UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Ottimizzazione dell'HP Flottante con Modelli di Machine Learning

Relatore:

Prof. Andrea Loreggia

Laureando:

Diego Mussolin

Data di laurea 23/09/2022 **ANNO ACCADEMICO 2021 - 2022**

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Informatica

Ottimizzazione dell'HP Flottante con Modelli di Machine Learning

Relatore:
Prof. Andrea Loreggia

Laureando:
Diego Mussolin
1222649

Supervisore Aziendale
MTA SpA
Ing. Piero Milana

Anno Accademico 2021/2022

Abstract

In questo lavoro di tesi, svolto in collaborazione con MTA SpA, azienda dove sono impiegato, si sono analizzati alcuni metodi innovativi nel campo del *Machine Learning*, per progettare un simulatore alternativo a quello attualmente disponibile in azienda, scritto in linguaggio VBA su Microsoft Excel, che simula le prestazioni degli impianti di refrigerazione.
In particolare il focus di questa tesi è quello di addestrare algoritmi di machine learning, ad esempio il modello XGBoost, per cercare di simulare con più accuratezza, se possibile, una delle soluzioni più note per il risparmio energetico nel campo del freddo, **l'Alta Pressione Flottante** (Floating High Pressure).
Per poter confrontare la bontà del simulatore VBA Vs algoritmi ML, è stato costruito un refrigeratore d'acqua, con condensazione ad aria attraverso ventilatori a regolazione elettronica, che successivamente è stato messo in funzione e testato, con l'ausilio delle cabine di collaudo, per poter ottenere dati prestazionali reali, nelle condizioni più favorevoli (i mesi invernali) al funzionamento dell'impianto in *HP Flottante*.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Nell'epoca in cui viviamo, grazie ai dispositivi a nostra disposizione, ognuno di noi è produttore di una grande mole di dati, all'interno dei quali sono racchiuse importanti informazioni.

Il processo di analisi ed estrazione della conoscenza sta diventando sempre più attraente agli occhi delle aziende, perché queste informazioni possono rivelarsi utili e generare vantaggio competitivo.

In questa tesi verranno analizzate alcune metodologie per poter creare un modello intelligente, che possa rappresentare una delle soluzioni più usate per il risparmio energetico nel campo della *refrigerazione*, nota come **HP Flottante**.

L'alta pressione flottante o *condensazione flottante*, è una soluzione integrata per il risparmio energetico, che grazie ad una serie di algoritmi matematici, presenti nella memoria del controllo dei refrigeratori, permette di misurare la *pressione di condensazione ottima*, in funzione delle condizioni ambientali di lavoro, aumentando quindi il rendimento complessivo dell'impianto.

In determinate condizioni ambientali, temperatura e pressione di condensazione, "fluttuano" con la temperatura esterna, determinando un risparmio energetico in termini di energia elettrica assorbita, che cresce al diminuire della temperatura esterna.

Nello sviluppo di questa tesi verranno analizzati e implementati, alcuni tipi di algoritmi di machine learning, che fanno parte di una sottocategoria dell'*Apprendimento con Supervisione*: l'*Analisi a Regressione*.

Lo scopo finale, se questi modelli si "comportano bene", è quello di fornire strumenti che siano di supporto al funzionamento del refrigeratore finale, abi-

litando la modalità HP Flottante nel controllo della macchina, durante il suo funzionamento.

Il progetto è stato svolto, durante il tirocinio, in *MTA S.p.a.*, azienda nella quale sono attualmente impiegato a tempo pieno, ricoprendo il ruolo di *Data Management Support.*

MTA è una realtà imprenditoriale italiana che da 40 anni produce apparecchiature per il trattamento dei gas compressi, la refrigerazione industriale e il condizionamento dell'aria.

Negli anni ha saputo sviluppare una presenza commerciale a livello mondiale, guadagnandosi un'ottima reputazione per la qualità dei suoi prodotti, per la flessibilità e l'attenzione al cliente.

Oggi MTA è una Società fortemente radicata nel territorio italiano, in grado di offrire ai clienti soluzioni energetiche ottimali per aria e acqua. Un importante percorso di crescita accompagnato da sfide impegnative e coronato dalla soddisfazione dei risultati raggiunti nel mercato globale, pur mantenendo intatte le proprie origini.

Le potenzialità odierne del Gruppo MTA sono comprensibili focalizzando i suoi dati: una realtà che con oltre 500 dipendenti e 50 gamme di prodotti, ha realizzato nel 2021 un fatturato consolidato di quasi 100 milioni d'euro, con una produzione suddivisa nei tre siti produttivi di Conselve, Tribano e Bagnoli di Sopra, per un totale di 70.000 m^2 e con una capacità produttiva pari a 13.500 impianti all'anno. Per MTA l'export rappresenta l'80% del suo fatturato, di cui il 75% nell'Eurozona. I principali paesi sono: Germania, Francia, Russia, UK ed USA.

Lo svolgimento del progetto di tesi è stato svolto seguendo questi passaggi:

1. Partendo dal *software di simulazione MTA*, sviluppato in linguaggio VBA, si è generato un dataset di valori numerici, utile agli algoritmi di machine learning nella *fase di training.* Il dataset è composto da colonne che descrivono le proprietà fisiche di un refrigeratore, e da righe che rappresentano una simulazione termodinamica puntuale. Fra queste colonne, quella di maggior interesse è quella che fornisce l'informazione, del *numero di rotazioni al minuto* degli elettroventilatori installati nel refrigeratore, quando la macchina è in funzione. Questa colonna è quella dei valori che rappresentano la variabile di *target* per l'analisi a regressione.

2. E' stato costruito un refrigeratore modello AST 2 140. Il refrigeratore è stato collaudato nelle condizioni ambientali idonee alla simulazione in HP flottante. Da questi test ambientali è stato ottenuto un piccolo dataset della stessa forma del dataset estratto in VBA, contenente **dati reali**. Quest'ultimo viene tenuto da parte per essere usato nella *fase di test* per la valutazione delle prestazioni degli algoritmi di machine learning.

3. Sono stati proposti e valutati alcuni algoritmi di machine learning basati sul *bagging* e sul *boosting*.

4. Per migliorare le prestazioni degli algoritmi è stata fatta una fase iniziale di analisi, pulizia, ridimensionamento e standardizzazione dei dati.

5. I modelli sono stati addestrati sul dataset ottenuto dal simulatore VBA, e successivamente testati sul dataset ottenuti dal collaudo del refrigeratore AST 2 140.

6. Sono state misurate le prestazioni del simulatore VBA, effettuando simulazione software puntuali, impostando le stesse condizioni della simulazione reale, e confrontando questi risultati con quelli contenuti nel dataset dei dati reali.

7. Alla fine i risultati ottenuti dopo i test sui dati reali (punto 5), sono stati riportati, confrontati e analizzati, con quelli ottenuti dal modello VBA dopo i test sui dati reali (punto 6).

Di seguito presenterò la struttura della tesi, facendo una panoramica generale dei capitoli di cui si compone.

Il primo capitolo, l'introduzione all'elaborato di tesi.

Nel secondo capitolo verrà presentato il *background* necessario all'elaborato.

Nel terzo capitolo si vedranno tutte le *esperimentazioni* fatte per produrre i risultati dettagliati nella tesi.

Nel quarto e ultimo capitolo le *conclusioni finali*, e una visione sui possibili *scenari di sviluppo* futuri.

Capitolo 2

Background

In questo capitolo presento tutto il *Background* teorico, che descrive le metodologie, le tecniche e gli strumenti usati in questo elaborato di tesi.

Il focus sarà sui seguenti punti:

- Cos'è la tecnica dell'*HP Flottante*
- Come funziona un *Circuito Frigorifero*
- Il *Machine Learning* nelle sue varie applicazioni
- Gli *Strumenti Software* usati per analizzare i dati e produrre i risultati

2.1 HP Flottante

La refrigerazione e l'aria condizionata rappresentano il 15% del consumo globale di elettricità, che corrisponde al 4,5% delle emissioni globali di gas serra (GHG - Greenhouse gas emissions).

Che sia nell'industria alimentare, nei magazzini o nei grandi e medi negozi (supermercati), il freddo rappresenta una parte considerevole del consumo energetico (dal 30 all'80%). È uno fra i primi fattori su cui gli utenti vorrebbero risparmiare, ma è anche poco conosciuto e con più criticità. Nonostante le soluzioni esistenti da anni per la costruzione degli impianti per la refrigerazione e l'aria condizionata, la scelta iniziale privilegia il funzionamento e il prezzo a discapito dei consumi.

HP Flottante è una soluzione che invece predilige il contenimento dei consumi. Non tutti i sistemi di refrigerazione ne sono dotati. Alcune installazioni

con HP flottante fanno risparmiare poco, perché questa soluzione deve essere implementata correttamente per massimizzare il risparmio energetico senza causare problemi tecnici.

La soluzione dell'HP flottante e altre tecniche per il risparmio energetico, trovano molto interesse nelle strategie, messe in atto dalle autorità pubbliche francesi per la transizione energetica, sulla progettazione di strumenti che siano in grado di incoraggiare pratiche di consumo energetico più sostenibili, consentendo al contempo l'attuazione della direttiva europea 2012/27/UE denominata "efficienza energetica".

In questo contesto, la legge POPE (Programme fixant les Orientations de la Politique Energétique) del 13 luglio 2005 ha dato vita ai certificati di risparmio energetico (Certificats d'Économie d'Énergie - CEE), un meccanismo innovativo e ambizioso in materia di fiscalità ecologica.

Questo sistema, ha conosciuto la sua principale espansione nell'ambito dei principali produttori di energia francesi e delle società di certificazione.

Per quanto riguarda il contesto del risparmio energetico con la soluzione dell'HP Flottante, la legge POPE permette alle aziende che certificano i loro prodotti, rispettando *l'Operazione di standardizzazione N° IND-UT-16* - "Regolarizzazione di un gruppo di produzione del freddo che permetta di avere una elevata pressione di galleggiamento a seconda della temperatura dell'aria all'esterno", di godere di questi buoni di risparmio energetico, attraverso il finanziamento dei lavori.

2.1.1 Come funziona HP Flottante

HP flottante è una soluzione che permette, attraverso la modulazione continua dei ventilatori di un refrigeratore condensato ad aria, di ottimizzare il funzionamento dell'unità in qualsiasi punto di lavoro, garantendo un incremento dell'efficienza ai carichi parziali.

Come si vede dalla figura 2.1, HP flottante è dato dall'equilibrio fra la potenza da evacuare dell'intero sistema (curva in rosso) che si ottiene dalla somma data dalla potenza frigorifera istantanea sommata alla potenza dei compressori (Qevap + L), e la potenza che può essere evacuata dal condensatore (formato dalla coppia batteria alettata - ventilatore), a seconda delle sue condizioni di funzionamento. Maggiore è la differenza di temperatura fra l'aria esterna (temperatura ambiente) e la temperatura del refrigerante (temperatura di condensazione), maggiore è la capacità di evacuare.

