punti problematici tramite la distanza di Cook, descritta in 3.3.4.3, oppure applicando tecniche di regressione robusta meno sensibili alle le osservazioni problematiche, ad esempio con la funzione peso di Huber descritta in 3.3.4.2).

Nei modelli di regressione infatti anche un singolo outlier può influenzare in maniera sostanziale la capacità di adattamento del modello ai dati, soprattutto se il campione non risulta molto numeroso.

3.3.4 Modelli di regressione

3.3.4.1 Regressione lineare

Considerata una funzione lineare a due variabili x, y :

$$y = a + b * x$$

In questo caso si deve rendere minima la funzione:

$$\varphi(a, b) = \sum_{i=1}^n [y_i - (a + bx_i)]^2$$

Annullando le derivate parziali prime rispetto ad a e b si ha il sistema:

$$\begin{cases} \sum_{i=1}^n 2[y_i - (a + bx_i)](-1) = 0 \\ \sum_{i=1}^n 2[y_i - (a + bx_i)](-x_i) = 0 \end{cases}$$

che risolto, fornisce i valori dei parametri:

$$\begin{cases} \hat{a} = \bar{y} - b\bar{x} \\ \hat{b} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \end{cases}$$

dove \bar{x} e \bar{y} indicano le *medie aritmetiche*, rispettivamente di x_i e y_i .

La stima del parametro b , *coefficiente angolare* della funzione lineare, può essere rappresentato nella forma:

$$\hat{b} = \frac{\sum_{i=1}^n \frac{(x_i - \bar{x})(y_i - \bar{y})}{n}}{\sum_{i=1}^n \frac{(x_i - \bar{x})^2}{n}}$$

dove il denominatore è la *varianza* di X (σ_X^2), mentre il numeratore è detto *covarianza* di X e Y (σ_{XY}) e misura la variabilità congiunta delle coppie (x_i, y_i) di valori corrispondenti rispetto al proprio valor medio; quindi, il coefficiente b della retta interpolante esprime la variabilità congiunta di X e Y rapportata alla variabilità della sola X . [45]

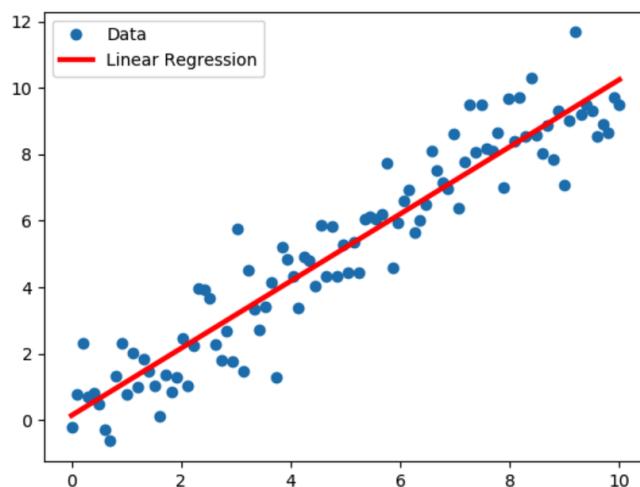


Figura 3.24: Esempio di retta di regressione lineare (in rosso)

I coefficienti della retta calcolata dalla regressione lineare dipendono dal grado di dispersione nei dati. Più l'andamento è lineare e più il modello risulterà accurato.

3.3.4.2 Regressione lineare robusta (Huber)

La regressione Huber (in inglese Huber regression) è una metodologia statistica per la stima dei parametri di un modello lineare in presenza di

outliers.

Ci sono situazioni in cui si verifica presenza di valori anomali che influiscono sul modello di regressione, nel senso che possono avere una forte influenza sul metodo dei minimi quadrati, di fatto influenzando negativamente l'accuratezza del modello di regressione. Il metodo dei minimi quadrati, infatti, in questi casi ha lo svantaggio di avere la tendenza ad essere dominato da questi valori — infatti sommando il quadrato dei residui ($\sum_{i=1}^n a_i^2$ dove a_i è il residuo i-esimo), la media risulta troppo influenzata da pochi valori a_i particolarmente grandi.

Ci sono due modi per affrontare questa situazione:

- Scartare le osservazioni *anomale* (vedi regressione lineare avanzata 3.3.4.3);
- Applicare procedure di stime robuste in modo che siano meno sensibili alle anomalie nelle osservazioni (figura 3.25).

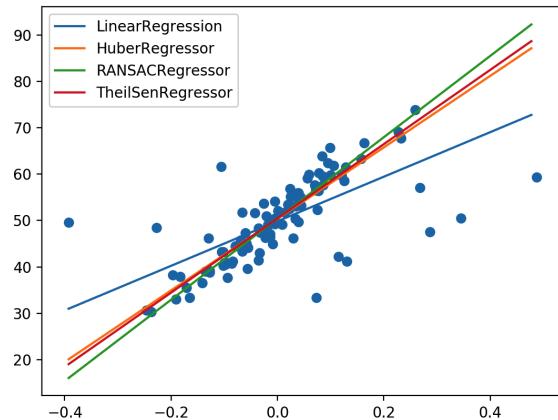


Figura 3.25: Comportamento di modelli di regressione robusta in presenza di outliers

Una delle funzioni di stima robusta, comunemente usata in diversi metodi di regressione per ridurre la sensibilità dei parametri alla presenza di

outliers, è la **funzione di Huber**, appartenente ad una classe di estimatori denominata **M-estimator**, e che risulta quadratica per piccoli valori di x , e lineare per valori più grandi. È definita come:

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{per } |a| \leq \delta \\ \delta(|a| - \frac{1}{2}\delta), & \text{altrimenti} \end{cases}$$

Dove la variabile a fa riferimento al residuo, cioè la differenza tra valore osservato e valore predetto ($a = y - f(x)$).

3.3.4.3 Regressione lineare avanzata

Come accennato in 3.3.4.2, un'altra tecnica per la gestione di outlier è quella di applicare il modello sul dataset dopo aver rimosso i valori anomali. Esistono molte metriche su cui basarsi per rimuovere gli outlier da un set di dati: un metodo che viene spesso utilizzato [46] nella regressione è la **distanza di Cook**.

La distanza di Cook è una stima dell'*influenza* di una osservazione in un dataset, in termini di residuo (outlier) o di elevato *leverage*: è una metrica utile per stimare di quanto cambierebbe un modello di regressione nel caso in cui venga rimossa l' i -esima osservazione.

In presenza di outliers la distanza di Cook aumenta, e quindi questi dati ad alta influenza hanno un maggiore impatto sulle stime dei parametri della regressione.

La distanza di Cook [47] dell'osservazione i ($\forall i = 1, \dots, n$) è definita come:

$$D_i = \frac{\sum_{j=1}^n (\hat{y}_j - \hat{y}_{j(i)})^2}{ps^2}$$

dove:

- n è il numero di osservazioni;
- \hat{y}_j è il valore predetto;
- $\hat{y}_{j(i)}$ è la risposta ottenuta escludendo l'i-esima osservazione.

Oppure, in modo equivalente:

$$D_i = \frac{e_i^2}{ps^2} \left[\frac{h_i}{(1 - h_i)^2} \right]$$

dove:

- $e_i = y_i - \hat{y}_i$ è l'i-esimo residuo;
- p è il numero di coefficienti della regressione;
- s^2 è l'errore quadratico medio (MSE);
- h_i è il peso che l'i-esimo osservazione ha sul valore della regressione (*leverage*).

Un esempio di rilevazione grafica di outlier tramite distanza di Cook è riportato in figura 3.26, ottenuta con l'ausilio della libreria Python `yellowbrick` [48].

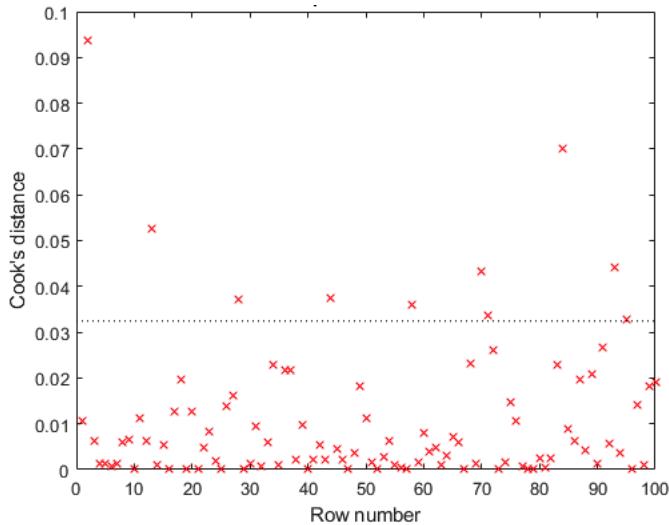


Figura 3.26: Riconoscimento di outlier tramite distanza di Cook

in letteratura vengono proposte alcune soglie di *cut-off*, oltre le quali un dato può essere considerato un outlier. In [49] viene proposta:

$$D_i > \frac{4}{n}$$

dove n è il numero di osservazioni.

3.3.4.4 Regressione Ridge

Nella statistica e nel Machine Learning, la regressione Ridge è un metodo di analisi di regressione che applica una fase di **regolarizzazione** al fine di migliorare l'accuratezza della previsione, prevenire l'*overfitting* e penalizzare la complessità del modello. In generale esistono due tipi di penalizzazione:

- **L1:** penalizza il valore assoluto dei coefficienti del modello (es. Lasso);
- **L2:** penalizza il quadrato del valore dei coefficienti del modello (es. Ridge).

Richiamando il metodo dei minimi quadrati (3.3.1) si deve minimizzare la somma dei quadrati dei residui (RSS):

$$\text{RSS} = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2$$

Nella regressione Ridge si aggiunge anche un termine di penalità, ottenendo quindi:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 = RSS + \lambda \sum_{j=1}^p \beta_j^2$$

Dove λ è un parametro di *tuning* che serve proprio a controllare l'effetto della penalità: un valore $\lambda = 0$ infatti non avrà effetto sul risultato finale (l'equazione viene ricondotta a quella dei minimi quadrati), al contrario per $\lambda \rightarrow \infty$ invece i coefficienti di regressione stimati tenderanno a zero poiché si darà molto peso alla penalità del modello. [50]

Il modello di regressione Ridge presenta dei vantaggi rispetto a quello dei minimi quadrati, soprattutto per quanto riguarda il *bias-variance trade-off*: quando λ aumenta la flessibilità del modello diminuisce, e questo comporta una riduzione della varianza ma un aumento del bias.

In generale, quando c'è una relazione lineare tra i predittori e la variabile risposta, il modello dei minimi quadrati comporta poco bias ma alta varianza. Questo si traduce nel fatto che una piccola variazione nell'osservazione può generare un cambiamento notevole nei coefficienti stimati. Al contrario, la regressione Ridge lavora bene nelle situazioni in cui il modello dei minimi quadrati genera ampia varianza nelle predizioni. [51]

3.3.4.5 Regressione Lasso

Lo svantaggio della regressione Ridge è il fatto di dover considerare tutte le variabili per la predizione nel modello finale. Il termine di regolarizzazione $\lambda \sum_{j=1}^p \beta_j^2$ tende ad assegnare ai coefficienti valori vicini allo zero, ma non perfettamente zero, a meno che $\lambda = 0$. Questo crea problemi non tanto per l'accuratezza della predizione, ma per l'interpretazione delle variabili, soprattutto quando il numero di queste diventa alto.

La regressione Lasso (acronimo di *least absolute shrinkage and selection operator*, ovvero operatore di restringimento e selezione minimo assoluto) è un'alternativa alla regressione Ridge utilizzata proprio per superare questo problema. L'unica differenza sta nel termine di regolarizzazione, ovvero:

$$\lambda \sum_{j=1}^p |\beta_j|$$

Per cui l'equazione del modello diventa:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| = RSS + \lambda \sum_{j=1}^p |\beta_j|$$

Anche nel caso di regressione Lasso il parametro di regolarizzazione tende a stimare i valori dei coefficienti verso lo zero ma, a differenza della regressione Ridge, la penalità $\lambda \sum_{j=1}^p |\beta_j|$ costringe uno o più coefficienti ad essere esattamente zero per certi valori di λ . [51]

3.3.4.6 Regressione polinomiale

La regressione polinomiale è una generalizzazione della regressione lineare, infatti utilizza lo stesso metodo matematico della variante lineare, ma assu-

me che la relazione di funzione che caratterizza i dati sia meglio descritta, anzichè da una retta, da un polinomio. In questo caso il metodo dei minimi quadrati può essere utilizzato anche per adattare una funzione polinomiale a un insieme di dati. Considerato un polinomio di grado k :

$$y = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_kx^k$$

In questo caso il sistema di equazioni da risolvere è:

$$\begin{cases} na_0 + a_1 \sum_{i=1}^n x_i + a_2 \sum_{i=1}^n x_i^2 + \dots + a_k \sum_{i=1}^n x_i^k = \sum_{i=1}^n y_i \\ a_1 \sum_{i=1}^n x_i + a_2 \sum_{i=1}^n x_i^2 + \dots + a_k \sum_{i=1}^n x_i^k = \sum_{i=1}^n x_i y_i \\ \dots \\ a_1 \sum_{i=1}^n x_i^k + a_2 \sum_{i=1}^n x_i^{k+1} + \dots + a_k \sum_{i=1}^n x_i^{2k} = \sum_{i=1}^n x_i^k y_i \end{cases}$$

che, risolto, permette di ricavare i parametri $a_0, a_1, a_2, \dots, a_k$.