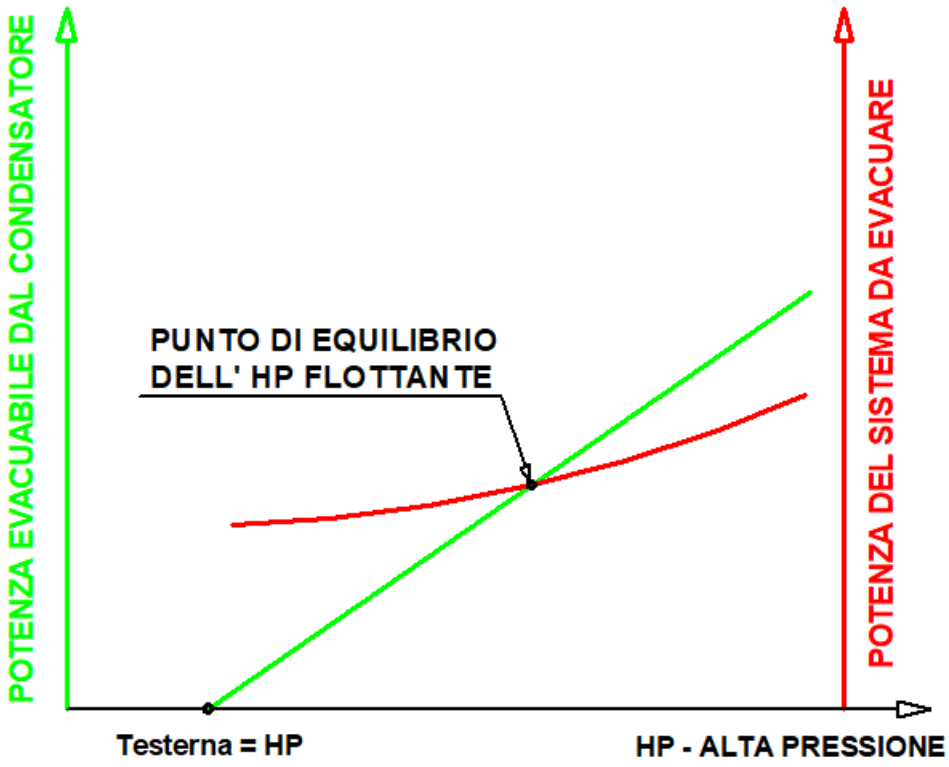


Figura 2.1: Alta pressione in funzione delle potenze evacuate

Sempre dalla figura 2.1, si può vedere che se la *Testerna* (temperatura esterna), raggiunge la temperatura dell'HP (temperatura di condensazione), allora *MTD* vale zero, e la potenza di scarico è nulla.

Dalla figura 2.2 si può vedere che per regolare il punto di equilibrio, la capacità di evacuare del condensatore viene regolata controllando i ventilatori, aumentando o diminuendo il flusso della ventilazione. La variazione della potenza da evacuare della sola batteria alettata è poca o poco flessibile, dipende infatti dalla sua forma (dimensioni, circuitazione, numero ranghi, passo e tipo alette). L'unico modo quindi per intervenire sull'HP flottante è quello di *controllare la ventilazione dei condensatori*.

In condizioni di bassa temperatura ambiente, HP flottante, permette di far diminuire la temperatura, e di conseguenza la pressione di condensazione, dando possibilità al compressore di diminuire il consumo di energia, aumentandone il rendimento. HP flottante consiste in sintesi nel regolare la pressione di condensazione ad un valore che permetta di ottenere il minor consumo delle coppie compressore/condensatore.

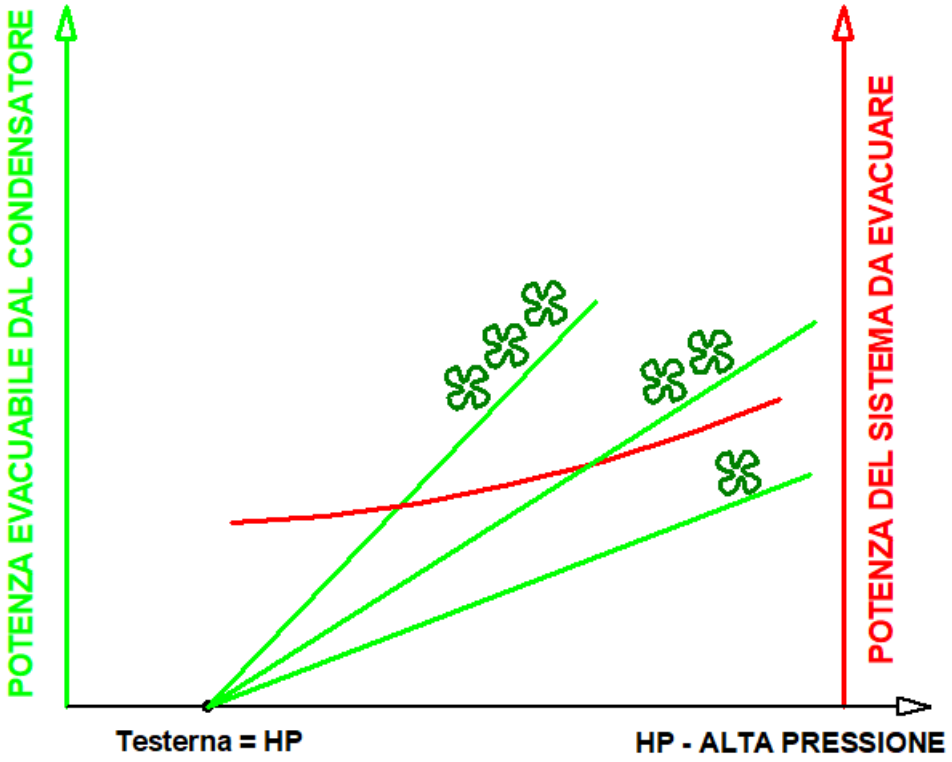


Figura 2.2: Influenza del flusso del ventilatore

In pratica si hanno i seguenti effetti diretti:

- aumento della potenza frigorifera dell'impianto
- riduzione della potenza elettrica assorbita
- aumento dell'efficienza energetica (EER)

Questi guadagni prestazionali sono significativi, in particolare durante la mezza stagione e i periodi invernali.

L'ottenimento di temperature di condensazione da 20°C a 25°C per gran parte dell'anno, rende possibile generare un risparmio energetico dal 15% al 25%.

HP flottante è una soluzione integrata nel controllo del refrigeratore, sviluppata attraverso una serie di algoritmi matematici che permettono di calcolare la pressione di condensazione ottimale in funzione delle condizioni ambientali.

Nelle condizioni ambientali sopra descritte, la temperatura/pressione di condensazione "fluttua" con la temperatura esterna, determinando un risparmio che cresce al diminuire di quest'ultima. Lo scopo di questa tesi è dunque quello di

modellare algoritmi, basati su modelli di machine learning, che siano in grado di fornire una regolazione più accurata dell'alta pressione di condensazione, rispetto a quella data dagli algoritmi matematici. Questi ultimi generalmente sono frutto di attente analisi sperimentali fatte dai produttori di refrigeratori, MTA in questo elaborato. In generale funzionano bene nella maggior parte dei casi, ma rispetto a modelli che si basano sui dati, sono meno propensi ad adattarsi a particolari condizioni di lavoro. Questo alla fine si traduce nel fatto che nella maggior parte delle situazione, con i modelli classici si hanno delle regolazioni di HP flottante che danno un buon risparmio energetico, ma probabilmente con algoritmi più performanti, in termini di "adattabilità", si potrebbero ottenere risultati e prestazioni più significative.

2.1.2 Il modello MTA

MTA, nei refrigeratori dove sono installati i ventilatori EC a regolazione elettronica, e viene richiesto il funzionamento in HP flottante, carica nella memoria del controllo, un programma che simula una funzione matematica. Lo scopo della funzione è quello di regolare i ventilatori EC al fine di far condensare l'impianto alla temperatura ottimale in termini di risparmio energetico. Quando abilitata, questa funzione lavora mantenendo un *MTD* (maximum temperature difference) **costante**. MTD = *Tc* - *Ta*, dove *Tc* è la temperatura di condensazione e *Ta* è la temperatura ambiente. Al variare di *Ta*, *Tc* varia in modo lineare. La temperatura di *condensazione ottimale* è quindi data dalla temperatura ambiente più l'*MTD*. Questo valore viene limitato inferiormente e superiormente dai valori di *Tc* minima e *Tc* massima, dati dal campo di lavoro (envelope) del compressore installato nel refrigeratore.

2.2 Il Ciclo Frigo

La tecnologia del freddo può considerarsi nata nel 1880 quando fu utilizzata per congelare i prodotti ittici. La forma base del ciclo frigo sfrutta una superficie fredda che a contatto con l'alimento estrae calore per conduzione.

Una valutazione diffusa recentemente indicava in 148 milioni di pezzi la produzione mondiale di frigoriferi; essa non sembra però accettabile se si tiene conto del fatto che la sola produzione cinese, rilevata dall'ente governativo di statistica NBSC, ammontava a 103 milioni di unità nel 2015. Più attendibile sembra un'al-

tra versione, che è invece di fonte cinese, che indica per lo stesso anno un totale mondiale di 205 milioni di pezzi.

Di essi i tre quarti sono stati prodotti nell'Asia Orientale: la Cina da sola rappresenta grosso modo la metà del totale, mentre un altro quarto è prodotto nella Corea del Sud, in Giappone, nella Tailandia o a Singapore. L'ultimo quarto si produce nelle Americhe, in particolare negli Stati Uniti, in Messico o nel Brasile, e in Europa. Qui ai primi posti sono l'Italia e la Germania.

I leader mondiali sono i gruppi Haier, Electrolux, LG, Samsung e Whirlpool; ognuno di essi detiene una quota di mercato significativa, compresa tra il 5% e il 7%, con produzioni ciascuno tra i 10 e i 15 milioni di apparecchi. Le percentuali sono ovviamente calcolate assumendo come totale mondiale la cifra di 205 milioni di unità.

Il mercato dei frigoriferi è ormai vicino alla saturazione nei Paesi sviluppati, pur essendovi ancora una lieve fase di crescita, destinata ad esaurirsi nei prossimi anni. Enormi possibilità vi sono invece nelle nazioni in via di sviluppo. Dall'Africa, Asia e Sudamerica verranno le future crescite.

2.2.1 Il circuito frigorifero

Per *Refrigerazione* si intende "il trasferimento di calore da un ambiente che deve essere mantenuto a bassa temperatura ad un altro in cui può essere agevolmente smaltito".

Questo processo viene realizzato da un *Fluido Refrigerante* in un circuito frigorifero.

Il fluido refrigerante deve avere la capacità di asportare calore dall'ambiente da refrigerare evaporando a bassi valori di pressione e temperatura e cederlo all'ambiente esterno condensando a più elevata pressione e temperatura.

Il calore che il fluido refrigerante asporta coincide con il suo *calore latente di vaporizzazione*, che è il calore necessario all'unità di massa [J/kg] di una sostanza per effettuare il *Passaggio di Stato dell'Evaporazione*, ad un certo valore di pressione e temperatura. Il calore latente dipende dalla pressione e della temperatura a cui avviene il cambiamento di fase. Nelle trasformazioni liquido $\rightarrow$ vapore e vapore $\rightarrow$ liquido vengono assorbite e rilasciate le stesse quantità di calore. Perciò il calore latente di vaporizzazione per una particolare sostanza è esattamente uguale al *calore di condensazione*.

Le grandezze fondamentali considerate in un circuito frigorifero sono le seguenti:

- *Temperatura*: (misurata sulla scala kelvin [K]) lo scopo della tecnica del freddo è scendere al di sotto della T ambiente.

- *Pressione*: (si misura in Pascal [Pa]) pressione e temperatura sono strettamente legate l'una a l'altra.

- *Entalpia*:(è un'energia si misura in joule [J]) è la grandezza usata in termodinamica (e quindi in tecnica del freddo) per definire lo stato energetico di una sostanza che si trova a determinate condizioni di pressione e temperatura. Esprime la quantità di energia interna che un sistema termodinamico può scambiare con l'ambiente.

Nella *Tecnica del Freddo* i cicli frigoriferi sono rappresentati in diagrammi $p-h$ (pressione - entalpia), dove possono essere evidenziati alcuni aspetti caratteristici del fluido frigorigeno e del ciclo stesso.

Punto Critico: punto di massimo della curva. Al di sopra della temperatura critica il fluido si presenta solo allo stato gassoso, al di sotto possono avvenire i passaggi di stato (evaporazione e condensazione).

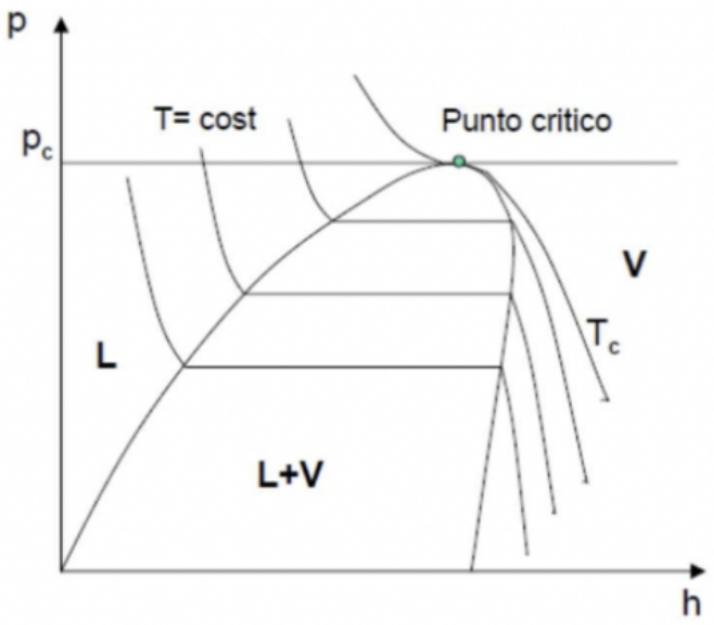


Figura 2.3: Diagramma $p-h$ (pressione-entalpia)

Il diagramma $p-h$ consente di determinare il *Calore Latente di Vaporizzazione* come differenza tra l'entalpia specifica del vapore saturo e quella del liquido saturo.

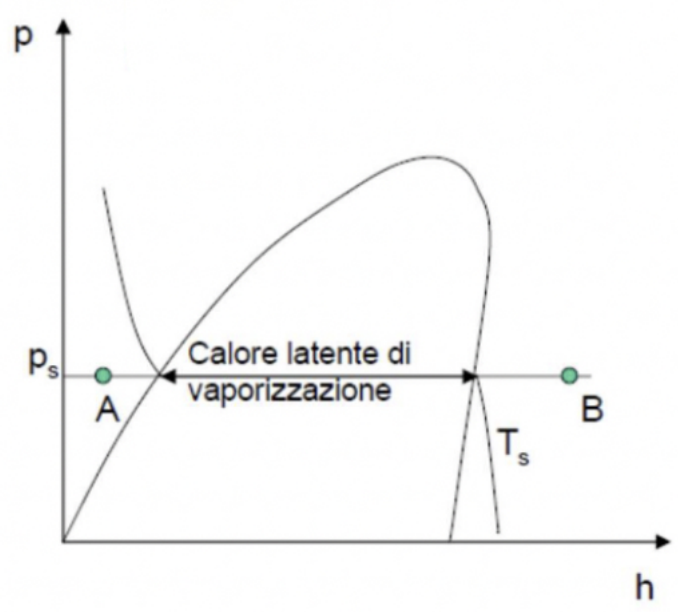


Figura 2.4: Calore Latente di Vaporizzazione

$$dq = dl + dl'$$

- **(Punto A) - Liquido sottoraffreddato**: liquido ad una temperatura più bassa di quella di saturazione alla pressione considerata.
- **(Punto B) - Vapore surriscaldato**: vapore ad una temperatura più elevata di quella di saturazione alla pressione considerata.

L'entalpia di un vapore saturo è data dalla somma dell'entalpia del liquido saturo più Il calore latente di vaporizzazione.

2.2.2 Come si produce il freddo

E' esperienza comune che il calore passi spontaneamente da corpi a temperatura elevata a corpi a bassa temperatura.

Per fare avvenire il processo contrario è necessario mettere a punto degli opportuni dispositivi che vengono detti **macchine frigorifere**, i quali hanno le seguenti caratteristiche:

1. Funzionano secondo *trasformazioni termodinamiche cicliche*. Alla fine di ogni ciclo la macchina si ripresenta nelle condizioni iniziali pronta a trasferire ancora calore.

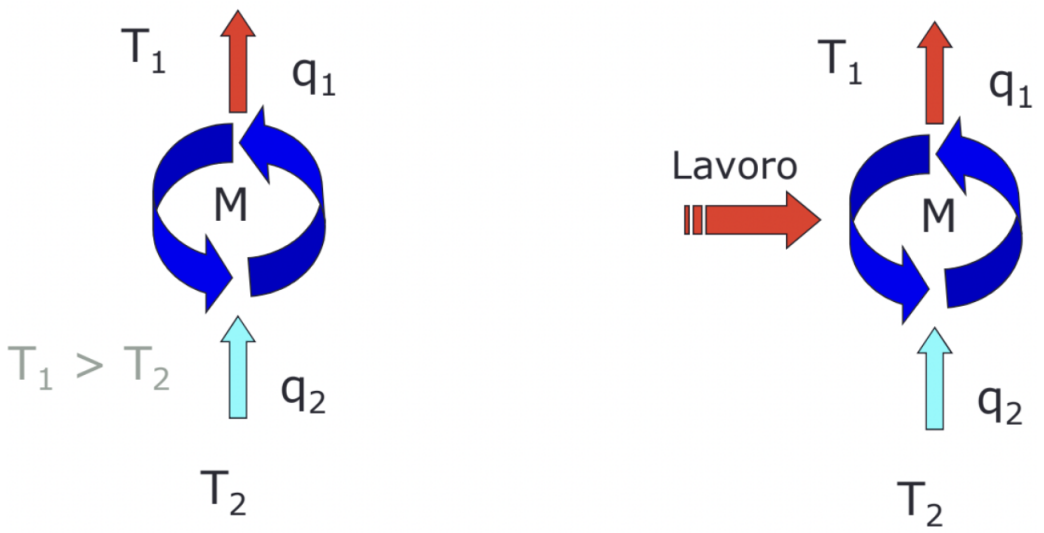


Figura 2.5: Scambio termico fra sorgenti di calore

2. *Assorbono calore* da una sorgente a bassa temperatura.
3. *Cedono il calore* assorbito a una sorgente a temperatura elevata.
4. Necessitano di una ulteriore *quota di energia* immessa dall'esterno sotto forma di lavoro (o calore).

Dal secondo principio della termodinamica si evince che **non esiste** una macchina operante in modo ciclico il cui unico effetto sia lo scambio termico con sorgente a bassa temperatura verso una sorgente a temperatura più elevata.

Lo scambio termico avviene *spontaneamente* solo da sistemi a temperatura più alta verso sistemi a temperatura più bassa.

2.2.2.1 Tipi di macchine frigorifere

Le principali tecnologie per lo sviluppo di macchine frigorifere sono le seguenti:

- A compressione di vapori saturi (le più usate)
- Ad aria (usate nella climatizzazione di velivoli a reazione)
- Ad assorbimento (utili se si dispone di cascami termici)
- Ad effetto termoelettrico Peltier (usate per piccolissime potenze: qualche watt)

2.2.2.2 Il ciclo frigorifero a compressione di vapore

Il ciclo inverso a compressione di vapore costituisce uno dei cicli utilizzati nelle macchine per il trasferimento di calore da sistemi a bassa temperatura a sistemi ad elevata temperatura.

Il ciclo si basa sul fatto che il *passaggio di stato liquido-vapore* può avvenire in un verso o nell'altro al variare della pressione e delle condizioni ambientali con assorbimento o cessione di calore.

Si utilizza un fluido "frigorigeno" in grado, nel passaggio di stato, di assorbire una elevata quantità di calore per unità di massa.

Le quattro fasi fondamentali (l'ordine con cui sono indicate è quello più naturale) in un ciclo inverso a compressione di vapore sono:

1. **Compressione**: a carico del compressore
2. **Condensazione**: a carico del condensatore
3. **Espansione**: a carico della valvola di laminazione
4. **Evaporazione**: a carico dell'evaporatore

La fase di compressione

Il compressore è azionato da energia elettrica fornita dall'esterno, il cui equivalente termico si somma al calore assorbito dal frigorigeno nell'evaporatore.

L'equivalente termico del lavoro di compressione (vedi figura 2.5 lato destro, il compressore rappresenta il "Lavoro" freccia rossa entrante) si determina come differenza del contenuto entalpico del frigorigeno tra uscita e ingresso del compressore.

La fase di condensazione

Il calore totale da smaltire al condensatore è dato dalla somma del calore dovuto al desurriscaldamento, alla condensazione e al sottoraffreddamento.

La fase di espansione

Durante la fase di espansione una frazione di refrigerante evapora e costituisce il cosiddetto «flash-gas» che non compie lavoro utile ai fini dell'effetto frigorifero.

Per rendere possibile il raffreddamento del refrigerante in fase di espansione, il refrigerante cede del calore fra le sue molecole, così facendo evapora.

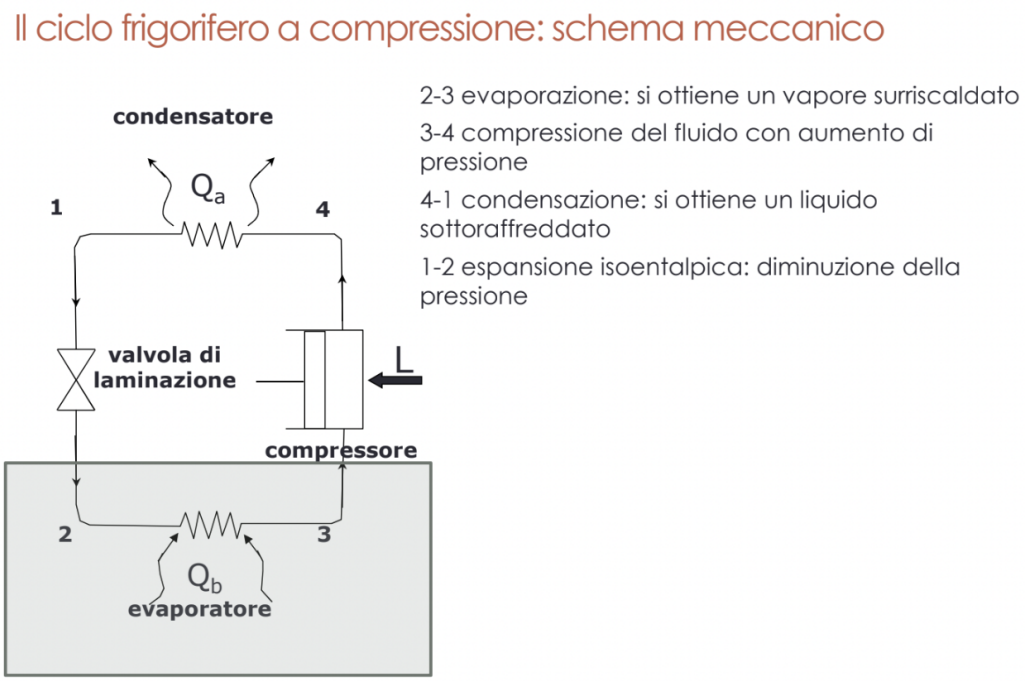
La fase evaporazione

La cessione di calore dal fluido che occorre raffreddare al frigorigeno può avvenire se la temperatura del frigorigeno è inferiore a quella del fluido stesso.