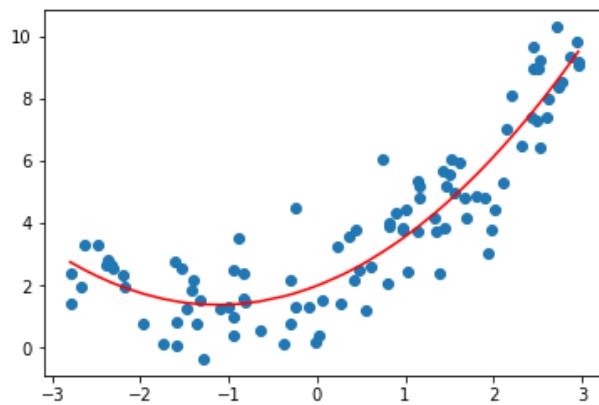


Figura 3.27: Esempio di regressione polinomiale di terzo grado

Ci sono diverse considerazioni importanti che emergono quando si adatta un polinomio in una variabile: una di queste riguarda la scelta dell'ordine del

modello. Come regola generale, l'uso di polinomi di ordine elevato ($n > 3$) dovrebbe essere evitato: un modello di ordine basso è quasi sempre preferibile a un modello di ordine elevato per ragioni di minore complessità, di coerenza con i dati e per evitare *overfitting*. [44]

Infatti se da una parte è sempre possibile trovare un polinomio di grado $n - 1$ ad n punti che risulti in un buon adattamento dei dati, non è detto che questo sia poi un buon predittore sui dati futuri (è il problema dell'*overfitting*).

3.3.4.7 Regressione con Random Forest

La regressione Random Forest è un algoritmo di apprendimento supervisionato che utilizza il metodo di apprendimento **ensemble** per la regressione tramite alberi di decisione. Il metodo di apprendimento ensemble è una tecnica che combina le previsioni di più algoritmi di apprendimento automatico per effettuare una previsione più accurata rispetto a un singolo modello. [52]

In particolare, Random Forest opera adattando una serie di alberi decisionali su vari sottocampioni del set di dati e utilizza la media dei risultati per migliorare l'accuratezza predittiva e controllare l'*overfitting*. In breve, l'algoritmo funziona eseguendo i seguenti passi:

1. Sceglie a caso k osservazioni dati dal *training set*;
2. Costruisce un albero decisionale associato a queste k osservazioni;
3. Sceglie il numero N di alberi da costruire e ripete i passaggi 1 e 2 per ciascuno;
4. Per una nuova osservazione, fa in modo che ciascuno degli N alberi preveda il valore di y , e assegna il nuovo punto alla media su tutti i valori y previsti.

Uno dei principali svantaggi degli alberi decisionali è che risultano molto inclini a provocare *overfitting*: funzionano bene sui dati di training, ma non sono così flessibili per fare previsioni su campioni non visti in precedenza. Sebbene esistano soluzioni alternative per migliorare questo aspetto, come ad esempio ridurre il numero di alberi, questo riduce il loro potere predittivo.

3.3.4.8 Regressione con Gradient Boosting

Il Gradient Boosting è una tecnica di machine learning che si applica sia a problemi di regressione che di classificazione statistica. Questa tecnica riprende il concetto matematico del **gradient descent**, per cui lo split scelto sarà quello che favorisce l'avvicinamento al punto di minimo della funzione obiettivo. Il *gradient descent* è un algoritmo di ottimizzazione che consente di individuare il valore minimo di una funzione di costo per sviluppare un modello con una previsione accurata.

L'algoritmo Gradient Boosting applicato a problemi di regressione può essere descritto nei seguenti passi:

1. Si inizializza il modello con un valore noto;
2. Si considera sul training set una *loss function*, ovvero una funzione differenziabile che esprima una valutazione della predizione (es. $\frac{1}{2}(y_i - \hat{y}_i)$ dove y_i è l'osservazione e \hat{y}_i è la predizione);
3. Scelto un numero massimo, si itera modellando un albero di regressione seguendo una procedura di discesa del gradiente, in modo da minimizzare la *loss function*. L'albero ottenuto viene aggiunto alla sequenza di alberi già esistente, nel tentativo di correggere o migliorare l'output finale del modello.

3.3.4.9 Regressione con SVR

Un altro modello per la regressione è SVR (*support-vector regression*), basato sui modelli SVM (*support-vector machines*) di apprendimento supervisionato, spesso impiegati nel problema della classificazione. [53]

Rispetto alla regressione lineare e al metodo dei minimi quadrati, SVR presenta più flessibilità perché consente di definire una soglia di accettazione dell'errore nel modello. Infatti l'algoritmo SVR si propone di minimizzare non l'errore quadratico, come nel metodo dei minimi quadrati, ma i coefficienti (nello specifico, la norma al quadrato del vettore dei coefficienti). La funzione obiettivo quindi diventa:

$$\min \frac{1}{2} \|\mathbf{w}\|^2$$

Il termine di errore invece è gestito nei vincoli, dove si imposta l'errore assoluto minore o uguale a un margine specificato, chiamato errore massimo (ε):

$$|y_i - w_i x_i| \leq \varepsilon$$

Il tuning del parametro ε consente di ottenere la precisione desiderata del modello di regressione. Solitamente alla funzione obiettivo si aggiunge anche delle variabili di *slack* (ξ_i), che indicano la deviazione dal margine di ciascun valore che supera la soglia ε (lo scopo è di minimizzarle il più possibile). La funzione obiettivo in questo caso diventa:

$$\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n |\xi_i|$$

dove C è un altro iperparametro regolabile: all'aumentare di C , aumenta anche la tolleranza per i punti al di fuori di ε . Quando invece C si avvicina a 0, la tolleranza si avvicina a 0 e l'equazione ricade nel caso semplificato.

Il tipo di kernel da usare nell'algoritmo è un altro parametro da definire nel modello: i kernel più comuni sono quello lineare, polinomiale (di grado n) o RBF (non lineare, basato su *funzione di base radiale*).

3.3.4.10 Regressione con KernelRidge

La regressione Kernel Ridge (o KRR, *Kernel Ridge Regression*) è un altro modello che combina la regressione Ridge (descritta in 3.3.4.4) con il *kernel trick*, imparando quindi una funzione lineare nello spazio indotto dal rispettivo kernel e dai dati. Per i kernel non lineari, questo corrisponde a una funzione non lineare nello spazio originale. [54]

La forma del modello di regressione KRR è identica a quello basato su SVR (descritto in 3.3.4.9), ma vengono utilizzate diverse funzioni di *loss*: KRR minimizza l'errore quadratico mentre SVR minimizza i coefficienti in base alla soglia ε . In generale il modello KRR risulta più veloce per dataset di medie dimensioni. [54]

3.4 Esperimenti e risultati ottenuti

Una volta collezionati i dati sia dalle centraline AirQino che ARPAT è stata avviata la fase di calibrazione, con le seguenti analisi per ciascun inquinante (NO_2 , $\text{PM}_{2.5}$ e PM_{10}):

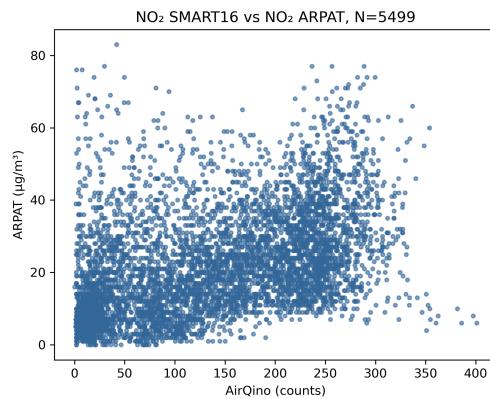
- **Scatterplot** preliminare delle misurazioni AirQino confrontate con le misurazioni di riferimento ARPAT, per individuare relazioni tra le variabili. Questo perché in generale
 - ci si aspetta che la risposta del sensore sia lineare all'aumentare delle concentrazioni e correlata al segnale di riferimento;
- **Analisi dei residui** della regressione lineare, per verificare le assunzioni del modello (vedi sezione 3.3.3);
- **Applicazione di modelli** di regressione lineare e non lineare su tutto il dataset:
 - Lineare semplice (3.3.4.1);
 - Lineare *robusto*, per ridurre l'influenza di outlier utilizzando la funzione peso di Huber (3.3.4.2);
 - Lineare *avanzato*, con rimozione di *outlier* tramite distanza di Cook basata sull'influenza della singola osservazione (3.3.4.3);
 - Ridge e Lasso (3.3.4.4 e 3.3.4.5);
 - Polinomiale (grado 2 e 3, 3.3.4.6);
 - SVR (con kernel lineare, polinomiale e RBF, come descritto in 3.3.4.9);
 - Random Forest, Gradient Boosting e KernelRidge (3.3.4.7, 3.3.4.8 e 3.3.4.10).
- Applicazione e valutazione degli stessi modelli con **cadenza mensile**, per validare la consistenza dei risultati;
- Infine, **valutazione della performance** e accuratezza dei modelli applicati.

Per tutti i risultati ottenuti e presentati nella sezione seguente sono state implementate le seguenti operazioni:

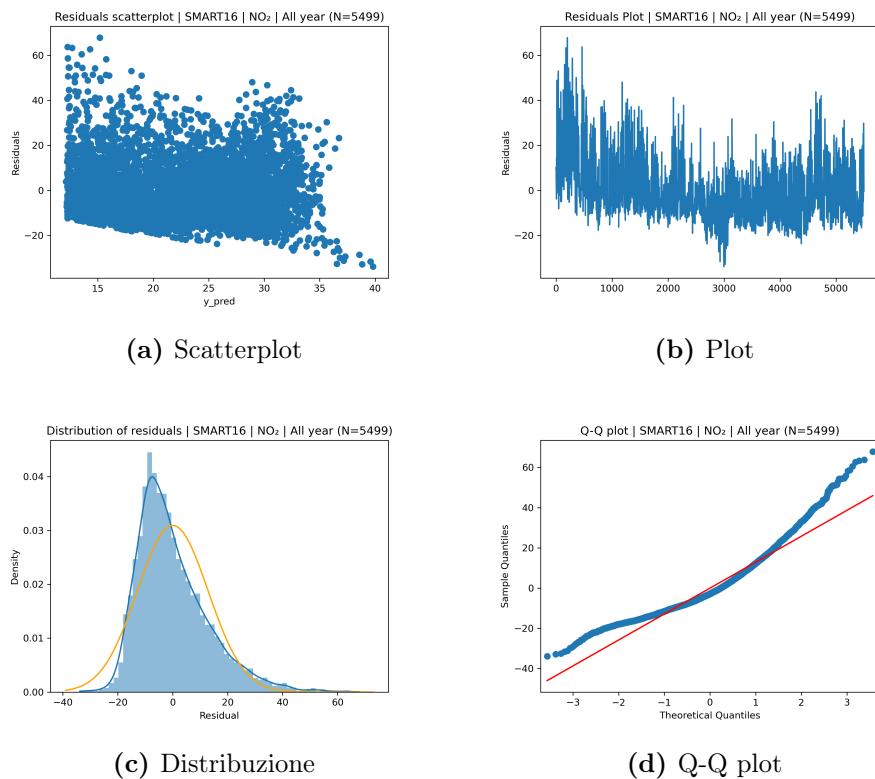
- Per misurare le performance dei vari modelli di regressione sono state utilizzate due metriche: il **coefficiente di determinazione** (R^2) e la **radice dell'errore quadratico medio** (RMSE), come visto in 3.3.2. Ciascun risultato rappresenta la media su 1000 iterazioni, con **shuffle** dei dati ad ogni iterazione;
- I modelli Gradient Boosting, Random Forest e SVR sono stati preparati tramite **tuning** di parametri ottimali con tecniche di *cross-validation*;
- Per l'addestramento il dataset è stato suddiviso in 70% train set e 30% test set, con l'ausilio della funzione **train_test_split** della libreria **scikit-learn** [55];
- Tutti gli esperimenti sono stati eseguiti con Python 3.9 e con l'aiuto di librerie aggiuntive quali **scikit-learn** [55], **pandas** [56], **matplotlib** [57], **numpy** [58] e **seaborn** [59].

3.4.1 Risultati NO₂

Il confronto tra le misurazioni AirQino e le misurazioni di riferimento ARPAT per NO₂ sono riportate nello *scatterplot* di figura 3.28.

**Figura 3.28:** Scatterplot dataset NO₂

I risultati dell'analisi grafica dei residui sono riportati in figura 3.29.

**Figura 3.29:** Analisi dei residui dataset NO₂

La tabella 3.3 mostra i risultati, in termini di R^2 e RMSE, della procedura di calibrazione applicata al sensore di NO_2 per ciascun modello di regressione, su tutto il dataset. In grassetto è stato evidenziato il modello migliore in termini di R^2 e RMSE. In figura 3.30 sono riportati gli stessi risultati in forma di istogramma, ordinati per performance.

Modello di regressione	R^2	RMSE ($\mu\text{g}/\text{m}^3$)
Lineare	0.304	11.393
Lineare robusto (Huber)	0.312	11.424
Lineare avanzato (Cook)	0.373	9.447
Ridge	0.305	11.375
Lasso	0.303	11.403
Polinomiale (grado 2)	0.305	11.382
Polinomiale (grado 3)	0.302	11.381
Random Forest	-0.034	13.839
Gradient Boosting	0.103	12.887
SVR (Kernel lineare)	0.287	11.528
SVR (Kernel polinomiale)	0.255	11.816
SVR (Kernel RBF)	0.294	11.445
KernelRidge	0.301	11.344

Tabella 3.3: Risultati della calibrazione NO_2

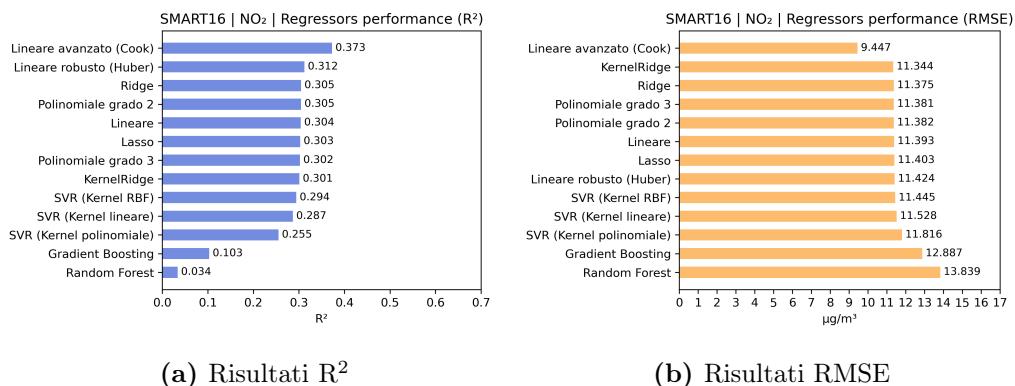


Figura 3.30: Istrogramma dei risultati della calibrazione NO_2

Per quanto riguarda l'applicazione del modello lineare *avanzato*, è stata riapplicata la regressione dopo aver eliminato tutti i punti con distanza di Cook (3.3.4.3) sopra una certa soglia ($\frac{4}{\text{numero di osservazioni}} = \frac{4}{5110} \approx 0,0007$). In particolare sono stati individuati e rimossi 282 outlier su 5110 osservazioni totali, circa il 5.52% (figura 3.31).