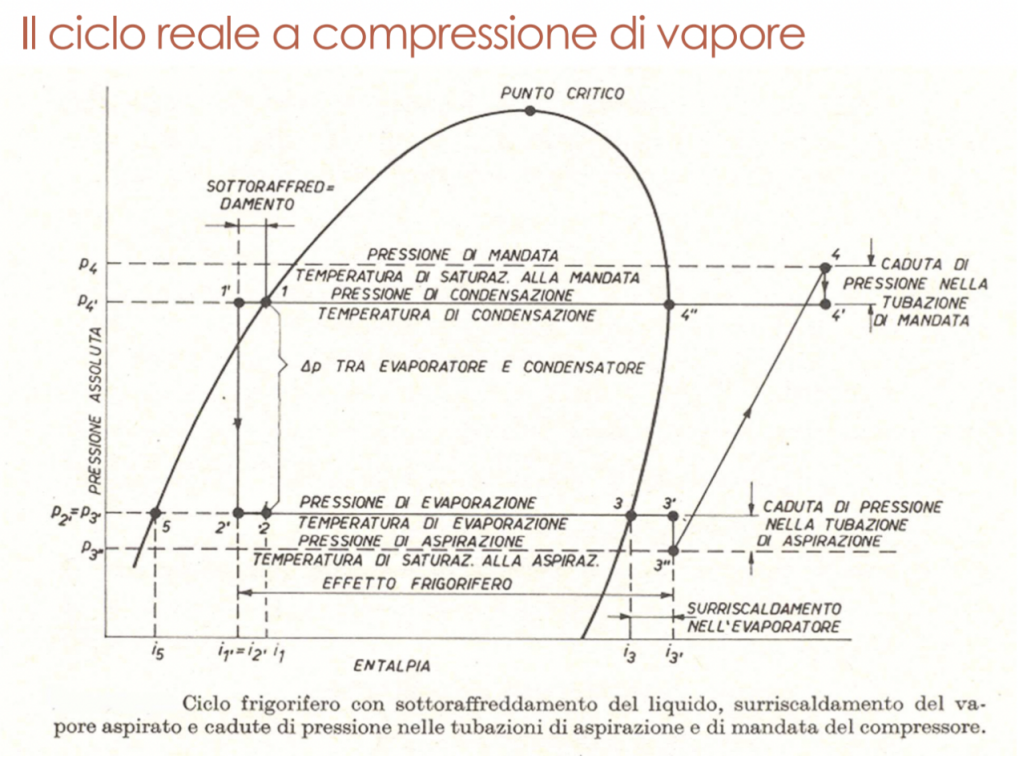
L'effetto frigorifero è dato dalla differenza di entalpia tra il punto di uscita del frigorigeno dall'evaporatore e il punto di entrata.

Di seguito due immagini:

la prima rappresentativa del "ciclo frigorifero a compressione di vapore".



la seconda presenta il diagramma del "ciclo frigorifero reale a compressione di vapore".



2.2.2.3 I coefficienti prestazionali

La bontà e l'efficienza di un buon impianto frigorifero viene misurata attraverso dei coefficienti di prestazione, in particolare:

- **C.O.P.** (Coefficient of Performance) è il rapporto tra l'effetto frigorifero *Qevap* (potenza scambiata dell'evaporatore), e il lavoro meccanico L, fatto dal compressore (potenza meccanica scambiata). E' un parametro che indica *l'efficienza elettrica* di una macchina termica, mentre funziona in **riscaldamento** (pompa di calore).

$$C.O.P. = \frac{Q_{evaporatore}}{P_{compressore}} \tag{2.1}$$

- **E.E.R.** (Energy Efficiency Ratio) è analogo al COP, ma in questo caso il rapporto tra le diverse grandezze è espresso in unità coerenti (es. W/ W). E' un parametro che indica *l'efficienza elettrica* di una macchina termica, mentre funziona in **raffreddamento** (chiller).

$$E.E.R. = \frac{Q_{evaporatore}}{P_{compressore}} = C.O.P. \tag{2.2}$$

- **S.E.E.R.** (Seasonal Energy Efficiency Ratio) è calcolato come il rapporto tra la domanda annuale di refrigerazione e il consumo annuale di elettricità della macchina. Tiene conto delle variazioni del carico termico e della temperatura ambiente durante l'anno, nonché della capacità della macchina di adattarsi alla variabilità di queste condizioni.

- **S.E.P.R.** (Seasonal Energy Performance Ratio) è calcolato come il rapporto tra la domanda annuale di refrigerazione e il consumo annuale di elettricità della macchina. Tiene conto delle variazioni del carico termico e della temperatura ambiente durante l'anno, nonché della capacità della macchina di adattarsi alla variabilità di queste condizioni.

SEER è un indici energetico valevole nell'ambito civile/residenziale, il SEPR è unicamente legato all'ambito industriale. Questi indici sono la naturale evoluzione degli indici COP ed EER, con una differenza fondamentale. Se infatti i precedenti indici prevedevano valori puntuali, calcolando la prestazione della macchina ad una determinata condizione, questi nuovi indici cercano invece di determinare con un solo valore il livello di efficienza di una macchina considerando l'arco temporale di un anno.

2.2.3 La simulazione in VBA

A partire dai primi anni 2000 e fino ad oggi, MTA ha sviluppato, implementato e perfezionato un software all'interno del pacchetto *Excel* di Microsoft, scritto in linguaggio *VBA - Visual Basic for Application.*

VBA è un linguaggio di scripting è può essere usato per implementare molti aspetti di un'applicazione, ad esempio la sua *GUI.*

Il simulatore MTA scritto in VBA permette di scrivere ed inserire dati di input e calcola in output dati termodinamici e prestazionali del modello che si sta simulando.

L'oggetto della simulazione è il *ciclo frigorifero a compressione di vapore*, che si sviluppa attraverso le sue quattro fasi fondamentali:

- Compressione
- Condensazione
- Espansione

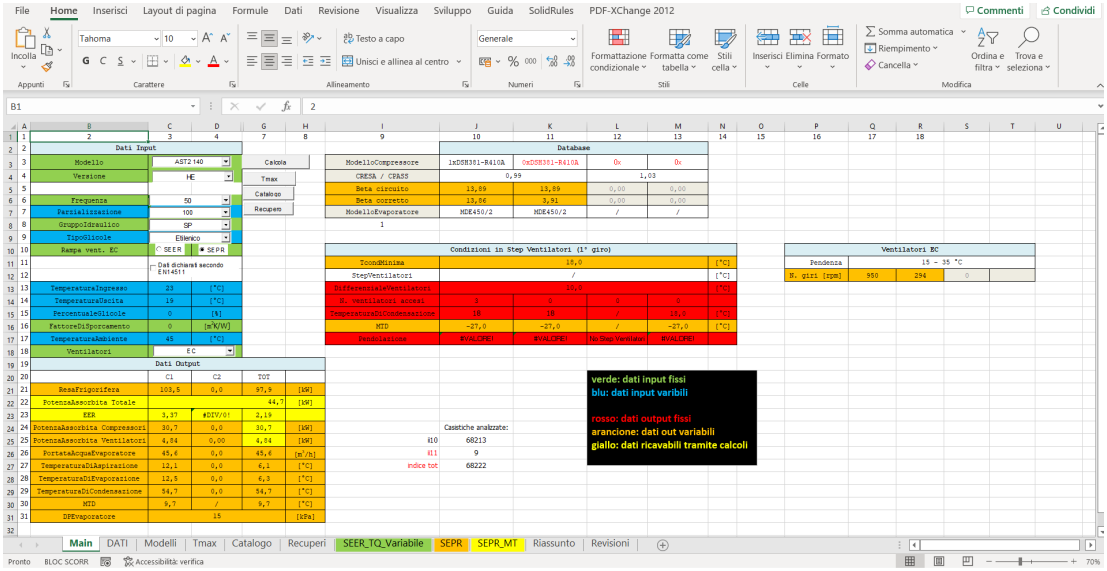


Figura 2.6: GUI simulatore VBA

- Evaporazione

Ognuna della quattro fasi è stata astratta in pseudocodice e poi scritta in VBA, costruendo scripting più o meno complessi che simulano la termodinamica del componente interessato alla fase in esame.

In base alle scelte tecniche fatte dal capo progetto, durante le fasi preliminare di studio di una nuova gamma con nuovi modelli, vengono dimensionati quelli che saranno i componenti meccanici necessari ad un refrigeratore quali il compressore, la batteria, il ventilatore e tutti gli altri.

Ad esempio, per la fase di *Compressione*, dove il *compressore* è il soggetto principale, ne viene selezionato tipo, fornitore, modello. Il capo progetto usa i simulatori messi a disposizione dai produttori dei compressori, per simulare il comportamento del compressore montato nel refrigeratore MTA, ottenendo da questi simulatori una serie di coefficienti prestazionali del compressore, coefficienti che saranno i dati di input del simulatore VBA per simulare appunto la fase di *Compressione*.

Con la stessa logica descritta sopra vengono così selezionati i condensatori, il tipo e numero di ventilatori da installare, l'evaporatore, la termostatica e tutti i componenti che con i loro indici, forniranno i dati di input al simulatore VBA.

Nel modello AST 2 140, costruito e testato in questo tirocinio, sono stati installati compressori di tipo Scrool, batterie condensanti a pacco alettato, ventilatori EC a regolazione elettronica, termostatiche elettroniche ed evaporatore a

fascio tubiero.

Dalla simulazione si è ottenuto il dataset dei dati oggetto di questa tesi.

2.2.4 Il dataset

Il dataset è stato generato impostando nel simulatore i seguenti parametri:

- modello: **AST 2 140**
- versione: **HE**
- tipo ventilatore: **EC**
- curva di lavoro ventilatore: **SEPR**
- min/max ambiente: **-10°C - +45°C**
- min/max IN acqua: **9°C - 23°C**
- delta temperatura IN/OUT acqua: **5°C**
- gradini di parzializzazione circuti: **25%, 50%, 100%**

Siccome ogni simulazione fornisce una prestazione puntuale, è stato modificato il software con un loop nei valori in ingresso, per fare in modo che simulasse più prestazioni. Ogni simulazione è stata fatta incrociando i dati di input in modo da fornire in output, solo dati ammissibili, ossia simulazioni che possono essere replicate in fase di collaudo.

Prima di ottenere il dataset finale si è operata una ulteriore modifica al software. Come detto nel paragrafo 2.1.2, l'algoritmo caricato nel controllo del refrigeratore, che abilita l'opzione *HP Flottante*, lavora mantenendo un *MTD* costante, variando in modo lineare la temperatura di condensazione al variare della temperatura ambiente.

Il simulatore VBA invece, ha come impostazione la curva di lavoro del ventilatore *SEPR - Seasonal Energy Performance Ratio*, che lavora ad *MTD* variabile.

HP Flottante fa lavorare i ventilatori con curve di lavoro diverse da quelle date dalle curve SEPR.

Per fare in modo che il software di simulazione scritto in VBA, si *comportasse* come se stesse simulando l'opzione *HP Flottante*, è stato fatto un altro loop, impostando la colonna *MTD* ad un valore fisso, non calcolato, e ricalcolando in tutte le simulazioni ottenute dal primo loop, nuovamente i dati in output.

Il dataset finale comprende **68175** righe (ogni record è una simulazione) per **51** colonne (caratteristiche termodinamiche e prestazionali).

Il dataset non contiene valori nulli o inconsistenti, è stato ottenuto dando in input al simulatore solo valori ammissibili, e in output si hanno soluzioni di simulazioni replicabili in collaudo.

2.3 Il Machine Learning

Negli ultimi anni Il *Machine Learning* o "Apprendimento Automatico" è diventata una delle tecnologie più importanti del nostro tempo. È adottato da molte aziende, fra le quali, Google, Apple, Amazon, Microsoft, Facebook, che hanno fortemente investito nella ricerca e nelle applicazioni di machine learning. La prima applicazione (algoritmo) di machine learning, che ha rivoluzionato la vita di milioni di persone risale agli anni 90, è stata il *Filtro Antispam*.

Oggi Il machine learning è una tecnologia diventata quasi indispensabile nella nostra vita quotidiana, ed è usata su innumerevoli prodotti e funzionalità che si utilizzano regolarmente tutti i giorni, anche senza farne caso. Si pensi agli assistenti vocali sugli smartphone, solo per fare qualche esempio di uso comune.

Ecco una delle tante definizioni di machine learning: "*Il Machine Learning è una branca dell'Intelligenza Artificiale che si occupa dello sviluppo di algoritmi e tecniche finalizzate all'apprendimento automatico mediante la statistica computazionale e l'ottimizzazione matematica*".

Qui una lista di attività dove gli algoritmi di machine learning possono fare la differenza rispetto ad un approccio tradizionale:

- Problemi per i quali le soluzioni esistenti richiedono una grande messa a punto o lunghi elenchi di regole: un algoritmo di machine learning può spesso semplificare il codice e funzionare meglio dell'approccio tradizionale.

- Problemi complessi per i quali l'utilizzo di un approccio tradizionale non porta a una buona soluzione: le migliori tecniche di machine learning possono forse trovare una soluzione.

- Ambienti fluttuanti: un sistema di machine learning può adattarsi a nuovi dati.

- Ottenere informazioni dettagliate su problemi complessi e grandi quantità di dati.

2.3.1 Sistemi di machine learning

Esistono diversi sistemi di machine learning ed è utile classificarli in ampie categorie, in base ai diversi criteri:

- Sistemi addestrati o meno con la supervisione umana:
  - Supervisionati
  - Non Supervisionati
  - Semi Supervisionati
  - Apprendimento per Rinforzo

- Sistemi che possono apprendere in modo incrementale al volo, ad esempio online rispetto all'apprendimento in batch.

- Sistemi che funzionano semplicemente confrontando nuovi punti dati con punti dati noti, oppure rilevando modelli nei dati di addestramento e costruendo un modello predittivo.

Questi criteri non sono esclusivi, ma possono essere combinati fra di loro per ottenere modelli più elaborati ed efficienti.

2.3.1.1 Supervised / Unsupervised Learning

I sistemi di machine learning possono essere classificati in base alla quantità e al tipo di supervisione che ottengono durante la formazione. Esistono quattro categorie principali: *Apprendimento Supervisionato*, *Apprendimento Senza Supervisione*, *Apprendimento Semi Supervisionato* e *Apprendimento Per Rinforzo*.

Ognuno di essi può essere utilizzato per risolvere uno o più problemi pratici in cui possono essere applicati.

Apprendimento con Supervisione (Supervised Learning)

Lo scopo principale dell'apprendimento con supervisione è quello di istruire un modello a partire da dati di addestramento etichettati, che gli consentirà di eseguire predizioni su dati che non ha mai visto prima o futuri. La qualifica "con supervisione" fa riferimento ad un insieme di esempi di addestramento, i dati di input, in cui i segnali di output desiderati, le *etichette*, sono già noti. Nell'apprendimento con supervisione, i dati di addestramento etichettati vengono passati ad un algoritmo di machine learning, per adattare un *modello predittivo* in grado di effettuare predizioni su dati di input nuovi, senza etichette.

Una sottocategoria dell'apprendimento con supervisione è la *Classificazione*. Tornando all'esempio del filtro antispam accennato all'inizio, si può addestrare un modello impiegando un algoritmo di machine learning con supervisione, su un set di messaggi di posta elettronica già etichettati come spam o non-spam, per predire (classificare in questo caso) a quale delle due categorie appartiene un nuovo messaggio. Il filtro per il rilevamento dello spam è un esempio di classificazione binaria, dove l'algoritmo di machine learning apprende un insieme di regole con lo scopo di distinguere fra due classi possibili: messaggi di posta elettronica spam o non-spam.

Un'altra sottocategoria dell'apprendimento con supervisione è la *Regressione*. La regressione è utile per la predizione di valori continui, un compito chiamato anche *analisi a regressione*. Nell'analisi a regressione, si ha un certo numero di variabili predittrici descrittive (*feature*) e una variabile di risposta continua (*target*). Il modello di machine learning tenta di trovare una relazione fra tali variabili che consenta di predire un risultato.

Apprendimento Senza Supervisione (Unsupervised Learning)

Nell'*Apprendimento senza Supervisione*, i dati di addestramento *non sono etichettati* o hanno una struttura ignota. Con le tecniche di apprendimento senza supervisione si è in grado di esplorare la struttura dei dati senza la guida di una variabile risultante nota Θ.

Il *Clustering* è una tecnica di analisi esplorativa dei dati, che consente di organizzare una massa di informazioni in sottogruppi significativi (*cluster*), senza precedente conoscenza delle loro appartenenze a gruppi. Se l'analisi evidenza dei cluster, questi ultimi raggruppano oggetti che condividono un certo livello di similarità, ma che differiscono significativamente da oggetti di altri cluster, motivo per cui il cluster viene anche chiamato *classificazione senza supervisione*.

Un altro sottocampo dell'apprendimento senza supervisione è la *Riduzione della Dimensionalità* in cui l'obiettivo è semplificare i dati senza perdere troppe informazioni. Un modo per farlo è unire diverse funzionalità correlate in una sola. Questo permette di ridurre lo spazio di memorizzazione e migliorare le prestazioni computazionali per gli algoritmi di machine learning. La riduzione della dimensionalità senza supervisione è un approccio usato nella pre elaborazione delle caratteristiche, con lo scopo di eliminare il rumore dei dati, in grado di degradare le prestazioni predittive di determinati algoritmi, comprimendo i dati in sottospazi a dimensionalità inferiore, mantenendo la maggior parte delle informazioni rilevanti.

2.3.1.2 Reinforcement Learning

Nel Reinforcement Learning, il sistema di apprendimento è chiamato *Agente*. L'agente esegue azioni che modificano l'*ambiente*, provocando passaggi da uno *stato* all'altro. A seguito di queste azioni che modificano gli stati, l'agente può ottenere in cambio ricompense (o penalità sotto forma di ricompense negative). Deve quindi imparare da solo qual è la strategia migliore, chiamata *Policy*, per ottenere la massima ricompensa nel tempo. Una policy definisce quale azione l'agente deve scegliere quando si trova in una determinata situazione.

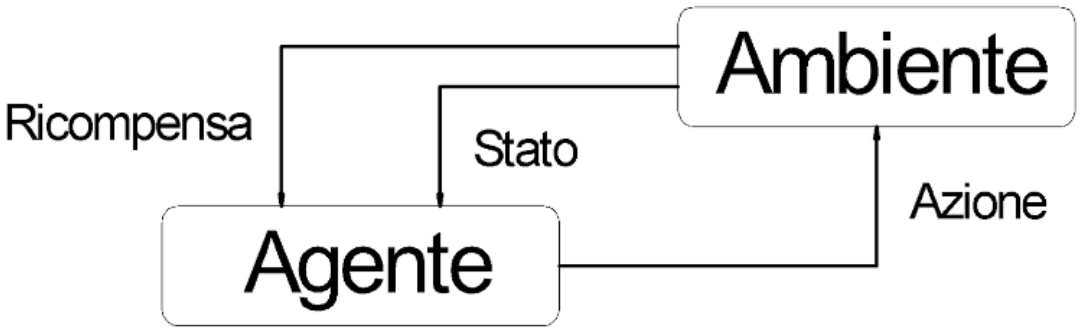


Figura 2.7: Agente Vs Ambiente

2.3.1.3 Batch and Online Learning

Un altro criterio utilizzato per classificare i sistemi di machine learning è se il sistema può apprendere in modo incrementale da un flusso di dati in arrivo.

Batch learning

Nell'*apprendimento batch*, il sistema non è in grado di apprendere in modo incrementale: deve essere addestrato utilizzando tutti i dati disponibili. Ciò richiede generalmente molto tempo e risorse di elaborazione, quindi in genere viene eseguito offline. Prima il sistema viene addestrato, quindi viene avviato in produzione e viene eseguito senza più apprendere; applica solo ciò che ha appreso. Questo si chiama *apprendimento offline*. Attualmente la maggior parte dei sistemi di machine learning opera in questo modo.

Per fare in modo che un sistema di apprendimento in batch sia a conoscenza di nuovi dati (come un nuovo tipo di spam), è necessario addestrare una nuova versione del sistema da zero sul set di dati completo (non solo i nuovi dati, ma anche i vecchi dati), quindi arrestare il vecchio sistema e sostituirlo con quello nuovo. Questa soluzione è semplice da implementare è generalmente funziona bene, ma l'addestramento può richiedere parecchio tempo e molte risorse di elaborazione. Generalmente viene pianificato, ad esempio settimanalmente.

Online learning

Nell'apprendimento online, si addestra il sistema in modo *incrementale*. Lo si alimenta con istanze di dati in sequenza, individualmente o in piccoli gruppi chiamati *mini-batch*. Ogni fase di apprendimento è veloce ed economica, quindi il sistema può apprendere nuovi dati al volo, non appena arrivano. L'apprendimento online è anche una buona opzione se si dispone di risorse limitate. Una volta che il sistema ha appreso dalle nuove istanze di dati, non ne ha più bisogno, possono essere scartate, e questo può far risparmiare enormi quantità di spazio in memoria.

Un parametro importante per i sistemi di apprendimento online è il *tasso di apprendimento*. Questo valore misura la velocità con cui il sistema dovrebbe adattarsi ai dati in cambiamento. Se si imposta un alto tasso di apprendimento il sistema si adatterà rapidamente ai nuovi dati in ingresso, ma tenderà anche a dimenticare rapidamente i vecchi dati. Viceversa il sistema avrà più inerzia, apprende più lentamente, ma sarà anche meno sensibile al rumore (valori anomali) dei nuovi dati in ingresso.

2.3.1.4 Apprendimento basato sulle istanze e apprendimento basato sui modelli

Un altro modo per classificare i sistemi di machine learning è il modo in cui *generalizzano*. La maggior parte delle attività di machine learning riguardano la *realizzazione di previsioni*. Dato un numero di esempi di addestramento, il sistema deve essere in grado di fare buone previsioni, in modo da "generalizzare" esempi che non ha mai visto prima. Avere una buona misura delle prestazioni sui dati di allenamento è buono, ma insufficiente; il vero obiettivo è ottenere buoni risultati su nuove istanze.

Esistono due approcci principali alla generalizzazione:

- Apprendimento basato su istanze
- Apprendimento Basato su modelli

Apprendimento basato sulle istanze (Instance-based learning)

Nell'apprendimento basato sull'istanza, il sistema apprende gli esempi a memoria, quindi generalizza a nuovi casi utilizzando una *misura di somiglianza*, per confrontarli con gli esempi appresi. Tornando sempre all'esempio del filtro antispam, il sistema non contrassegna come spam solo le mail identiche a quelle già note come spam, ma attraverso una misura di somiglianza fra mail, ad esempio il numero di parole che hanno in comune, il sistema andrà a quantificare determinate similitudini fra le mail. In questo modo il sistema non contrassegna come spam solo le mail identiche alle e-mail di spam note, ma anche quelle che hanno molte parole in comune.

Apprendimento basato sui modelli (Model-based learning)

Un altro modo per generalizzare da una serie di esempi, è quello di costruire un modello di questi esempi per poi utilizzarlo per fare previsioni. Questo tipo di apprendimento è chiamato *apprendimento basato su modello*.

Per quantificare la bontà del modello è necessario specificare una *misura* delle sue prestazioni. In generale si può definire una *funzione di utilità* (chiamata anche fitness function) che misura quanto il modello è "buono", o, si può definire una *funzione di costo* (chiamata anche loss function) che misura quanto il modello è "cattivo".

Ad esempio, nei modelli di regressione lineare, in genere si utilizza una funzione di costo, che misura la distanza tra le previsioni del modello lineare e gli esempi di addestramento. L'obiettivo è ridurre al minimo questa distanza.