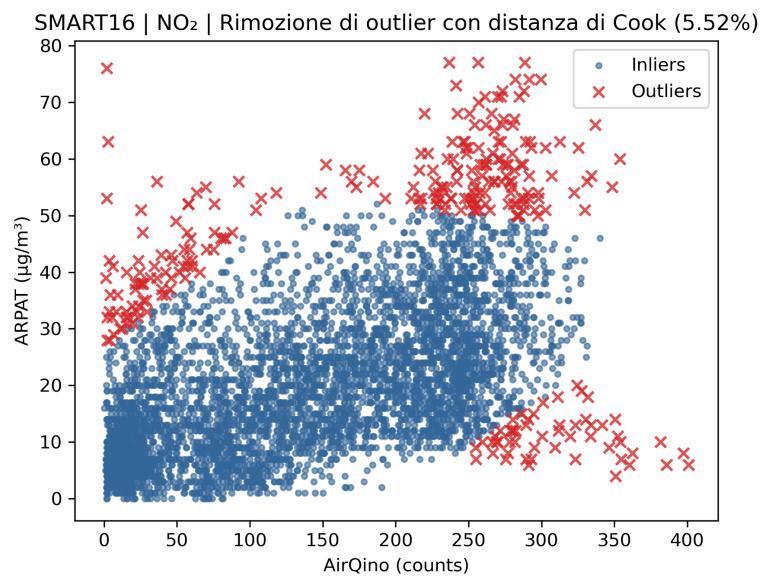


Figura 3.31: Rimozione di outlier con distanza di Cook su dataset NO₂

Nelle tabelle 3.4 e 3.5 sono invece riportati rispettivamente R^2 e RMSE della calibrazione su dataset NO₂ fatta con cadenza mensile.

Anche in questo caso il modello di regressione migliore è risultato in media quello *avanzato*, con riconoscimento e rimozione di outlier, anche se per alcuni mesi il modello polinomiale di terzo grado e il modello KernelRidge hanno ottenuto risultati migliori.

Modello	Gen	Feb	Mar	Apr	Set	Ott	Nov	Dic
Lineare	0.44	0.45	0.35	0.21	0.16	0.36	0.16	0.42
Lineare robusto (Huber)	0.44	0.45	0.35	0.21	0.16	0.36	0.16	0.42
Lineare avanzato (Cook)	0.53	0.48	0.38	0.27	0.29	0.39	0.18	0.45
Ridge	0.45	0.45	0.35	0.20	0.17	0.37	0.16	0.43
Lasso	0.43	0.45	0.34	0.21	0.16	0.37	0.17	0.42
Polinomiale (grado 2)	0.42	0.46	0.37	0.24	0.27	0.39	0.17	0.42
Polinomiale (grado 3)	0.42	0.51	0.48	0.23	0.27	0.41	0.17	0.42
Random Forest	0.15	0.33	0.21	-0.14	-0.07	0.12	-0.16	0.12
Gradient Boosting	0.13	0.17	0.16	0.08	0.09	0.13	0.06	0.14
SVR (lineare)	0.42	0.43	0.32	0.18	0.14	0.37	0.14	0.42
SVR (polinomiale)	0.37	0.37	0.22	0.05	0.07	0.29	0.13	0.39
SVR (RBF)	0.38	0.50	0.46	0.19	0.25	0.39	0.14	0.43
KernelRidge	0.44	0.45	0.37	0.24	0.26	0.39	0.18	0.41

Tabella 3.4: Risultati della calibrazione NO₂ con cadenza mensile (R^2)

Modello	Gen	Feb	Mar	Apr	Set	Ott	Nov	Dic
Lineare	13.31	11.68	9.90	9.30	8.73	8.72	12.30	9.86
Lineare robusto (Huber)	13.27	11.55	9.91	9.34	8.70	8.75	12.36	9.80
Lineare avanzato (Cook)	11.32	10.13	8.64	6.95	7.08	7.99	10.64	9.42
Ridge	13.23	11.52	9.91	9.18	8.73	8.67	12.33	9.72
Lasso	13.43	11.56	9.99	9.31	8.73	8.81	12.23	9.94
Polinomiale (grado 2)	13.54	11.46	9.66	9.11	8.19	8.57	12.36	9.72
Polinomiale (grado 3)	13.54	10.99	8.88	9.17	8.15	8.39	12.27	9.79
Random Forest	16.27	12.76	10.85	11.11	9.84	10.27	14.58	12.03
Gradient Boosting	16.64	14.30	11.30	9.90	9.09	10.34	13.04	11.98
SVR (lineare)	13.51	11.82	10.21	9.53	8.82	8.77	12.35	9.86
SVR (polinomiale)	14.10	12.43	10.74	10.07	9.29	9.23	12.51	10.05
SVR (RBF)	13.70	10.98	9.02	9.39	8.24	8.57	12.50	9.74
KernelRidge	13.35	11.51	9.80	9.17	8.21	8.62	12.06	9.78

Tabella 3.5: Risultati della calibrazione NO₂ con cadenza mensile (RMSE)

Infine, la tabella 3.6 riporta i coefficienti delle curve di regressione ottenuti applicando i modelli al dataset NO_2 . I coefficienti a, b, c, d, e si riferiscono ad una equazione polinomiale generica del tipo $y = a + bx + cx^2 + dx^3 + ex^4$, dove x rappresenta il dato grezzo e y il dato calibrato.

Modello di regressione	a	b	c	d	e
Lineare	6.52	12.25	/	/	/
Lineare robusto (Huber)	6.62	11.98	/	/	/
Lineare avanzato (Cook)	6.84	9.56	/	/	/
Lasso	6.52	12.26	/	/	/
Ridge	6.52	12.25	/	/	/
Polinomiale (grado 2)	12.11	6.83	-0.10	/	/
Polinomiale (grado 3)	12.71	4.23	1.91	-0.40	/
Polinomiale (grado 4)	9.97	22.86	-22.40	10.21	-1.47

Tabella 3.6: Coefficienti della calibrazione NO_2

3.4.2 Risultati $\text{PM}_{2.5}$

Il confronto tra le misurazioni AirQino e le misurazioni di riferimento ARPAT per $\text{PM}_{2.5}$ sono riportate nello *scatterplot* di figura 3.32.

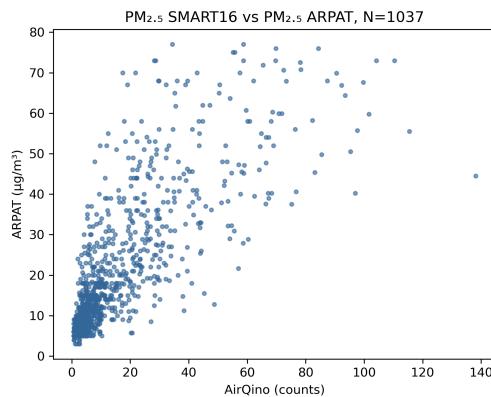


Figura 3.32: Scatterplot dataset $\text{PM}_{2.5}$

I risultati dell'analisi grafica dei residui sono riportati in figura 3.33.

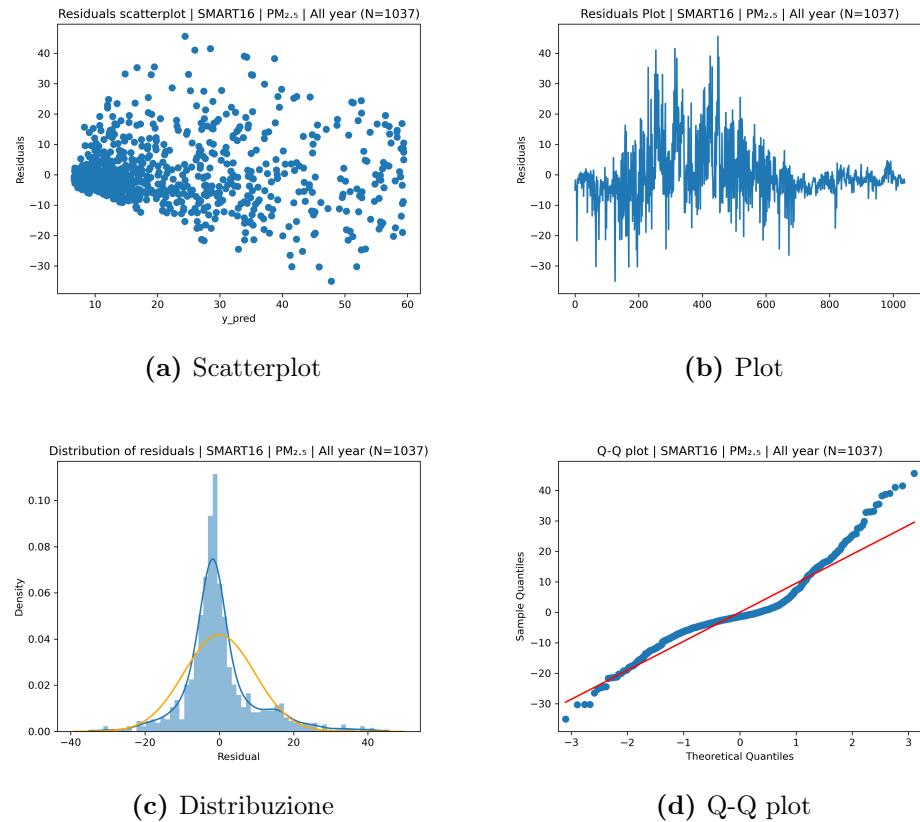


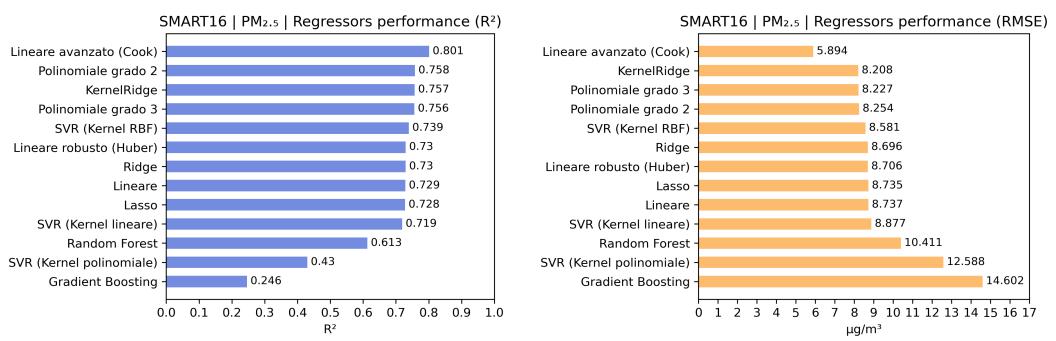
Figura 3.33: Analisi dei residui dataset PM_{2.5}

La tabella 3.7 mostra i risultati della procedura di calibrazione applicata al sensore di PM_{2.5} per ciascun modello di regressione, su tutto il dataset.

Modello di regressione	R^2	RMSE ($\mu\text{g}/\text{m}^3$)
Lineare	0.729	8.737
Lineare robusto (Huber)	0.730	8.706
Lineare avanzato (Cook)	0.801	5.894
Ridge	0.730	8.696
Lasso	0.728	8.735
Polinomiale (grado 2)	0.758	8.254
Polinomiale (grado 3)	0.756	8.227
Random Forest	0.613	10.411
Gradient Boosting	0.246	14.602
SVR (Kernel lineare)	0.719	8.877
SVR (Kernel polinomiale)	0.430	12.588
SVR (Kernel RBF)	0.739	8.581
KernelRidge	0.757	8.208

Tabella 3.7: Risultati della calibrazione PM_{2.5}

E in figura 3.34 gli stessi risultati in forma di istogramma, ordinati per performance:

(a) R²

(b) RMSE

Figura 3.34: Iстограмма dei risultati della calibrazione PM_{2.5}

Anche in questo caso nel modello lineare *avanzato*, è stata riapplicata la regressione dopo aver eliminato tutti i punti con distanza di Cook (3.3.4.3) sopra la soglia pari a $\frac{4}{\text{numero di osservazioni}} = \frac{4}{1037} \approx 0,0038$. In particolare sono stati individuati e rimossi 80 outlier su 1037 osservazioni totali, circa il 7.71% (figura 3.35).

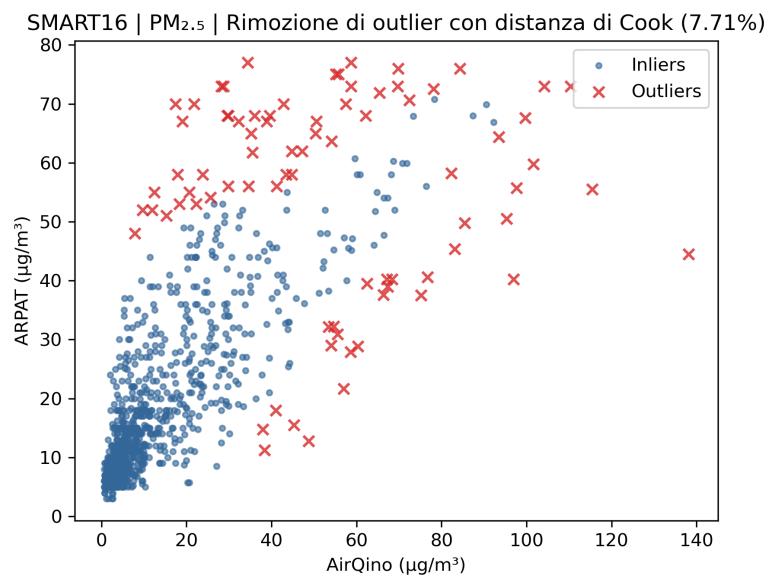


Figura 3.35: Rimozione di outlier con distanza di Cook su dataset PM_{2.5}

Nelle tabelle 3.8 e 3.9 sono invece riportati rispettivamente R^2 e RMSE della calibrazione su dataset PM_{2.5} fatta con cadenza mensile.

Modello	Set	Ott	Nov	Dic	Gen	Feb	Mar	Apr	Mag	Giu	Lug	Ago
Lineare	0.29	0.49	0.18	0.72	0.61	0.60	0.56	0.19	0.02	0.26	0.55	0.61
Lineare robusto (Huber)	0.28	0.48	0.19	0.72	0.62	0.58	0.58	0.18	-0.00	0.27	0.55	0.61
Lineare avanzato (Cook)	0.58	0.62	0.15	0.73	0.66	0.68	0.62	0.29	0.02	0.28	0.69	0.70
Ridge	0.26	0.48	0.18	0.71	0.61	0.59	0.57	0.18	0.01	0.26	0.52	0.60
Lasso	0.28	0.47	0.18	0.72	0.60	0.59	0.56	0.18	0.01	0.26	0.54	0.62
Polinomiale (grado 2)	0.32	0.46	0.22	0.73	0.68	0.55	0.54	0.09	-0.03	0.28	0.59	0.59
Polinomiale (grado 3)	-0.70	0.31	0.19	0.66	0.65	0.54	0.53	-0.24	-0.10	0.26	0.33	0.60
Random Forest	0.23	0.37	-0.03	0.65	0.50	0.29	0.28	-0.17	-0.68	-0.24	0.46	0.49
Gradient Boosting	0.09	0.10	0.03	0.20	0.17	0.08	0.12	-0.02	-0.08	0.02	0.14	0.15
SVR (lineare)	0.14	0.45	0.14	0.69	0.60	0.53	0.53	0.15	0.01	0.24	0.52	0.61
SVR (polinomiale)	-0.55	-0.15	0.08	0.50	0.26	0.40	0.51	0.12	-0.02	0.17	-0.09	0.57
SVR (RBF)	0.16	0.44	0.22	0.65	0.58	0.39	0.37	-0.06	-0.36	-0.07	0.62	0.50
KernelRidge	0.36	0.45	0.25	0.73	0.67	0.57	0.54	0.06	-0.06	0.29	0.62	0.60

Tabella 3.8: Risultati della calibrazione PM_{2.5} con cadenza mensile (R^2)

Modello	Set	Ott	Nov	Dic	Gen	Feb	Mar	Apr	Mag	Giu	Lug	Ago
Lineare	3.78	6.27	13.35	11.55	12.39	7.70	6.27	3.74	1.38	1.92	1.54	2.05
Lineare robusto (Huber)	3.77	6.25	13.19	11.58	12.17	7.87	6.27	3.77	1.38	1.93	1.53	2.04
Lineare avanzato (Cook)	2.92	4.99	12.78	10.62	11.55	7.08	5.53	3.38	1.25	1.80	1.26	1.72
Ridge	3.86	6.31	13.36	11.69	12.33	7.82	6.29	3.80	1.38	1.96	1.60	2.06
Lasso	3.82	6.32	13.32	11.56	12.48	7.85	6.29	3.77	1.36	1.95	1.57	2.04
Polinomiale (grado 2)	3.63	6.42	12.94	11.38	11.29	8.03	6.43	3.95	1.38	1.88	1.45	2.11
Polinomiale (grado 3)	5.10	6.89	13.13	12.57	11.46	8.13	6.56	4.43	1.42	1.94	1.73	2.03
Random Forest	3.85	6.86	14.86	12.62	13.54	10.14	7.93	4.44	1.76	2.48	1.68	2.27
Gradient Boosting	4.33	8.32	14.71	19.83	18.33	11.96	9.30	4.24	1.44	2.29	2.14	3.09
SVR (lineare)	3.98	6.46	13.55	12.10	12.46	8.30	6.50	3.79	1.39	1.98	1.57	2.05
SVR (polinomiale)	5.27	9.25	14.23	15.22	16.60	9.09	6.74	3.94	1.39	2.08	2.20	2.13
SVR (RBF)	3.99	6.51	12.98	12.88	12.60	9.41	7.54	4.25	1.57	2.33	1.41	2.23
KernelRidge	3.56	6.45	12.65	11.31	11.31	7.92	6.36	4.01	1.42	1.88	1.39	2.06

Tabella 3.9: Risultati della calibrazione PM_{2.5} con cadenza mensile (RMSE)

Infine, la tabella 3.10 riporta i coefficienti delle curve di regressione ottenuti applicando i modelli al dataset PM_{2.5}. Anche in questo caso i coefficienti a , b , c , d , e si riferiscono ad una equazione polinomiale generica del tipo $y = a + bx + cx^2 + dx^3 + ex^4$, dove x rappresenta il dato grezzo e y il dato calibrato.

Modello di regressione	a	b	c	d	e
Lineare	13.15	9.50	/	/	/
Lineare robusto (Huber)	13.33	9.23	/	/	/
Lineare avanzato (Cook)	10.31	7.67	/	/	/
Lasso	13.02	9.60	/	/	/
Ridge	13.01	9.61	/	/	/
Polinomiale (grado 2)	5.93	22.37	-2.34	/	/
Polinomiale (grado 3)	5.19	25.32	-4.07	0.23	/
Polinomiale (grado 4)	4.17	30.76	-9.29	1.75	-0.13

Tabella 3.10: Coefficienti della calibrazione PM_{2.5}

3.4.3 Risultati PM₁₀

Il confronto tra le misurazioni AirQino e le misurazioni di riferimento ARPAT per PM₁₀ sono riportate nello *scatterplot* di figura 3.36.

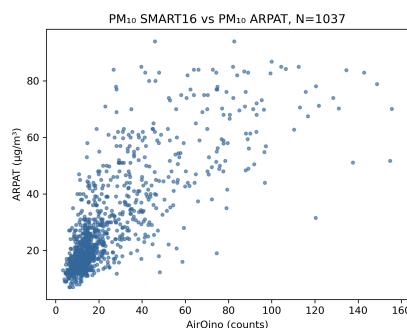


Figura 3.36: Scatterplot dataset PM₁₀

I risultati dell'analisi grafica dei residui sono riportati in figura 3.37.

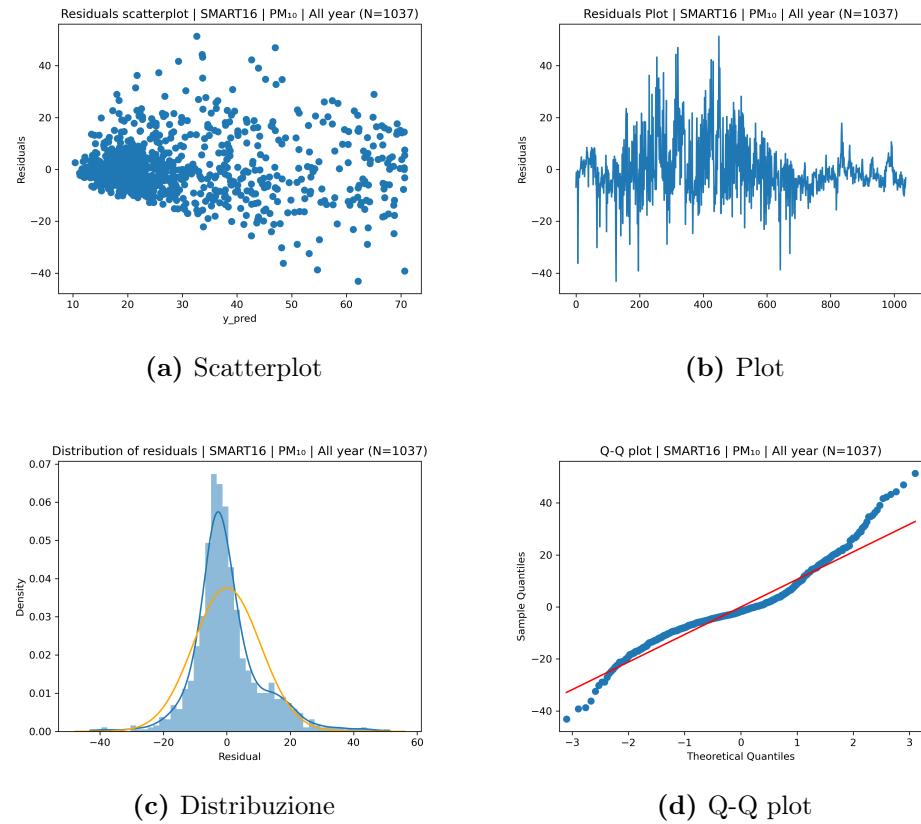


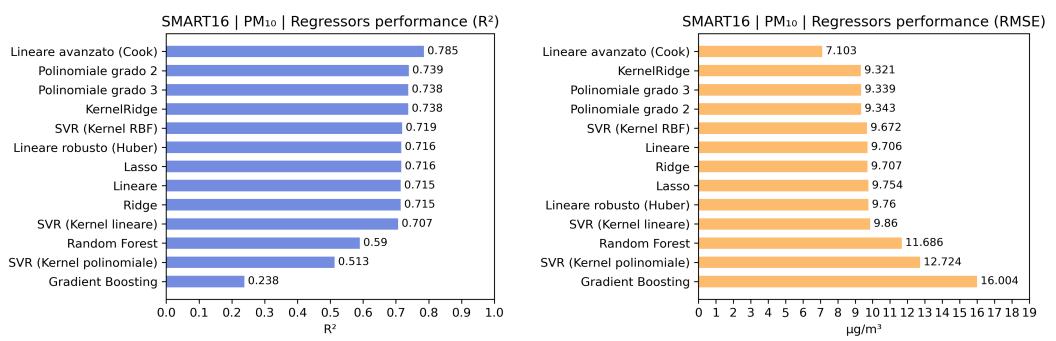
Figura 3.37: Analisi dei residui dataset PM₁₀

La tabella 3.11 mostra i risultati della procedura di calibrazione applicata al sensore di PM₁₀ per ciascun modello di regressione, su tutto il dataset.

Modello di regressione	R ²	RMSE ($\mu\text{g}/\text{m}^3$)
Lineare	0.715	9.706
Lineare robusto (Huber)	0.716	9.760
Lineare avanzato (Cook)	0.785	7.103
Ridge	0.715	9.707
Lasso	0.716	9.754
Polinomiale (grado 2)	0.739	9.343
Polinomiale (grado 3)	0.738	9.339
Random Forest	0.590	11.686
Gradient Boosting	0.238	16.004
SVR (Kernel lineare)	0.707	9.860
SVR (Kernel polinomiale)	0.513	12.724
SVR (Kernel RBF)	0.719	9.672
KernelRidge	0.738	9.321

Tabella 3.11: Risultati della calibrazione PM₁₀

E in figura 3.38 gli stessi risultati in forma di istogramma, ordinati per performance:

**Figura 3.38:** Istogramma dei risultati della calibrazione PM₁₀

Anche in questo caso nel modello lineare *avanzato*, è stata riapplicata la regressione dopo aver eliminato tutti i punti con distanza di Cook (3.3.4.3) sopra la soglia pari a $\frac{4}{\text{numero di osservazioni}} = \frac{4}{1037} \approx 0,0038$. In particolare sono stati individuati e rimossi 70 outlier su 1037 osservazioni totali, circa il 6.71% (figura 3.39).

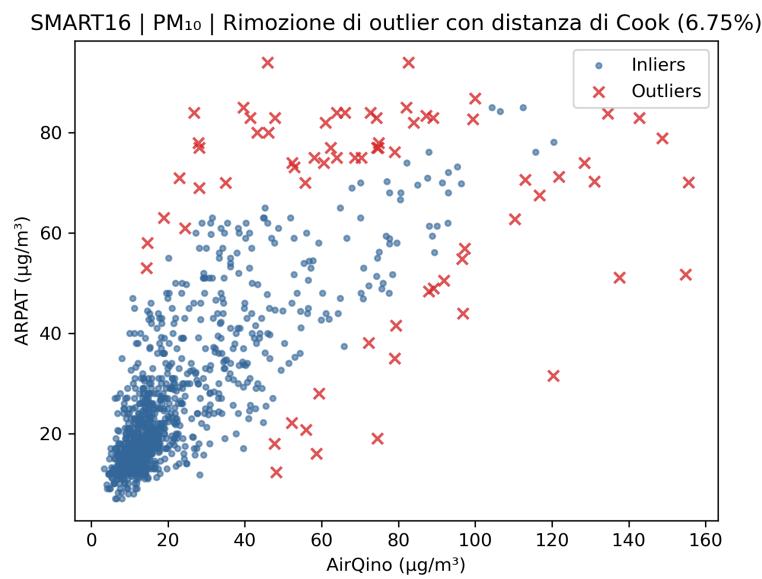


Figura 3.39: Rimozione di outlier con distanza di Cook su dataset PM₁₀

Nelle tabelle 3.12 e 3.13 sono invece riportati rispettivamente R^2 e RMSE della calibrazione su dataset PM₁₀ fatta con cadenza mensile.