In genere, il comportamento di un algoritmo di machine learning è regolato da un set di parametri Θ. L'apprendimento consiste nel determinare il valore ottimo Θ^* di questi parametri.

Molti algoritmi di apprendimento basati sui modelli richiedono di definire, prima dell'apprendimento vero e proprio, il valore degli *iperparametri*, che sono parametri specifici dell'algoritmo in uso. Gli iperparametri vanno impostati inizialmente ad un valore *ragionevole*, si esegue quindi la fase di apprendimento e al termine della procedura si scelgono gli iperparametri che hanno fornito le prestazioni migliori.

2.3.2 I principali problemi

Per insegnare ad un bambino che cos'è una mela, basta indicargli una mela e ripetere alcune volte, a voce alta, la parola mela. Il bambino sarà ora in grado di riconoscere le mele in tutti i tipi colori e forme.

L'apprendimento automatico non ha ancora raggiunto questi livelli. Per la maggior parte degli algoritmi di machine learning ci vogliono molti dati perché funzionino correttamente. Anche per problemi molto semplici in genere sono necessari migliaia di esempi e per problemi complessi come il riconoscimento di immagini o vocali potrebbero essere necessari milioni di esempi.

Per poter generalizzare bene, è fondamentale che i dati di addestramento siano rappresentativi dei nuovi casi a cui si desidera generalizzare. Questo è vero sia che si utilizzi l'apprendimento basato sull'istanza o l'apprendimento basato sul modello.

Se il campione di dati è troppo piccolo, si avrà rumore di campionamento (cioè dati non rappresentativi), ma anche campioni molto grandi possono non essere rappresentativi se il metodo di campionamento è difettoso. Questo è chiamato *Bias di Campionamento*.

Se i dati di campionamento sono pieni di errori, valori anomali o rumore, sarà difficile individuare un modello che funzioni bene. Per questo motivo si dedica molto tempo alla pulizia dei dati di allenamento.

Si possono avere problemi di *Overfitting*. Il modello si comporta bene sui dati di addestramento, ma poi non generalizza bene in fase di test. L'overfitting si verifica quando il modello è troppo complesso rispetto alla quantità e alla

rumorosità dei dati di addestramento. E' possibile adottare alcune strategie per risolvere i problemi di overfitting. Ad esempio:

- Semplificare il modello selezionando uno con meno parametri (ad esempio, un modello lineare piuttosto che un modello polinomiale di alto grado), riducendo il numero di attributi nei dati di addestramento o vincolando il modello
- Raccogliere più dati di allenamento
- Ridurre il rumore nei dati di addestramento (ad esempio, correggendo gli errori dei dati e rimuovendo i valori anomali)

Si possono avere problemi di *Underfitting*. L'underfitting è l'opposto dell'overfitting. Si verifica quando il modello è troppo semplice per apprendere la struttura sottostante dei dati.

2.3.3 Valutazione delle prestazioni

Riepilogando quanto visto fino ad ora, l'apprendimento automatico consiste nel migliorare le macchine in alcune attività imparando dai dati invece di dover codificare esplicitamente le regole.

Esistono diversi tipi di machine learning: supervisionato o meno, batch o online, basato su istanza o basato su modello.

In un progetto di machine learning, si raccolgono i dati in un set di addestramento, che viene successivamente passato all'algoritmo di machine learning per la fase di apprendimento.

Se l'algoritmo è basato su modello, si regolano alcuni parametri per adattare il modello al set di addestramento. Spesso è necessario modificare anche gli iperparametri dell'algoritmo in uso.

Se l'algoritmo è basato su istanza, esso apprende gli esempi a memoria e generalizza a nuove istanze utilizzando una misura di somiglianza per confrontarli con le istanze apprese.

Se il set di addestramento è troppo piccolo, se i dati non sono rappresentativi o contengono rumore, molto probabilmente il sistema non funzionerà bene.

Il modello deve quindi essere valutato e se necessario perfezionato. Per valutare il modello serve sapere quanto bene generalizza sui nuovi casi, e per farlo è necessario mettere in produzione il modello e monitorare le sue prestazioni.

Purtroppo questa procedura richiede tempo e se il modello non funziona bene i risultati potrebbero essere molto deludenti.

Una soluzione per valutare le prestazioni del modello evitando i problemi appena descritti è quella di dividere i dati disponibili in due set: il set di addestramento e il set di test. Si andrà ad addestrare il modello utilizzando il set di addestramento e lo si testerà successivamente sul set di test. Il set di test deve essere usato solo per la validazione finale.

Il tasso di errore sui nuovi casi è chiamato *errore di generalizzazione* e valutando il modello sul test di set si ottiene una stima di questo errore. Questo valore indica il rendimento del modello sulle nuove istanze che non ha mai visto prima.

Se l'errore di addestramento è basso (ovvero, il modello fa pochi errori sul set di addestramento) ma l'errore di generalizzazione è alto, significa che il modello sta adattando eccessivamente i dati di addestramento.

Una soluzione a questo problema è chiamata *convalida di controllo*. Si tengono fuori parte dei dati del set di addestramento, creando un nuovo set ridotto che viene chiamato *set di convalida* e su di esso si valutano diversi modelli candidati, modificando i vari iperparametri, e si seleziona alla fine quello migliore. Una volta selezionato il modello più efficiente lo si addestra sul set di addestramento completo che comprende anche i dati del set di convalida, e questo fornisce il modello finale. Alla fine si valuta questo modello finale sul sul set di test per ottenere una stima dell'errore di generalizzazione.

Altri criteri per la valutazione delle prestazioni sono:

- nei problemi di classificazione:

  1. *l'accuratezza della classificazione*: (si esprime in percentuale), ossia il rapporto fra i pattern correttamente classificati e i pattern classificati

     $$accuratezza = \frac{pattern\ correttamente\ classificati}{pattern\ classificati} \tag{2.3}$$

     l'*errore di classificazione* è il valore complementare dell'accuratezza

     $$errore\ di\ classificazione = 100\% - accuratezza \tag{2.4}$$

  2. la definizione di una *soglia*. Si assegna il pattern alla classe più probabile, solo quando la probabilità è superiore alla soglia

3. nei problemi di classificazione binaria dove le due classi corrispondono a esempi positivi (la classe corretta) o negativi si possono utilizzare le seguenti tecniche:
   - *precision*: indica quanto il sistema è accurato:

$$precision = \frac{tp}{tp + fp} \tag{2.5}$$

     dove *tp* sono i *True Positive*, un pattern positivo è stato assegnato ai positivi e *fp* sono i *False Positive*, un pattern negativo è stato erroneamente assegnato ai positivi
   - *recall*: indica quanto un sistema è selettivo:

$$recall = \frac{tp}{tp + fn} \tag{2.6}$$

     dove *tp* sono i *True Positive*, un pattern positivo è stato assegnato ai positivi e *fn* sono i *False Negative*, un pattern positivo è stato erroneamente assegnato ai negativi
4. guardare la *matrice di confusione*. L'idea generale è contare il numero di volte in cui le istanze di classe A sono classificate come classe B

- nei problemi di regressione:

1. si valuta in genere l'RMSE (*Root Mean Squared Error*), ossia la radice della media dei quadrati degli scostamenti tra valore vero e valore predetto

$$RMSE(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2} \tag{2.7}$$

2. il *coefficiente di determinazione* R^2, che indica una proporzionalità tra la variabilità dei dati e la correttezza del modello statistico utilizzato R^2 è un valore compreso fra 0 e 1, se pari a 1 esiste una perfetta correlazione fra il fenomeno analizzato e il modello statistico

$$R^2 = 1 - \sum \frac{(y_i - \hat{y}_i)^2}{(y_i - \bar{y}_i)^2} \tag{2.8}$$

3. il *valore massimo del residuo max_error*

2.3.4 Conclusioni

In generale per ottenere delle buone prestazioni dai modelli di machine learning è necessario che i dati usati per i set di addestramento, di convalida o di test siano i più rappresentativi possibili dei dati che poi ci si aspetta di utilizzare in produzione.

Meglio non utilizzare approcci di machine learning per problemi sui quali non si hanno a disposizione sufficienti esempi per il training e per il test.

Automatizzare fin dall'inizio le procedure di valutazione delle prestazioni, visto che poi saranno usate molte volte. Si risparmierà molto tempo successivamente.

Confrontare le prestazioni del sistema solo con altri modelli addestrati sullo stesso dataset e con lo stesso protocollo.

Infine, ma molto importante, scrivere codice strutturato, ordinato, eseguendo debug incrementali e unit testing. Gli algoritmi di machine learning non sono "esatti" e trovare bug nel codice può essere molto difficile.

2.4 Strumenti di Sviluppo

In questa sezione verranno illustrati gli strumenti utilizzati durante lo svolgimento del progetto. In particolare parlerò di *Python*, e delle principali librerie usate nello sviluppo del progetto di tesi, *Pandas*, *Scikit-Learn* e *Matplotlib*. Gli ambienti di lavoro utilizzati sono la distribuzione *Anaconda* e la piattaforma *Google Colab* che permettono l'accesso ai *notebook Jupyter*. Infine gli algoritmi di machine learning per l'analisi a regressione: *Decision Tree*, *Random Forest*, *AdaBoost*, *XGBoost*.

2.4.1 Python

Negli ultimi trent'anni la tecnologia dei calcolatori e le relative applicazioni sono diventate sempre più sofisticate. Di riflesso i linguaggi di programmazione e le metodologie di risoluzione dei problemi hanno aumentato notevolmente la loro curva di apprendimento.

Nei primi anni novanta, Guido Van Rossum, progettò quello che sarebbe poi diventato il linguaggio *Python*.

Non era soddisfatto dei linguaggi disponibili all'epoca, erano stati progettati per scrivere programmi di grandi dimensioni che venissero eseguiti velocemente. Lui invece, aveva la necessità di un linguaggio che consentisse di scrivere e creare programmi più rapidamente. Python fu quindi progettato per avere una sintassi molto più semplice e chiara dei linguaggi di programmazione più diffusi come Java, C, C++.

Python gode dei vantaggi qui elencati:

- Ha una sintassi semplice e consueta. Gli enunciati sono molto simili a quelli dello pseudocodice usato per descrivere gli algoritmi e le espressioni usano la notazione comunemente adottata nell'algebra.

- Ha una semantica sicura, qualunque espressione o enunciato che violi la sintassi del linguaggio produce un messaggio di errore.

- Si adatta facilmente alle esigenze crescenti. Risulta semplice per un principiante per scrivere i primi programmi, ma il linguaggio contiene anche tutte le caratteristiche più avanzate dei moderni linguaggi di programmazione.

- E' altamente interattivo. Espressioni ed enunciati possono essere digitati direttamente nel prompt dell'interprete. Successivamente possono essere costruite sezioni di codice più lunghe memorizzate in file di script, per caricarle ed eseguirle come moduli o applicazioni indipendenti.

- E' un linguaggio general purpose adatto alle applicazioni più diverse. Mette a disposizione risorse per le applicazioni più moderne come quelle che coinvolgono le reti dei calcolatori, l'elaborazione multimediale e la data science.

- Python è gratuito e ampiamente diffuso in ambito industriale. La comunità di programmatori in linguaggio Python è estremamente vasta e i framework e le librerie disponibili per le più disparate applicazioni sono in continuo aumento.

Python presenta anche degli svantaggi.

- Velocità: Python è un linguaggio interpretato e quindi è relativamente più lento di altri linguaggi di programmazione.

- Python non è adatto per ambienti Android e iOS. Tuttavia, può essere utilizzato con ulteriori sforzi.
- Consumo di memoria: Python consuma una quantità significativa di RAM. Il processo diventa più lento quando è necessario accedere a più oggetti.
- Livelli di accesso al database: i livelli di accesso al database di Python sono sottosviluppati rispetto a Java Database Connectivity (JDBC) e Open Database Connectivity (ODBC), rendendolo una connettività di database meno utilizzata.
- Threading: il threading o il flusso di più funzioni contemporaneamente è uno svantaggio in Python a causa del suo Global Interpreter Lock (GIL).