Modello	Set	Ott	Nov	Dic	Gen	Feb	Mar	Apr	Mag	Giu	Lug	Ago
Lineare	-0.02	0.36	0.24	0.67	0.55	0.64	0.55	0.07	0.26	0.20	0.64	0.71
Lineare robusto (Huber)	-0.02	0.34	0.24	0.68	0.55	0.64	0.56	0.10	0.27	0.21	0.64	0.70
Lineare avanzato (Cook)	0.39	0.54	0.22	0.69	0.62	0.66	0.67	0.10	0.30	0.03	0.62	0.71
Ridge	-0.00	0.34	0.25	0.68	0.53	0.63	0.54	0.03	0.26	0.18	0.64	0.72
Lasso	0.01	0.35	0.23	0.69	0.53	0.64	0.53	0.05	0.26	0.20	0.62	0.71
Polinomiale (grado 2)	0.37	0.38	0.27	0.68	0.60	0.62	0.52	-0.03	0.25	0.11	0.65	0.67
Polinomiale (grado 3)	-1.00	0.26	0.23	0.68	0.58	0.58	0.49	-0.08	0.23	-1.17	0.59	0.60
Random Forest	0.16	0.03	0.11	0.55	0.41	0.46	0.45	-0.36	-0.10	-0.11	0.57	0.52
Gradient Boosting	0.09	0.07	0.07	0.18	0.13	0.12	0.15	-0.03	0.02	-0.03	0.15	0.14
SVR (lineare)	-0.36	0.33	0.19	0.66	0.54	0.59	0.49	0.09	0.25	0.27	0.62	0.71
SVR (polinomiale)	-0.98	-0.01	0.13	0.49	0.14	0.41	0.48	-0.00	0.20	0.17	0.65	0.68
SVR (RBF)	0.33	0.13	0.28	0.63	0.53	0.41	0.37	-0.21	0.07	-0.35	0.53	0.34
KernelRidge	0.38	0.40	0.30	0.68	0.61	0.65	0.52	-0.05	0.27	0.14	0.60	0.69

Tabella 3.12: Risultati della calibrazione PM₁₀ con cadenza mensile (R^2)

Modello	Set	Ott	Nov	Dic	Gen	Feb	Mar	Apr	Mag	Giu	Lug	Ago
Lineare	6.40	8.47	14.19	13.61	15.06	10.52	7.24	4.55	2.38	4.55	2.44	3.20
Lineare robusto (Huber)	6.35	8.54	14.19	13.53	15.21	10.54	7.26	4.46	2.41	4.52	2.48	3.21
Lineare avanzato (Cook)	5.02	6.60	13.37	12.06	13.86	10.26	6.09	4.22	2.24	3.46	1.94	2.88
Ridge	6.42	8.60	14.04	13.51	15.48	10.53	7.30	4.56	2.39	4.63	2.45	3.15
Lasso	6.43	8.52	14.22	13.51	15.40	10.64	7.32	4.62	2.41	4.59	2.47	3.23
Polinomiale (grado 2)	5.04	8.33	13.79	13.37	14.06	10.75	7.39	4.70	2.42	4.86	2.37	3.41
Polinomiale (grado 3)	6.04	8.84	14.11	13.34	14.22	11.44	7.67	4.77	2.47	6.54	2.50	3.82
Random Forest	5.80	10.29	15.10	15.86	17.13	12.85	8.01	5.39	2.91	5.35	2.56	4.23
Gradient Boosting	6.16	10.34	16.02	22.09	21.02	16.97	10.27	4.77	2.80	5.51	3.87	5.83
SVR (lineare)	7.25	8.66	14.60	13.95	15.03	11.13	7.55	4.39	2.47	4.52	2.51	3.23
SVR (polinomiale)	8.21	10.45	15.33	17.05	20.15	13.55	7.70	4.68	2.50	4.57	2.40	3.37
SVR (RBF)	5.11	9.84	13.65	14.44	15.01	13.25	8.53	5.03	2.65	6.12	2.73	4.88
KernelRidge	5.04	8.23	13.60	13.52	14.06	10.59	7.31	4.79	2.37	4.65	2.52	3.30

Tabella 3.13: Risultati della calibrazione PM₁₀ con cadenza mensile (RMSE)

Infine, la tabella 3.14 riporta i coefficienti delle curve di regressione ottenuti applicando i modelli al dataset PM₁₀. Anche in questo caso i coefficienti a , b , c , d , e si riferiscono ad una equazione polinomiale generica del tipo $y = a + bx + cx^2 + dx^3 + ex^4$, dove x rappresenta il dato grezzo e y il dato calibrato.

Modello di regressione	a	b	c	d	e
Lineare	14.11	13.61	/	/	/
Lineare robusto (Huber)	14.37	13.25	/	/	/
Lineare avanzato (Cook)	10.98	11.32	/	/	/
Lasso	13.98	13.75	/	/	/
Ridge	13.93	13.80	/	/	/
Polinomiale (grado 2)	6.78	25.69	-2.58	/	/
Polinomiale (grado 3)	5.80	28.01	-3.72	0.14	/
Polinomiale (grado 4)	4.92	30.76	-5.97	0.76	-0.05

Tabella 3.14: Coefficienti della calibrazione PM₁₀

3.5 Validazione

Un modello che si adatta bene ai dati potrebbe non avere lo stesso successo nella previsione di nuove osservazioni: fattori influenti che erano sconosciuti durante la fase di costruzione del modello possono influenzare in modo significativo le nuove osservazioni, rendendo le previsioni meno accurate.

Per questo motivo, la corretta convalida di un modello sviluppato per prevedere nuove osservazioni dovrebbe sempre implicare una fase di validazione fatta sul campo (*field validation*).

Una delle procedure per validare un modello di regressione consiste nel testarlo su nuove osservazioni (non viste in fase di addestramento) per poi valutare se le prestazioni predittive del modello siano effettivamente in linea

con le aspettative. In questo contesto è ragionevole aspettarsi che la predizione dei modelli risulti coerente con le concentrazioni attese in determinati periodi, ad esempio in base alla **stagionalità**. Ad esempio nel periodo estivo sono più alti i livelli di O₃ (dovuti alla maggiore radiazione solare), mentre nel periodo invernale salgono i valori di NO₂ e PM, dovuti sia al traffico che agli impianti di riscaldamento.

Per svolgere la fase di validazione è stato quindi considerato il periodo dal 01/09/21 al 20/11/21, di cui sono disponibili sia le misurazioni della centralina SMART16 di AirQino (ottenute con query sul database) che della stazione ARPAT di Capannori (disponibili sul sito ufficiale tramite API [5]).

Le tabelle 3.15 e 3.16 riassumono i risultati, in termini di R^2 e RMSE, del confronto di misurazione della concentrazione di PM_{2.5} e PM₁₀ della centralina SMART16 AirQino nelle due configurazioni (ovvero con sensori calibrati e non calibrati) nel periodo di validazione.

	R²	RMSE (µg/m³)
Calibrato	0.633	4.522
Non calibrato	0.582	4.954

Tabella 3.15: Statistiche a confronto nella misurazione delle concentrazioni medie di PM_{2.5} tra stazione AirQino calibrata e non calibrata rispetto alla stazione di riferimento ARPAT (periodo di validazione dal 01/09/21 al 20/11/21).

	R²	RMSE ($\mu\text{g}/\text{m}^3$)
Calibrato	0.494	5.711
Non calibrato	0.431	6.942

Tabella 3.16: Confronto nella misurazione delle concentrazioni medie di PM₁₀ tra stazione AirQino calibrata e non calibrata rispetto alla stazione di riferimento ARPAT (periodo di validazione dal 01/09/21 al 20/11/21).

Le figure 3.40 e 3.41 mostrano l'andamento temporale delle misurazioni di PM_{2.5} e PM₁₀, calibrate e non calibrate, della stazione AirQino rispetto alle osservazioni della stazione ARPAT nel periodo di validazione.

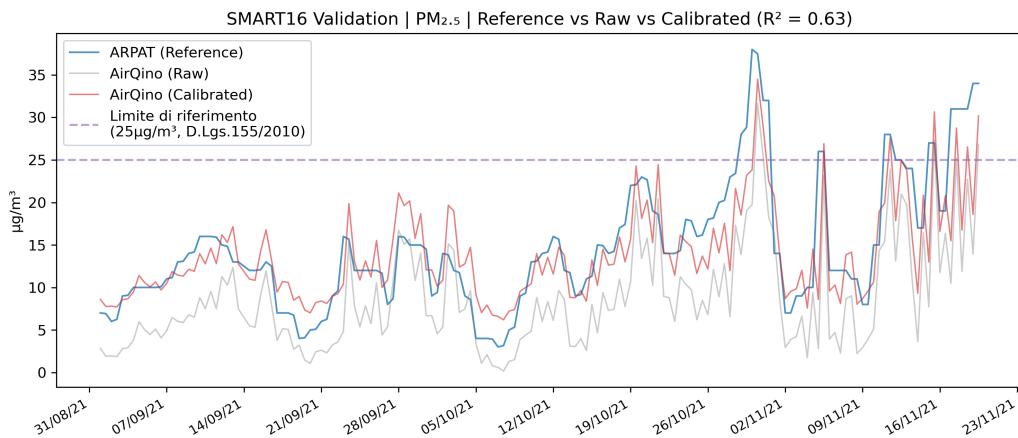


Figura 3.40: Confronto tra l'andamento temporale delle concentrazioni di PM_{2.5} misurate a Capannori (LU) dalla stazione AirQino calibrata e non calibrata con riferimento alla stazione ARPAT (medie a otto ore, periodo dal 01/09/21 al 20/11/21)

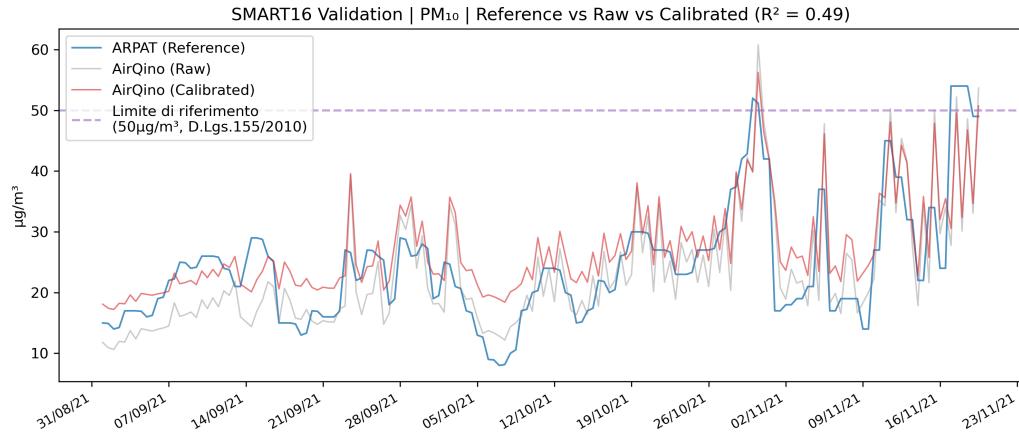


Figura 3.41: Confronto tra l'andamento temporale delle concentrazioni di PM_{10} misurate a Capannori (LU) dalla stazione AirQino calibrata e non calibrata con riferimento alla stazione ARPAT (medie a otto ore, periodo dal 01/09/21 al 20/11/21)

3.6 Discussione

Tutti risultati della fase di calibrazione hanno evidenziato un miglioramento generale nella misura delle concentrazioni di NO_2 e PM , sia nei valori di R^2 (da 0.582 a 0.633 per $\text{PM}_{2.5}$ e da 0.431 a 0.494 per PM_{10}) che per RMSE (da 4.954 a 4.522 $\mu\text{g}/\text{m}^3$ per $\text{PM}_{2.5}$ e da 6.942 a 5.711 $\mu\text{g}/\text{m}^3$ per PM_{10}). I modelli di regressione migliori sono riportati nella tabella 3.17, insieme ai coefficienti corrispondenti. In tutti i casi, il modello di regressione lineare *avanzata*, con rilevamento e rimozione di outlier tramite distanza di Cook (3.3.4.3), si è dimostrato essere il più accurato.

Inquinante	Modello di regressione	a	b	R^2	RMSE ($\mu\text{g}/\text{m}^3$)
NO_2	Lineare avanzato (Cook)	6.84	9.56	0.373	9.447
$\text{PM}_{2.5}$	Lineare avanzato (Cook)	10.31	7.67	0.801	5.894
PM_{10}	Lineare avanzato (Cook)	10.98	11.32	0.785	7.103

Tabella 3.17: Riepilogo dei migliori modelli di regressione, coefficienti ($y = a + bx$) e risultati rilevati dalla calibrazione di sensori NO_2 , $\text{PM}_{2.5}$ e PM_{10}

Dai confronti sull’andamento temporale (figure 3.40 e 3.41) risulta evidente che gran parte delle osservazioni ARPAT $\text{PM}_{2.5}$ e PM_{10} vengono sottostimate dai sensori AirQino non calibrati. Questa discrepanza viene tuttavia ridotta significativamente dal processo di calibrazione che in molti casi migliora le performance del sensore in termini di correlazione.

Modelli di regressione polinomiale così come altri modelli non lineari (Random Forest, Gradient Boosting, SVR, KernelRidge) non hanno portato miglioramenti rispetto ai modelli lineari. In particolare da questa analisi è risultato evidente come la presenza di *outlier* influisca significativamente sulle performance del modello, e strategie che implicano tecniche di rimozione di outlier abbiano riportato performance migliori rispetto a tecniche più complesse (anche non lineari).

In generale, i risultati per i $\text{PM}_{2.5}$ e PM_{10} sono risultati di gran lunga migliori rispetto alla calibrazione dei sensori di NO_2 . Questo si può ricondurre in parte alla diversa frequenza dei dati (medie orarie per NO_2 , medie a otto ore per i PM che riducono la rumorosità dei dati), e in parte anche alla natura e alle differenze dei sensori stessi (vedi 1.3.2), questi infatti hanno un principio fisico di funzionamento diverso (**SDS011** per i PM è ad infrarossi, più costoso e con output calibrato di fabbrica, mentre il **MiCS-2714** per NO_2 funziona

per ossidazione e richiede calibrazione). La difficoltà di utilizzare sensori low cost per i gassosi è ampiamente dimostrata anche in letteratura [60]. Inoltre questo tipo di sensori possono essere soggetti a contaminazione incrociata con altri gas, che interferiscono sia con la sensibilità che con la precisione della misurazione [61].

Infine, i risultati per $\text{PM}_{2.5}$ si sono dimostrati essere sempre tendenzialmente migliori dei corrispettivi PM_{10} . Questo risulta in linea con quanto riportato in studi correlati [12] e altre reti di monitoraggio ambientale low-cost [62], [63], [64].

Capitolo 4

Interfaccia di calibrazione

Questo capitolo è incentrato sulle motivazioni, lo sviluppo e la realizzazione di una dashboard web per la calibrazione multipla di centraline AirQino.

4.1 Motivazioni

I sensori possono essere soggetti a molteplici processi di calibrazioni durante loro ciclo di vita (vedi 3.1). Lo scopo è quello di trovare i coefficienti della curva che meglio approssima la relazione tra il segnale del sensore e un segnale di riferimento, da cui poter ricavare il valore calibrato a partire dal dato grezzo.

Attualmente, per tutte le centraline e sensori AirQino, il dato calibrato viene ottenuto con la seguente formula:

$$y(t) = k_1x(t) + k_2x^2(t) + k_3T_{int}(t) + k_4$$

Dove:

- x è il **dato grezzo**;
- $k1 \dots k4$ sono i **coefficienti di calibrazione** ottenuti applicando un modello di regressione ai valori misurati rispetto ad un segnale di riferimento;
- T_{int} è la **temperatura interna** della centralina al tempo t ;
- y è il **valore calibrato**.

I coefficienti $k1 \dots k4$ sono memorizzati nella tabella *field_calibration* del database AirQino. Si ha un set di coefficienti per ogni terna centralina, sessione, sensore: in questo modo si possono utilizzare parametri di calibrazione diversi per *sessions* diverse. I dati possono essere inseriti attraverso un form accessibile dalle pagine di amministrazione.

Questa soluzione risulta però poco flessibile, in quanto ogni singolo coefficiente di ogni sessione e di ogni centralina deve essere inserito manualmente. Come soluzione a questo problema è stato quindi previsto lo sviluppo di un componente di calibrazione, realizzato come servizio web, che permetta di superare i limiti sopra descritti, in particolare salvando sul database i coefficienti di calibrazione in modo massivo (ovvero ricevendo più set di coefficienti o applicando un set a più centraline).

4.2 Requisiti

Il sistema di calibrazione multipla delle centraline AirQino deve prevedere, oltre alla funzionalità base di salvare più coefficienti contemporaneamente, altri requisiti:

- la possibilità di aprire una nuova *sessione* per ciascuna centralina interessata. Questo risulta utile nel caso in cui non si voglia sovrascrivere i coefficienti già esistenti, ma salvarne dei nuovi;
- Se la centralina o il sensore specificati non sono presenti nel database il processo deve continuare senza interrompersi, mostrando l'errore soltanto alla fine in una pagina di riepilogo;
- Se la sessione non è presente nel database, il sistema deve aprirne una nuova prima di caricare i coefficienti;
- Il sistema deve proteggere l'accesso da parte di utenti di terze parti implementando un meccanismo di autenticazione.

Come input al servizio è stato previsto l'uso di file csv così strutturati:

id_airqino	param	formula	regression_type	score	a	b	c	d	q
smart42	O3	y=a*x1+q	cook	0.99	-0.1936	nan	nan	nan	283.
smart42	NO2	y=a*x1+q	cook	0.55	0.04068	nan	nan	nan	-7.75
smart42	CO	y=a*x+b*x^2+q	poly 2	0.39	0.00952	-0.00001633	nan	nan	-1.321
smart111	O3	y=a*x1+c*x2+q	temp_multiregression	0.29	-0.1342	nan	3.451	nan	-6.56
smart111	NO2	y=a*x1+c*x2+q	temp_multiregression	0.19	0.1029	nan	-0.7563	nan	15.47
smart111	CO	y=a*x1+c*x2+q	temp_multiregression	0.1	0.0002663	nan	-0.003038	nan	0.1405
smart41	O3	y=a*x1+c*x2+q	temp_multiregression	0.73	-0.2734	nan	9.24	nan	-81.2
smart41	NO2	y=a*x1+q	cook	0.41	0.1266	nan	nan	nan	-1.944
smart41	CO	y=a*x1+c*x2+q	temp_multiregression	0.01	0.000774	nan	-0.004684	nan	0.08716

Figura 4.1: Esempio di file csv per calibrazione di centraline AirQino

Dove:

- La colonna **id_airqino** indica il nome della centralina AirQino;
- La colonna **param** indica il sensore da calibrare;
- Le colonne **a**, **b**, **c** e **q** rappresentano i coefficienti di calibrazione (rispettivamente k_1 , k_2 , k_3 e k_4 della formula riportata in 4.1);

- La colonna **score** indica il coefficiente di determinazione R^2 ;
- Le colonne **formula**, **regression_type** e **d** sono opzionali e riguardano il tipo di regressione e altri parametri al momento inutilizzati (non influiscono sul risultato finale).

4.3 Tecnologie utilizzate

4.3.1 Backend

- **Flask** [65] è un framework scritto in Python per la realizzazione di applicazioni web. Ha la caratteristica di essere molto semplice e versatile nella gestione del routing e la creazione di API REST;
- **Pandas** [56] è una libreria Python che fornisce strumenti di analisi e manipolazione dei dati. Utilizzata nel progetto per leggere i dati da un file csv;
- **Psycopg2** [66] è una libreria Python per la connessione a database che utilizzano Postgres (come quello di AirQino);
- **Docker** [33] è una tecnologia di containerizzazione che consente di organizzare le applicazioni in *container*, componenti isolati che combinano il codice sorgente dell'applicazione con le librerie del sistema operativo e le dipendenze necessarie per eseguire il codice in qualsiasi ambiente.

4.3.2 Frontend

- **Angular** [16] è un framework di applicazioni web, basato su linguaggio TypeScript, che consente la creazione di Single Page Application (SPA) reattive;
- **Bootstrap** [67] è un framework CSS dedicato allo sviluppo frontend, mettendo a disposizione tipografia, pulsanti, navigazione e altri componenti dell’interfaccia web;
- **ng-bootstrap** [68] è un set di widget e componenti Angular reattivi che utilizzano lo stile Bootstrap.

4.4 Sviluppo

Il risultato finale è stato reso disponibile al seguente indirizzo:

<https://airqino-calibration.magentalab.it>.

Di seguito ne vengono descritte le caratteristiche principali.

4.4.1 Interfaccia

L’interfaccia, dopo il primo caricamento, si presenta come in figura 4.2, con un *form* che permette di scegliere e caricare il file csv, una *checkbox* che consente di aprire una nuova sessione per tutte le centraline, e un pulsante per avviare il caricamento.

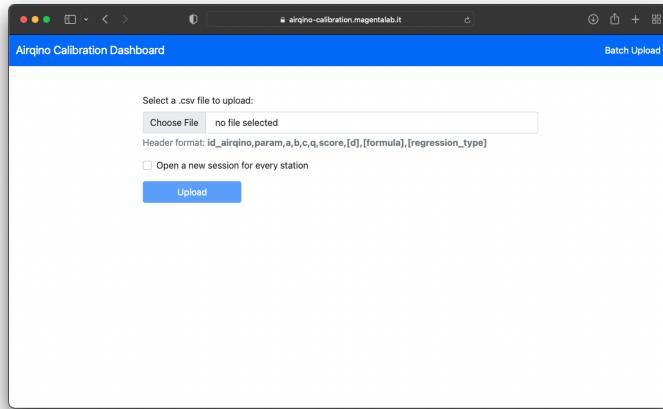


Figura 4.2: Interfaccia iniziale dell'applicazione

Se il formato del file csv scelto non è compatibile con quello previsto (vedi 4.2), la pagina restituisce un errore informativo dove viene indicata esattamente quali colonne mancano nell'*header* (figura 4.3).

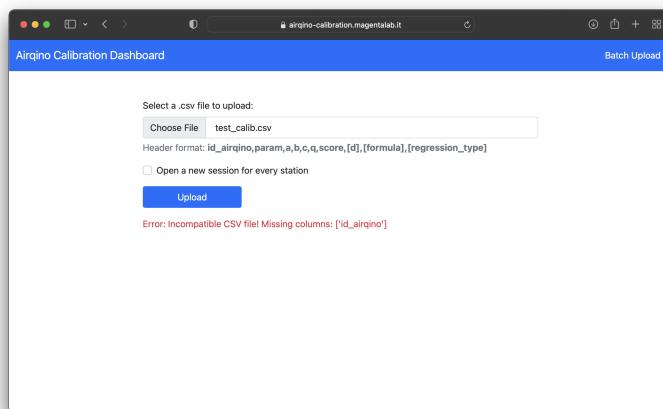


Figura 4.3: Errore informativo mostrato in caso di caricamento di un file csv con header non riconosciuto

Una pagina simile viene mostrata nel caso in cui l'applicazione non riesca a connettersi al database di AirQino (ad esempio se questo è in fase di

mantenimento).

Se il file csv caricato ha il formato corretto, e il database è raggiungibile, viene avviata la procedura di caricamento multiplo dei coefficienti. Al termine dell'operazione, viene mostrato un riepilogo in forma di tabella dove ogni riga corrisponde ad un sensore (figura 4.4).

#	Result	Station	Sensor	R	k1	k2	k3	k4	Last SQL Query
1	Updated	smart42	O3	0.99	-0.1936		283		UPDATE field_calibration SET k1=-0.1936,
2	Updated	smart42	NO2	0.55	0.04068		-7.75		UPDATE field_calibration SET k1=0.04068,
3	Updated	smart42	CO	0.39	0.00952	-0.00001633		-1.321	UPDATE field_calibration SET k1=0.00952,
4	Updated	smart111	O3	0.29	-0.1342		3.451	-6.56	UPDATE field_calibration SET k1=-0.1342,
5	Updated	smart111	NO2	0.19	0.1029		-0.7563	15.47	UPDATE field_calibration SET k1=0.1029, i
6	Updated	smart111	CO	0.1	0.0002663		-0.003038	01405	UPDATE field_calibration SET k1=0.0002663
7	Updated	smart41	O2	0.72	0.2724		0.24	-81.2	UPDATE field_calibration SET k1=-0.2724

Figura 4.4: Interfaccia in caso di caricamento corretto del file csv

La tabella, mostrata nel dettaglio in figura 4.5, presenta le seguenti colonne:

- **Result:** è l'esito del caricamento sul database (*Updated* se il dato era già presente ed è stato sovrascritto, *Created* se il dato è stato creato per la prima volta, e *Failed* se l'operazione invece è fallita per quel sensore);
- **Station, Sensor, R, k1, k2, k3, k4** sono gli stessi parametri passati nel file csv di input;
- **Last SQL Query:** è l'ultima query effettuata sul database per quel sensore, utile in per verificarne la correttezza e/o capire il motivo di eventuali errori (lo screenshot completo è riportato in figura 4.6).

#	Result	Station	Sensor	R	k1	k2	k3	k4	Last SQL Query
1	Updated	smart42	O3	0.99	-0.1936		283		UPDATE field_calibration SET k1=-0.1936;
2	Updated	smart42	NO2	0.55	0.04068		-7.75		UPDATE field_calibration SET k1=0.04068;
3	Updated	smart42	CO	0.39	0.00952	-0.00001633	-1.321		UPDATE field_calibration SET k1=0.00952;
4	Updated	smart111	O3	0.29	-0.1342		3.451	-6.56	UPDATE field_calibration SET k1=-0.1342;
5	Updated	smart111	NO2	0.19	0.1029		-0.7563	15.47	UPDATE field_calibration SET k1=0.1029;
6	Updated	smart111	CO	0.1	0.0002663		-0.003038	0.1405	UPDATE field_calibration SET k1=0.0002663;
7	Updated	smart41	O3	0.73	-0.2734		9.24	-81.2	UPDATE field_calibration SET k1=-0.2734;
8	Updated	smart41	NO2	0.41	0.1266		-1.944		UPDATE field_calibration SET k1=0.1266;
9	Updated	smart41	CO	0.01	0.000774		-0.004684	0.08716	UPDATE field_calibration SET k1=0.000774;
10	Updated	smart99	O3	0.19	0.06494		1.866	-42.44	UPDATE field_calibration SET k1=0.06494;
11	Updated	smart99	NO2	0.27	0.06793		-6.957		UPDATE field_calibration SET k1=0.06793;
12	Updated	smart99	CO	0.03	-0.01342	0.0000284	1.688		UPDATE field_calibration SET k1=-0.01342;
13	Updated	smart14	O3	0.24	-4.164		6.547	299.2	UPDATE field_calibration SET k1=-4.164;
14	Updated	smart14	NO2	0.21	0.09155		-11.65		UPDATE field_calibration SET k1=0.09155;
15	Updated	smart14	CO	0.01	0.001867		0.015526	-1.068	UPDATE field_calibration SET k1=0.001867;
16	Updated	smart90	O3	0.43	0.01831		-6.08	275.2	UPDATE field_calibration SET k1=0.01831;
17	Updated	smart90	NO2	0.21	-0.003250		-0.296	26	UPDATE field_calibration SET k1=-0.003250;

Figura 4.5: Dettaglio della tabella di riepilogo del caricamento dei coefficienti