Perché Python in questo elaborato di tesi?

Python è diventato il linguaggio di programmazione più popolare per la *data science* ed il *Machine Learning*. Rispetto ad *R*, altro linguaggio molto usato per le analisi statiche, Python risulta facilmente integrabile con ambienti di lavoro più complessi.

Python dispone di una grande quantità di librerie utili per il calcolo scientifico e il machine learning. Essendo un linguaggio interpretato, le prestazioni per i compiti ad elevata intensità di calcolo sono inferiori rispetto a quelle dei linguaggi di programmazione a basso livello. Le librerie di estensione come *Numpy* e *Scipy* sono state realizzate con linguaggi a basso livello come *Fortran* e *C* per accelerare le operazioni in Python.

2.4.2 Pandas

Pandas è una potente libreria utilizzabile in Python, che permette l'acquisizione di grandi set di dati da fonti esterne e l'utilizzo di statistiche, serie temporali e grafici interattivi. Le caratteristiche più importanti dei Pandas sono la *vettorizzazione* e *l'allineamento dei dati*. La vettorizzazione, attraverso *Numpy* di cui i Pandas ne sono discendenti, consente di scrivere codice coinciso basato su array, mentre l'allineamento dei dati assicura che non vi siano discrepanze di dati quando si lavora con più set di dati.

DataFrame e *Series* sono le strutture dati principali nei Pandas.

Pandas supporta l'integrazione con molti formati di file o origini dati fuori dagli schemi (csv, excel, sql, json, parquet,...). L'importazione dei dati da ciascuna di queste origini dati è fornita dalla funzione con il prefisso *read_ **. Allo stesso modo, i metodi *to_ ** vengono utilizzati per memorizzare i dati.

2.4.3 Scikit-Learn

Scikit-learn è una libreria di machine learning open source che supporta l'apprendimento supervisionato e non supervisionato. Fornisce inoltre vari strumenti per l'adattamento del modello, la pre-elaborazione dei dati, la selezione del modello, la valutazione del modello e molte altre utilità.

Scikit-Learn si muove su svariati campi di lavoro che vanno dalla classificazione, la regressione, il clustering, la riduzione della dimensionalità dei dataset, la selezione dei modelli, il pre-processing.

Ad eccezione del modello *XGBoost* che ha una sua libreria, tutti gli altri modelli di machine learning utilizzati in questa tesi, sono stati importati dai moduli di *Scikit-learn*.

2.4.4 Matplotlib

Matplotlib è una libreria completa per la creazione di visualizzazioni statiche, animate e interattive in Python. In questo progetto viene usata assieme a *Seaborn* che è un'interfaccia di alto livello per disegnare grafici statistici con Matplotlib. Seaborn rende la visualizzazione, una parte centrale dell'esplorazione e della comprensione di set di dati complessi.

2.4.5 Anaconda e Google Colab

Gli ambienti di sviluppo per la scrittura del codice, che ho usato in questa tesi sono:

- *Anaconda*: probabilmente la distribuzione Python più popolare utilizzata per la scienza dei dati. Viene fornita con centinaia di pacchetti di terze parti preinstallati inclusi i *notebook Jupyter*. Questi ultimi consentono di combinare codice Python eseguibile con testo formattato, immagini e grafici in un taccuino interattivo che viene eseguito nel browser.

- *Google Colaboratory (Colab)*: colab è una piattaforma gratuita che permette a chiunque di scrivere ed eseguire codice Python attraverso un browser, utilizzando i notebook Jupyter.

2.4.6 I modelli per il machine learning

2.4.6.1 Decision Tree

Decision Tree (alberi decisionali) è l'algoritmo di machine learning principale da cui vengono creati i modelli XGBoost. Gli alberi decisionali si fondano sulle tecniche di bagging, il cui funzionamento consiste nel raccogliere le previsioni dai weak leaner, confrontarle e ottenere un'unica previsione.

Gli alberi decisionali tendono a sovradattare i dati. Possono mappare troppo da vicino i dati di addestramento, un problema in termini di varianza e bias. La regolazione fine degli iperparametri è una soluzione per prevenire l'overfitting. Un'altra soluzione è aggregare le previsioni di molti alberi, una strategia utilizzata da Random Forests e XGBoost. Gli alberi decisionali funzionano suddividendo i dati in rami. I rami vengono seguiti fino alle foglie dove vengono fatte le previsioni.

Decision Tree è selezionabile tra i modelli della libreria Scikit-Learn e può essere usato sia come classificatore che come regressore.

2.4.6.2 Random Forest

Random Forest (foreste casuali) sono *ensemble* di alberi decisionali. Un ensemble o metodo di insieme, è un modello di apprendimento automatico che aggrega le previsioni dei singoli modelli. Poiché i metodi d'insieme combinano i risultati di più modelli, sono meno soggetti a errori e quindi tendono a funzionare meglio.

Le foreste casuali sono classificate come algoritmi di bagging perché prendono gli aggregati di campioni bootstrap.

I metodi di apprendimento ensemble sfruttano i modelli di apprendimento deboli, i quali utilizzati singolarmente non raggiungono un buon livello di accuratezza, ma combinati tra loro possono generalizzare bene creando un modello forte. In questo contesto con il termine "weak learner" si intende un modello capace di generare delle risposte leggermente migliori di quelle che si otterrebbero in modo casuale, mentre con "strong learner" si indica un modello che tende ad avvicinarsi a un modello ideale, e può risolvere anche problemi tipici dei modelli tradizionali come l'*overfitting*.

L'obiettivo delle metodologie di ensemble learning come Random Forest è quello di ottenere una predizione attendibile, riducendo varianza e distorsione.

Anche Random Forest è disponibile tra i modelli della libreria Scikit-Learn e può essere usato sia come classificatore che come regressore.

2.4.6.3 Gradient Boosting con AdaBoost

Gli algoritmi di *Gradient Boosting* (aumento del gradiente) sfruttano la tecnica del boosting che consiste nell'unire classificatori deboli (weak learner) al fine di ottenere un classificatore che abbia una migliore accuratezza rispetto ai precedenti. Il funzionamento è simile a quello di un algoritmo per costruire alberi decisionali, ma in questo caso l'elaborazione si ripete per più iterazioni, creando in modo adattivo la successione di classificatori deboli.

Il dataset che viene fornito in input verrà chiamato training set ed è composto da un certo numero di oggetti i quali hanno tutti lo stesso peso in termini di importanza. A partire da questi elementi viene costruito il primo albero decisionale, che corrisponde alla prima ipotesi di previsione. Successivamente l'algoritmo verifica l'esattezza dei risultati ottenuti e aumenta l'importanza dei risultati sbagliati. Il training set non sarà più uguale a quello iniziale perché gli elementi avranno pesi diversi.

In seguito, l'algoritmo costruisce un secondo albero decisionale prendendo in input il set di dati pesato ottenuto precedentemente. A questo punto i risultati ottenuti verranno nuovamente verificati e, come in precedenza, verrà attribuito un peso maggiore ai risultati sbagliati. Il ciclo si ripete n volte, costruendo n alberi decisionali e migliorando la previsione iterativamente.

Con il boosting ci si concentra maggiormente sull'insieme dei dati che non si riesce a classificare in modo corretto.

AdaBoost è uno dei primi e più popolari modelli di Gradient Boosting. In AdaBoost, ogni nuovo albero regola i suoi pesi in base agli errori degli alberi precedenti. Viene prestata maggiore attenzione alle previsioni che sono andate storte regolando i pesi che influiscono su quei campioni a una percentuale più alta.

AdaBoost è disponibile tra i modelli della libreria Scikit-Learn e può essere usato sia come classificatore che come regressore.

2.4.6.4 XGBoost

XGBoost, noto anche come Extreme Gradient Boosting è un metodo di ensemble, il che significa che è composto da diversi modelli di machine learning che si combinano per lavorare insieme. Rispetto a Gradient Boosting, XGBoost utilizza una formalizzazione del modello più regolare, per cercare di controllare l'overfitting.

La parte *extreme* si riferisce al superamento dei limiti del calcolo per ottenere guadagni in termini di precisione e velocità. La crescente popolarità di XGBoost è in gran parte dovuta al suo impareggiabile successo nelle competizioni di *Kaggle*. Nelle competizioni Kaggle, i concorrenti creano modelli di apprendimento automatico nel tentativo di fare le migliori previsioni e vincere premi in denaro redditizi. Rispetto ad altri modelli, XGBoost ha schiacciato la concorrenza.

E' una libreria open source che permette di costruire un modello e fare previsioni, implementando il framework Gradient Boosting.

XGBoost è disponibile in diversi package per diversi linguaggi di programmazione, fra questi Python, R, ma anche Swift per iOS.

XGBoost implementa diversi parametri di configurazione (iperparametri) che consentono di aumentare la precisione del modello durante le fasi di addestramento.

In questa tesi ho usato il metodo *XGBRegressor*, fornito direttamente dalla libreria XGBoost.

Capitolo 3

Esperimenti

In questo capitolo si vedranno tutte le *esperimentazioni* fatte per produrre i risultati dettagliati nella tesi.

Il focus sarà sui seguenti punti:

- Il collaudo *termodinamico - prestazionale* del refrigeratore AST 2 140
- La progettazione: i *dataset*. Le fasi di *Training* e *Test*
- Implementazione dei *modelli* di machine learning
- I *risultati* ottenuti dalle esperimentazioni

3.1 Il Collaudo Termodinamico

Per poter ottenere un campione di dati reali significativi, sui quali poi verificare non solo il predicting degli algoritmi di *machine learning*, ma anche lo stesso simulatore *VBA*, l'azienda ha dato la disponibilità per realizzare, costruire e collaudare un refrigeratore d'acqua condensato ad aria, della gamma AST 2, modello 140. Il numero *140* rappresenta la potenza elettrica totale del chiller, espressa in *kW*.

Il modello AST 2 140, nelle condizioni di lavoro nominali stabilite da progetto, è in grado di fornire una potenza frigo di 508,4 kW.

3.1.1 Il piano prove

Il piano prove è una *specifica di produzione*, che contiene la check-list dei passaggi, che gli operatori del reparto collaudi effettuano per validare la messa in

funzionamento di un impianto.

Per il chiller AST 2 140, in prova per il tirocinio, oltre alla check-list dei collaudi standard, sono stati aggiunti anche tutti i check per la rilevazione dei dati necessari al dataset di tesi. In particolare quelli che misuravano prestazioni e consumi energetici con l'opzione *HP Flottante* abilitata, e il più importante, il dato che è poi diventato la variabile di *target* dei modelli per l'analisi a regressione, ossia il numero di giri al minuto degli elettroventilatori a regolazione elettronica.

3.1.2 Il collaudo

Il collaudo del refrigeratore è stato fatto nei primi mesi del febbraio 2021 durante il periodo invernale, dove le basse temperature ambiente, hanno permesso di eseguire le prove in condizioni favorevoli ad *HP Flottante*.