Results		0 rows created, 93 rows updated, 0 rows failed	Search
		Last SQL Query	
	283	UPDATE field_calibration SET k1=-0.1936, k2=0, k3=0, k4=283.0, r=0.99 WHERE id=34244715;	
	-7.75	UPDATE field_calibration SET k1=0.04068, k2=0, k3=0, k4=-7.75, r=0.55 WHERE id=34244716;	
I01633	-1.321	UPDATE field_calibration SET k1=0.00952, k2=1.633e-05, k3=0, k4=-1.321, r=0.39 WHERE id=34244717;	
	3.451	UPDATE field_calibration SET k1=-0.1342, k2=0, k3=3.451, k4=-6.56, r=0.29 WHERE id=31935862;	
	-0.7563	UPDATE field_calibration SET k1=0.1029, k2=0, k3=-0.7563, k4=15.47, r=0.19 WHERE id=31935863;	
	0.003038	UPDATE field_calibration SET k1=0.0002663, k2=0, k3=-0.003038, k4=0.1405, r=0.01 WHERE id=31935864;	
	0.1405	UPDATE field_calibration SET k1=-0.2734, k2=0, k3=-9.24, k4=-81.2, r=0.73 WHERE id=34336978;	
	9.24	UPDATE field_calibration SET k1=-0.2734, k2=0, k3=-9.24, k4=-81.2, r=0.73 WHERE id=34336978;	
	-1.944	UPDATE field_calibration SET k1=0.1266, k2=0, k3=0, k4=-1.944, r=0.41 WHERE id=34336979;	
	-0.004684	UPDATE field_calibration SET k1=0.000774, k2=0, k3=-0.004684, k4=0.08716, r=0.01 WHERE id=34336980;	
	0.08716	UPDATE field_calibration SET k1=0.06494, k2=0, k3=1.866, k4=-42.44, r=0.19 WHERE id=31935898;	
	1.866	UPDATE field_calibration SET k1=0.06494, k2=0, k3=1.866, k4=-42.44, r=0.19 WHERE id=31935898;	
	-6.957	UPDATE field_calibration SET k1=0.06793, k2=0, k3=0, k4=-6.957, r=0.27 WHERE id=31935899;	
I284	1.688	UPDATE field_calibration SET k1=-0.01342, k2=2.84e-05, k3=0, k4=1.688, r=0.03 WHERE id=34244409;	
	6.547	UPDATE field_calibration SET k1=-4.164, k2=0, k3=6.547, k4=299.2, r=0.24 WHERE id=34244409;	
	-11.65	UPDATE field_calibration SET k1=0.09155, k2=0, k3=0, k4=-11.65, r=0.21 WHERE id=34244410;	
	0.015526	UPDATE field_calibration SET k1=0.001867, k2=0, k3=0.015526, k4=-1.068, r=0.01 WHERE id=34244411;	
	-1.068	UPDATE field_calibration SET k1=0.01831, k2=0, k3=-6.08, k4=275.2, r=0.43 WHERE id=34333737;	
	-6.08	UPDATE field_calibration SET k1=-0.003359, k2=0, k3=-0.296, k4=26.0, r=0.01 WHERE id=34333736;	
	-0.296	UPDATE field_calibration SET k1=-0.003359, k2=0, k3=-0.296, k4=26.0, r=0.01 WHERE id=34333736;	

Figura 4.6: Dettaglio completo per la colonna *Last SQL Query* della tabella riassuntiva di caricamento dei coefficienti

La tabella è ordinabile per colonne (tramite un click su ciascuna di esse) e filtrabile con ricerca *full text* su qualsiasi campo (figura 4.7).

#	Result	Station	Sensor	R	k1	k2	k3	k4	Last SQL Query
1	Updated	smart42	03	0.99	-0.1936				283 UPDATE field_calibration SET k1=-0.1936,
4	Updated	smart111	03	0.29	-0.1342				4.391 -6.58 UPDATE field_calibration SET k1=-0.1342,
7	Updated	smart41	03	0.73	-0.2734				0.24 -81.2 UPDATE field_calibration SET k1=-0.2734,
10	Updated	smart99	03	0.19	0.06494				1.866 -42.44 UPDATE field_calibration SET k1=0.06494,
13	Updated	smart14	03	0.24	-0.164				6.547 299.2 UPDATE field_calibration SET k1=-0.164,
16	Updated	smart90		0.43	0.02831				-6.0 275.2 UPDATE field_calibration SET k1=0.02831,
19	Updated	smart104	03	0.28	-0.2411				4.33 304.8 UPDATE field_calibration SET k1=-0.2411,
22	Updated	smart170	03	0.43	0.02831				-20.4 304.8 UPDATE field_calibration SET k1=0.02831,
25	Updated	smart62	03	0.51	-0.3197				8.28 -192.1 UPDATE field_calibration SET k1=-0.3197,
28	Updated	smart62	03	0.79	-0.179				5.234 76.44 UPDATE field_calibration SET k1=-0.179,
31	Updated	smart98	03	0.35	-0.2107				719 -63.8 UPDATE field_calibration SET k1=-0.2107,
34	Updated	smart99	03	0.3	-0.1046				5.246 -65.7 UPDATE field_calibration SET k1=-0.1046,
37	Updated	smart102	03	0.19	-0.000979				3.735 -32.2 UPDATE field_calibration SET k1=-0.000979,
40	Updated	smart21	03	0.13	-0.281				4.17 116.44 UPDATE field_calibration SET k1=-0.281,
43	Updated	smart40	03	0.82	-0.1342				5.34 -83.4 UPDATE field_calibration SET k1=-0.1342,
46	Updated	smart103	03	0.14	0.0538				2.08 -21.34 UPDATE field_calibration SET k1=0.0538,
49	Updated	smart96	03	0.89	2.13				-152 UPDATE field_calibration SET k1=2.13, k2

(a) Ordinamento per colonne

(b) Filtro con ricerca *full text*

Figura 4.7: Esempio di ordinamento e ricerca della tabella riassuntiva di caricamento dei coefficienti

In caso di esito negativo per un sensore (ad esempio se il sensore o la centralina specificati non esistono nel database), la riga corrispondente risulta colorata di rosso e la colonna **Result** fornisce un aiuto nel capire il motivo (es. "Failed: station SMART999 does not exist"). Se invece i coefficienti non erano presenti nel database e sono stati caricati per la prima volta, la riga risulta colorata di azzurro. La figura 4.8 riporta un esempio di diversa colorazione delle righe in base all'esito: verde per coefficienti aggiornati, rosso per caricamento fallito e azzurro per coefficienti aggiunti per la prima volta.

#	Result	Station	Sensor	R	k1	k2	k3	k4	d	Last SQL Query
1	Updated	smart41	NO2	0.41	0.1266	0.1323	0.6323	-1.944	-0.3245	UPDATE field_calibration SET k1=0.1266, k2=0.1323, k3=0.6323, k4=-1.944, d=-0.3245 WHERE id=smart41;
2	Failed: station SMART999 does not exist	smart999	NO2	0.73	0.1598	0.7429	0.1243	-1.098	-0.7531	SELECT id FROM stations WHERE name='SMART999';
3	Created	smart104	CO2	0.51	0.1366	0.1546	0.2435	-0.3245	-0.8754	INSERT INTO field_calibration (id, station, sensor, r, k1, k2, k3, k4, d) VALUES (smart104, 'CO2', 0.51, 0.1366, 0.1546, 0.2435, -0.3245, -0.8754);

Figura 4.8: Esempio di colorazione delle righe della tabella

4.4.2 Autenticazione

Per la gestione dell'autenticazione è stato scelto il software **Keycloak** [69], che risultava essere già installato sul server di Magenta. Keycloak è un prodotto software open source che abilita l'autenticazione *Single Sign-On* (SSO) per applicazioni e servizi. È scritto in Java e supporta i protocolli di federazione delle identità per impostazione predefinita SAML v2 e OpenID Connect (OIDC) o OAuth2. [70]

In particolare, Keycloak consente a un'applicazione di delegare la propria autenticazione ad un server web predisposto, consentendo allo sviluppatore di concentrarsi sulle funzionalità, senza doversi preoccupare di aspetti di sicurezza e crittografia. Keycloak infatti include un server e un agente: l'agente è installato sulle applicazioni che richiedono l'autenticazione, mentre il server gestisce tutte le richieste di autenticazione. Quando un utente tenta di accedere a una applicazione protetta da Keycloak, l'agente verifica se l'utente è autenticato e, in caso affermativo, fornisce le credenziali appropriate all'applicazione.

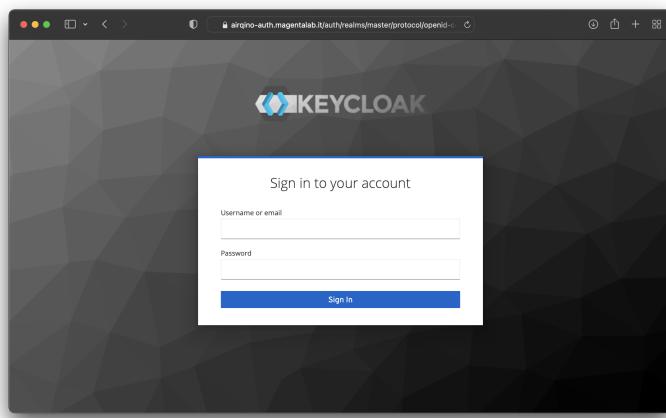


Figura 4.9: Autenticazione con Keycloak

Keycloak è stato integrato nell'applicazione con l'aiuto della libreria open source `keycloak-angular` [71], un wrapper della libreria ufficiale JavaScript `keycloak-js` [72] che rende semplice l'utilizzo del software per applicazioni Angular. I principali parametri da impostare sono:

- **url**: l'url del server Keycloak;
- **realm**: il nome del *realm*, impostato nel pannello di amministrazione, che serve a gestire un insieme di utenti, credenziali e ruoli;
- **clientId**: l'identificativo del *client*, cioè l'entità che richiede a Keycloak il permesso di autenticare un utente, anche questo impostato nel pannello di amministrazione;
- **initOptions**: le opzioni di configurazione, ad esempio è possibile specificare se l'autenticazione avviene automaticamente al caricamento della pagina (come in questo caso) oppure scatta in un secondo momento (utile se ad esempio l'applicazione fornisce funzionalità anche senza loggare l'utente).

È possibile inizializzare la configurazione in un file JavaScript di questo tipo (file `keycloak-init.js`, da caricare poi all'avvio del modulo principale dell'applicazione):

```
1 import {KeycloakService} from "keycloak-angular";  
2  
3 export function initializeKeycloak(keycloak: KeycloakService) {  
4   return () =>  
5     keycloak.init({  
6       config: {  
7         url: 'https://airqino-auth.magentalab.it/auth',  
8         realm: 'master',
```

```
9     clientId: 'calibration',
10    },
11    initOptions: {
12      onLoad: 'login-required',
13      flow: 'implicit'
14    },
15  });
16 }
```

Questi parametri sono sufficienti a definire l'integrazione tra applicazione e server Keycloak, garantendo in maniera efficiente funzionalità di autenticazione e autorizzazione.

4.4.3 CI/CD e deploy automatico

L'integrazione continua (*continuous integration*, CI o CI/CD) è una metodologia di sviluppo software *agile*, spesso utilizzata in combinazione con approccio **DevOps**, che prevede la continua e costante integrazione dei cambiamenti effettuati dagli sviluppatori all'interno di un codice sorgente (figura 4.10).

Con il termine CD (*continuous delivery*) si indica invece la distribuzione e/o il deployment continuo, un concetto comunque correlato alla CI, ovvero processo con il quale le modifiche apportate da uno sviluppatore all'applicazione vengono automaticamente testate e caricate in una repository, dalla quale vengono automaticamente distribuite in un ambiente di produzione. [73]

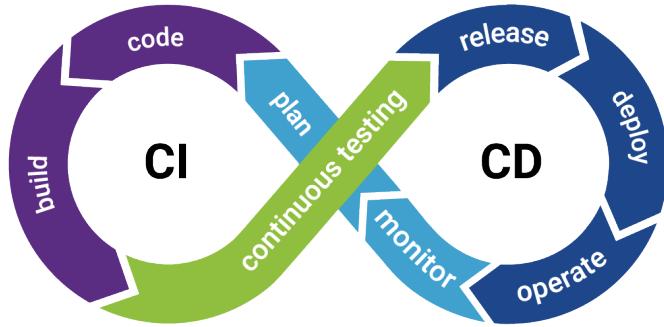


Figura 4.10: Schema di funzionamento CI/CD

Fonte: flagship.io

La CI/CD porta diversi vantaggi, tra cui:

- Solidità del prodotto, poiché le integrazioni sono frequenti e anche i test vengono eseguiti in anticipo;
- Aumento della qualità del codice;
- Consegnà più rapida del software.

Lo strumento di CI scelto per questa applicazione è **Jenkins** [74]. Si tratta di un software open source di integrazione continua, scritto in Java, che permette di automatizzare il processo di integrazione dei cambiamenti all'interno di un codice sorgente, con la possibilità di eseguire una serie di controlli per verificarne la correttezza.

Tra le varie caratteristiche, Jenkins offre un sistema chiamato *pipeline*, ovvero uno script (chiamato *Jenkinsfile*) che fornisce a Jenkins una serie di lavori (*stage*) da eseguire in modo simile a una pipeline (uno dopo l'altro).

Il *Jenkinsfile* utilizzato per questo progetto contiene tre stage, ognuno con i propri comandi, ed è riportato di seguito (ridotto per semplicità):

```
1 pipeline {
2     agent any
3     stages {
4         stage('Build Angular frontend') {
5             steps {
6                 sh 'npm install && ng build --configuration production'
7             }
8         }
9         stage('Docker deploy backend') {
10            steps {
11                sh 'flask run'
12            }
13        }
14        stage('Deploy frontend') {
15            steps {
16                'scp -r dist/* ${HOST}:${PATH}/'
17            }
18        }
19    }
20 }
```

In particolare:

- Il primo stage esegue il build del frontend, installando le dipendenze con `npm install` e avviando il processo con `ng build` in configurazione di produzione. Questo genera l'artefatto finale all'interno della cartella `dist`;
- Il secondo stage esegue il rilascio del backend con il comando `flask run`;

- Infine, il terzo stage rilascia il frontend copiando i contenuti della cartella `dist` nella cartella corretta del server di produzione.

L’interfaccia web di Jenkins permette inoltre di monitorare l’andamento dei vari step e di verificare eventuali errori. Jenkins può essere anche combinato con Docker [33] in modo da garantire processi di rilascio automatizzati e allo stesso tempo avere l’isolamento delle dipendenze, che verranno installate solo sul container e non su tutto il server.

Conclusioni e sviluppi futuri

Lo studio effettuato in questo lavoro di tesi ha evidenziato come le reti di sensori *low-cost* per il monitoraggio della qualità dell'aria, come **AirQino** (1.3), possano rappresentare una soluzione efficace per il rilevamento dell'inquinamento atmosferico ad alta risoluzione sia temporale che spaziale. Queste soluzioni possono integrarsi con le reti di monitoraggio regionali già esistenti, che garantiscono misurazioni più accurate ma con un elevato costo di strumentazione, fornendo un quadro più completo della qualità dell'aria in ambiente urbano.

I risultati ottenuti mostrano che anche con sensori a basso costo è possibile misurare inquinanti come $\text{PM}_{2.5}$ e PM_{10} con una buona accuratezza, ancora migliore se si ha a disposizione un segnale di riferimento ARPAT con cui correggere i dati proveniente dai sensori, applicando tecniche di regressione robusta ai modelli di regressione statistici. Per sensori di rilevamento di inquinanti gassosi (come NO_2), invece, questo è risultato più complicato, anche se l'applicazione di tecniche di regressione robusta in fase di calibrazione ha comunque riportato miglioramenti significativi in termini di accuratezza.

Uno sviluppo futuro potrebbe essere quello di perfezionare il processo di calibrazione delle centraline, ad esempio aggiungendo ulteriori variabili in input modello di regressione, come la **temperatura** e l'**umidità** relativa, e verificare se questo porti ad un aumento dell'accuratezza della predizione.

L'inclusione di nuovi predittori potrebbe migliorare anche la performance sul sensore di NO_2 . Questa idea trova riscontro nella figura 4.11, che mostra la matrice di correlazione tra $\text{PM}_{2.5}$, PM_{10} , temperatura e umidità misurate dalla centralina SMART16 nel periodo 18/08/2020 - 30/08/2021. Dai dati si nota infatti che la temperatura tende a correlare negativamente con i PM (all'aumentare della temperatura le polveri sottili tendono a calare), mentre l'umidità presenta l'effetto contrario. Inoltre fattori esterni come temperatura e umidità possono anche influenzare l'elettronica stessa del sensore.

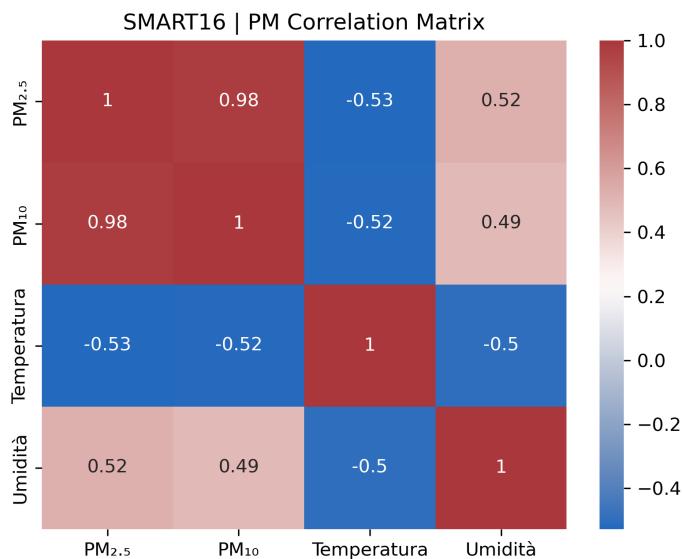


Figura 4.11: Correlazione tra polveri sottili, temperatura e umidità misurate dalla centralina SMART16 (AirQino) nel periodo 18/08/2020 - 30/08/2021

Inoltre, come ulteriore sviluppo si potrebbe prevedere la realizzazione di una **procedura di allerta** sul segnale delle centraline. In particolare, effettuando un controllo in tempo reale sull'andamento del segnale e monitorando la differenza con il segnale di riferimento (es. stazione ARPAT), il sistema potrebbe inviare un messaggio di allerta nel caso in cui il segnale inizi a deri-

vare, ovvero quando lo scarto tra i due non risulta più costante (ad esempio per l’usura del sensore), segnalando la necessità di una ricalibrazione. Proprio in questo scenario risulta utile l’interfaccia progettata (4.4.1) che permette la ricalibrazione di più centraline contemporaneamente.

Bibliografia

- [1] “Magenta srl.” <https://www.magentalab.it>.
- [2] “Istituto per la BioEconomia (IBE) - CNR.” <https://www.ibe.cnr.it>.
- [3] “Il sistema AirQino.” <https://airqino.it>.
- [4] “Direttiva 2008/50/EC - Un’aria più pulita in Europa.”
<https://eur-lex.europa.eu/legal-content/IT/LSU/?uri=CELEX:32008L0050>.
- [5] “ARPAT - agenzia regionale per la protezione ambientale della Toscana.”
<http://www.arpat.toscana.it/>.
- [6] A. Cavaliere, “Relazione sull’attività svolta nell’anno 2016,” *Antrophic sensors and control networks for smart city*, 2016.
- [7] M. Mead, O. Popoola, G. Stewart, P. Landshoff, M. Calleja, M. Hayes, J. Baldovi, M. McLeod, T. Hodgson, J. Dicks, A. Lewis, J. Cohen, R. Baron, J. Saffell, and R. Jones, “The use of electrochemical sensors for monitoring urban air quality in low-cost, high-density networks,” *Atmospheric Environment*, vol. 70, pp. 186–203, 2013.

- [8] A. Velasco, R. Ferrero, F. Gandino, B. Montrucchio, and M. Rebaudengo, “A mobile and low-cost system for environmental monitoring: A case study,” *Sensors*, vol. 16, no. 5, 2016.
- [9] E. G. Snyder, T. H. Watkins, P. A. Solomon, E. D. Thoma, R. W. Williams, G. S. W. Hagler, D. Shelow, D. A. Hindin, V. J. Kilaru, and P. W. Preuss, “The changing paradigm of air pollution monitoring,” *Environmental Science & Technology*, vol. 47, no. 20, pp. 11369–11377, 2013. PMID: 23980922.
- [10] P. Kumar, L. Morawska, C. Martani, G. Biskos, M. Neophytou, S. Di Sabatino, M. Bell, L. Norford, and R. Britter, “The rise of low-cost sensing for managing air pollution in cities,” *Environment International*, vol. 75, pp. 199–205, 2015.
- [11] A. L. Clements, W. G. Griswold, A. RS, J. E. Johnston, M. M. Herting, J. Thorson, A. Collier-Oxandale, and M. Hannigan, “Low-cost air quality monitoring tools: From research to practice (a workshop summary),” *Sensors*, vol. 17, no. 11, 2017.
- [12] A. Cavaliere, F. Carotenuto, F. Di Gennaro, B. Gioli, G. Gualtieri, F. Martelli, A. Matese, P. Toscano, C. Vagnoli, and A. Zaldei, “Development of low-cost air quality stations for next generation monitoring networks: Calibration and validation of pm2.5 and pm10 sensors,” *Sensors*, vol. 18, no. 9, 2018.
- [13] G. Gualtieri, F. Camilli, A. Cavaliere, T. De Filippis, F. Di Gennaro, S. Di Lonardo, F. Dini, B. Gioli, A. Matese, W. Nunziati, L. Rocchi, P. Toscano, C. Vagnoli, and A. Zaldei, “An integrated low-cost road traffic and air pollution monitoring platform to assess vehicles’ air qua-

- lity impact in urban areas,” *Transportation Research Procedia*, vol. 27, pp. 609–616, 2017. 20th EURO Working Group on Transportation Meeting, EWGT 2017, 4-6 September 2017, Budapest, Hungary.
- [14] “Spring.” <https://spring.io>.
- [15] “Timescale: Time-series data simplified.” <https://www.timescale.com>.
- [16] “Angular.” <https://angular.io>.
- [17] G. Bernardo, “Il sensore sds011 e le polveri sottili,” 2020.
- [18] “Prato Urban Jungle.” <https://pratourbanjungle.it>.
- [19] “SMART Treedom.” <https://smart.treedom.net>.
- [20] “Trafair.” <https://trafair.eu>.
- [21] “Smart Garda Lake.” <https://smartgardalake.it/progetto>.
- [22] “Planetwatch.” <https://planetwatch.io>.
- [23] “Brenner LEC.” <https://brennerlec.life>.
- [24] “Airly.” <https://airly.org>.
- [25] “Aqicn.” <https://aqicn.org>.
- [26] “Iqair.” <https://www.iqair.com>.
- [27] “Decentlab.” <https://decentlab.com>.
- [28] “Hackair.” <https://hackair.eu>.
- [29] “Mysql.” <https://www.mysql.com>.

- [30] “Postgresql.” <https://www.postgresql.org>.
- [31] S. Insausti, *PostgreSQL Streaming Replication - a Deep Dive*. 2018.
- [32] “PostgreSQL Streaming Replication.” <https://www.postgresql.org/docs/current/warm-standby.html#STREAMING-REPLICATION>.
- [33] “Docker.” <https://www.docker.com>.
- [34] “Dockerfile reference.” <https://docs.docker.com/engine/reference/builder/>.
- [35] “Timescale - About continuous aggregates.” <https://docs.timescale.com/timescaledb/latest/how-to-guides/continuous-aggregates/about-continuous-aggregates/>.
- [36] S. D’Amilo, *Progetto e implementazione di un’architettura innovativa per la gestione e visualizzazione dei dati IoT di una filiera produttiva*. 2021.
- [37] “Timescale - Refresh continuous aggregates.” <https://docs.timescale.com/timescaledb/latest/how-to-guides/continuous-aggregates/refresh-policies/>.
- [38] “DustTrak™ DRX Aerosol Monitor 8533.” <https://tsi.com/products/aerosol-and-dust-monitors/dust-monitors/dusttrak-drx-aerosol-monitor-8533/>.
- [39] “Horiba.” <https://www.horiba.com/int/>.
- [40] “Attuazione della direttiva 2008/50/ce relativa alla qualità dell’aria ambiente e per un’aria più pulita in europa.” <https://web.camera.it/parlam/leggi/deleghe/10155d1.htm>.

- [41] E. Belluco, *Excel per la statistica*. Franco Angeli, 2005.
- [42] M. S. Paoletta, *Linear Models and Time-Series Analysis*. Wiley, 2019.
- [43] P. Pozzolo, *Analisi dei residui del modello di regressione lineare*. 2020.
- [44] D. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*. Wiley, 2012.
- [45] J. Neter, M. H. Kutner, C. J. Nachtsheim, and W. Wasserman, *Applied Linear Statistical Models*. Chicago: Irwin, 1996.
- [46] W. Mendenhall and T. Sincich, “A second course in statistics: Regression analysis,” *Journal of the American Statistical Association*, vol. 92, 06 1997.
- [47] R. D. Cook, *Detection of Influential Observation in Linear Regression*. Taylor and Francis, Ltd., 1977.
- [48] “Yellowbrick: Machine Learning Visualization.” <https://www.scikit-yb.org/en/latest/index.html>.
- [49] J. Fox, *Applied Regression Analysis and Generalized Linear Models*. Sage Publications, 2015.
- [50] G. Spanò, *Lasso vs Ridge Regression*. 2017.
- [51] A. Felice, *Applicazione di modelli di Machine Learning ad Albero Decisionale con R per il Credit Scoring nel settore elettronico*. 2021.
- [52] T. K. Ho, *Random decision forests*. 1995.
- [53] C. Cortes, *Support-Vector Networks*. 1995.

- [54] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.
- [55] “Scikit-learn: machine learning in Python.” <https://scikit-learn.org/stable/>.
- [56] “Pandas - Python Data Analysis Library.” <https://pandas.pydata.org>.
- [57] “Matplotlib: Visualization with Python.” <https://matplotlib.org>.
- [58] “Numpy: The fundamental package for scientific computing with Python.” <https://numpy.org>.
- [59] “Seaborn: statistical data visualization.” <https://seaborn.pydata.org>.
- [60] D. Hasenfratz, O. Saukh, and L. Thiele, “On-the-fly calibration of low-cost gas sensors,” 2012.
- [61] “Cross-Contamination of Sensors by Other Gases and the Influence on Detector Accuracy.” <https://braschenvtech.com/wp-content/uploads/2020/06/Cross-Contamination.pdf>.
- [62] N. Zíková, P. Hopke, and A. Ferro, “Evaluation of new low-cost particle monitors for pm2.5 concentrations measurements,” *Journal of Aerosol Science*, vol. 105, 11 2016.
- [63] A. Mukherjee, L. G. Stanton, A. R. Graham, and P. T. Roberts, “Assessing the utility of low-cost particulate matter sensors over a 12-week period in the cuyama valley of california,” *Sensors*, vol. 17, no. 8, 2017.

- [64] N. Zikova, M. Masiol, D. C. Chalupa, D. Q. Rich, A. R. Ferro, and P. K. Hopke, “Estimating hourly concentrations of pm2.5 across a metropolitan area using low-cost particle monitors,” *Sensors*, vol. 17, no. 8, 2017.
- [65] “Flask.” <https://flask.palletsprojects.com>.
- [66] “Psycopg2.” <https://pypi.org/project/psycopg2>.
- [67] “Bootstrap.” <https://getbootstrap.com>.
- [68] “Angular powered Bootstrap.” <https://ng-bootstrap.github.io>.
- [69] “Keycloak - Open Source Identity and Access Management For Modern Applications and Services.” <https://github.com/keycloak/keycloak>.
- [70] D. Crizt, *Keycloak: una soluzione open source per la gestione delle identità e degli accessi*. 2020.
- [71] “Easy Keycloak setup for Angular applications.” <https://github.com/mauriciovigolo/keycloak-angular>.
- [72] “Keycloak-js.” <https://www.npmjs.com/package/keycloak-js>.
- [73] “What is CI/CD?.” <https://www.redhat.com/it/topics/devops/what-is-ci-cd>.
- [74] “Jenkins.” <https://www.jenkins.io>.