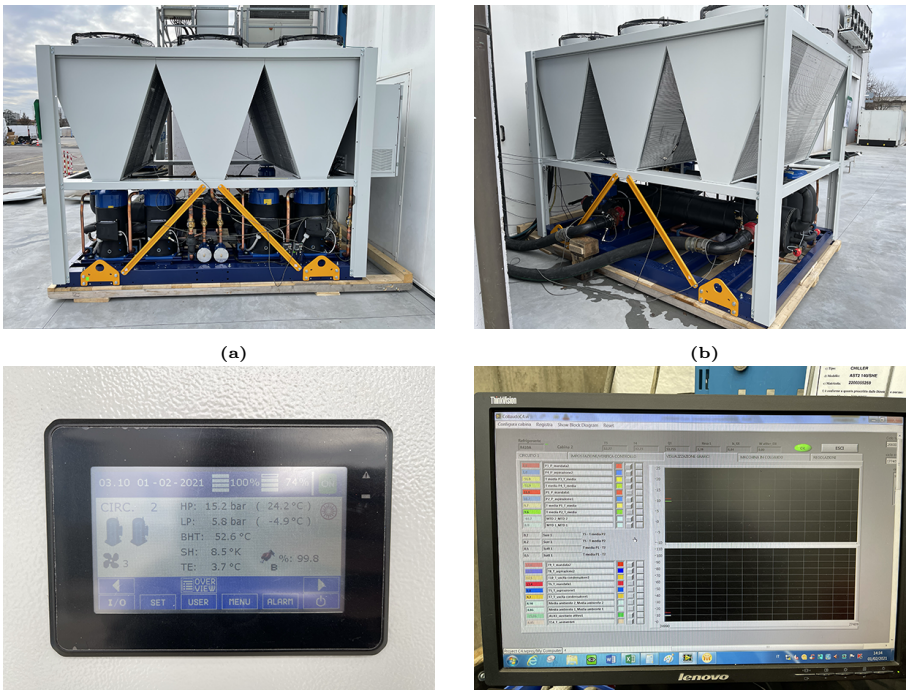


Figura 3.1: Il collaudo del modello AST 2 140

Durante i test sono stati registrati diversi parametri di collaudo, tutti quelli necessari al dataset di tesi, più altri per il funzionamento dell'impianto. I parametri in ingresso, in particolare la *temperatura ambiente*, sono stati acquisiti nelle condizioni ambientali in essere mentre veniva eseguita la registrazione.

In risultato finale dei collaudi è un report di 40 acquisizioni, fatte sui punti di lavoro stabiliti dal piano prove, salvato in formato Excel.

3.1.3 Il dataset

Il dataset finale estrapolato dal report dei collaudi è della stessa forma di quello ottenuto dalla simulazione con il software *VBA*. Le righe dei record sono passate da 40 a **32**, in quanto 8 acquisizioni, non sono state considerate valide per l'inserimento dei dati nel dataset. Queste ultime presentavano colonne con valori mancanti.

3.2 Progettazione

Il seguente paragrafo ha la funzione di illustrare i passaggi che sono stati eseguiti nel progetto.
L'obiettivo del progetto è quello di ottenere un modello che sia capace di valutare i dati a disposizione e sulla base di questi effettuare previsioni in modo più accurato. Per il raggiungimento di tale scopo sono stati proposti e valutati quattro algoritmi di machine learning basati sul bagging e sul boosting.

3.2.1 I dataset

I dataset a disposizione sono tre. Il primo è stato ottenuto dal software di simulazione scritto in linguaggio VBA. Il secondo dalle prove di collaudo sul refrigeratore AST 2 140. La loro struttura iniziale è sostanzialmente diversa. Il simulatore ha fornito un dataset con oltre 68000 record di prestazioni *simulate* del refrigeratore AST 2 140, mentre il dataset fornito dal collaudo ha reso disponibili poco più di una trentina di record che però rappresentano dati *reali*. Il terzo dataset verrà presentato alla fine del capitolo.

Il dataset dei dati simulati è stato usato per valutare i modelli di machine learning nelle fasi di analisi e *training*.
Il dataset ottenuto dai collaudi è invece stato tenuto da parte ed usato alla fine per la fase di *testing*.

Ogni riga dei dataset rappresenta una simulazione, mentre ogni colonna una caratteristica termodinamica o prestazionale del modello AST 2 140.

La variabile su cui si vuole effettuare la previsione è il numero di rotazioni al minuto (*giri/minuto*) degli elettroventilatori con regolazione elettronica (EC) montati nel refrigeratore.

Nel primo dataset (dati simulati) le caratteristiche sono formate da **51 variabili**, le righe delle simulazioni sono invece **68175 record**.
Il dataset dei dati reali invece ha **9 variabili** e **32 record**.
Nelle successive fasi del processo, il primo dataset verrà ridotto nel numero di variabili, per ottenere una struttura dati con lo stesso numero di colonne, ma diverso numero di righe, rispetto alla struttura dati del secondo dataset. Il terzo è ultimo dataset è stato preparato direttamente in formato *csv*, ed ha la stessa forma, stesso numero di colonne e di righe, del secondo dataset.

3.2.2 Pre-processing

Per poter effettuare delle analisi e delle previsioni attendibili, è necessario preparare i dati per gli algoritmi di machine learning.
Nei passaggi che seguiranno illustrerò il codice in Python, le librerie usate per importare i dataset nei notebook Jupyter e tutte le fasi di analisi, pulizia, ridimensionamento e standardizzazione dei dati.

I notebook Jupyter sono utilizzati all'interno dell'ambiente Google Colab. I dataset sono inizialmente dei file excel (.xlsx), posizionati nel mio cloud in Google Drive.

```
# importo le librerie

import numpy as pd
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt

# monto il driver virtuale per caricare i file da Google Drive

from google.colab import drive
drive.mount("/content/gdrive")

# importo il dataset da file excel dei dati del simulatore VBA

```

```
dataset_simulatore_VBA = pd.read_excel('/content/gdrive/My Drive/TESI/Jupyter Notebook/dataset_simulatore_VBA.xlxs')

# cambio il tipo di dato di alcune Series da int a float

dataset_simulatore_VBA['Temperatura Water IN'] = dataset_simulatore_VBA['Temperatura Water IN'].astype(float)
dataset_simulatore_VBA['Temperatura Water OUT'] = dataset_simulatore_VBA['Temperatura Water OUT'].astype(float)
dataset_simulatore_VBA['Temperatura ambiente'] = dataset_simulatore_VBA['Temperatura ambiente'].astype(float)
dataset_simulatore_VBA['RMP ventilatori 1'] = dataset_simulatore_VBA['RMP ventilatori 1'].astype(float)

# uso il metodo info() per descrivere i dati

dataset_simulatore_VBA.info()

# info

RangeIndex: 68175 entries
Data columns (total 52 columns)

Index  Column                    Count  Dtype
0      Temperatura Water IN      68175  float64
1      Temperatura Water OUT     68175  float64
2      Temperatura ambiente      68175  float64
3      C1 R A0                   68175  float64
4      C1 R A1                   68175  float64
5      C1 R A2                   68175  float64
5      C1 R A2                   68175  float64
5      C1 R A2                   68175  float64
5      C1 R A2                   68175  float64
5      C1 R A2                   68175  float64
5      C1 R A2                   68175  float64
5      C1 R A2                   68175  float64
5      C1 R A2                   68175  float64
5      C1 R A2                   68175  float64
5      C1 R A2                   68175  float64
6      C1 R A3                   68175  float64
7      C1 R A4                   68175  float64
8      C1 R A5                   68175  float64
9      C1 R A6                   68175  float64
10     C1 R A7                   68175  float64
```

```
11      C1 R A8                    68175  float64
12      C1 R A9                    68175  float64
13      C1 P A0                    68175  float64
14      C1 P A1                    68175  float64
15      C1 P A2                    68175  float64
16      C1 P A3                    68175  float64
17      C1 P A4                    68175  float64
18      C1 P A5                    68175  float64
19      C1 P A6                    68175  float64
20      C1 P A7                    68175  float64
21      C1 P A8                    68175  float64
22      C1 P A9                    68175  float64
23      C2 R A0                    68175  float64
24      C2 R A1                    68175  float64
25      C2 R A2                    68175  float64
26      C2 R A3                    68175  float64
27      C2 R A4                    68175  float64
28      C2 R A5                    68175  float64
29      C2 R A6                    68175  float64
30      C2 R A7                    68175  float64
31      C2 R A8                    68175  float64
32      C2 R A9                    68175  float64
33      C2 P A0                    68175  float64
34      C2 P A1                    68175  float64
35      C2 P A2                    68175  float64
36      C2 P A3                    68175  float64
37      C2 P A4                    68175  float64
38      C2 P A5                    68175  float64
39      C2 P A6                    68175  float64
40      C2 P A7                    68175  float64
41      C2 P A8                    68175  float64
42      C2 P A9                    68175  float64
43      Alfa h2o                   68175  float64
44      Alfa R410A                 68175  float64
45      Alfa zdp                   68175  float64
46      Superficie                 68175  float64
47      Beta                       68175  float64
48      OUTPUT                     0      float64
49      Resa frigo C1              68175  float64
50      Potenza assorbita TOT C1   68175  float64
51      RMP ventilatori 1          68175  float64

# ci sono 68175 record per ogni istanza, tutti i dati sono di
    tipo float64
```

```
# La colonna OUTPUT e' una colonna segnaposto, sara' eliminata

# il metodo describe() mostra un riepilogo degli attributi numerici. Uso describe() per valutare le tre caratteristiche di input fondamentali "Temperatura Water IN", "Temperatura Water OUT", "Temperatura ambiente" e per valutare la variabile di target "RMP ventilatori 1"

dataset_simulatore_VBA["Temperatura Water IN"].describe()

# describe

count    68175.000000
mean        16.000000
std          4.320525
min          9.000000
25%         12.000000
50%         16.000000
75%         20.000000
max         23.000000

# la temperatura di ingresso acqua nell'evaporatore varia in un range fra i 9 C di minimo e i 23 C di massimo, con una media di 9 C e una deviazione standard di 4.3 C

dataset_simulatore_VBA["Temperatura Water OUT"].describe()

# describe

count    68175.000000
mean        11.000000
std          4.397001
min          3.000000
25%          7.000000
50%         11.000000
75%         15.000000
max         19.000000

# la temperatura di uscita acqua dall'evaporatore varia in un range fra i 3 C di minimo e i 19 C di massimo, con una media di 11 C e una deviazione standard di 4.4 C

# dai due report sopra si puo' verificare che gli ingressi e le uscita acqua evaporatore, sono stati impostati con un delta di
```

```
    temperatura di 5 C per ogni grado ambiente con una variazione
    di 1 C in uscita acqua

dataset_simulatore_VBA["Temperatura ambiente"].describe()

# describe

count     68175.000000
mean         17.554455
std          16.193551
min         -10.000000
25%           4.000000
50%          18.000000
75%          32.000000
max          45.000000

# la temperatura ambiente varia in un range fra i -10 C di minimo
    e i 45 C di massimo, con una media di 17.6 C e una deviazione
    standard di 16.2 C

dataset_simulatore_VBA["RMP ventilatori 1"].describe()

# describe

count     68175.000000
mean        675.925266
std         215.868607
min          95.000000
25%         499.000000
50%         750.000000
75%         850.000000
max         950.000000

# il numero di giri dei ventilatori varia in un range fra i 95
   giri/minuti di minimo a 950 giri/minuto di massimo, con una
   media di 676 giri/minuto e una deviazione standard di 216 giri
   /minuto

# le colonne presenti nel dataset da "C1 R A0" fino a "C1 P A9" e
    da "C2 R A0" fino a "C2 P A9" rappresentano i  coefficienti
   delle polinomiali dei modelli di compressore installati sul
   refrigeratore AST 2 140

# queste polinomiali sono usate dal simulatore VBA per poter
```

```
    calcolare la resa e la potenza assorbita di ogni singolo
    conpressore. I compressori interessati dalla simulazione sono
    due, indicati nel simulatore con C1 e C2

# le colonne presenti nel dataset etichettati con "Alfa h2o", "
    Alfa R410A", "Alfa zdp", "Superficie", rappresentano i
    coefficienti di scambio termico dell'evaporatore installato
    nel refrigeratore AST 2 140

# tutte le colonne indicate sopra, sia quelle inerenti ai
    compressori, sia quelle dedicate all'evaporatore saranno
    eliminate dal dataset. Infatti la base dati sta simulando solo
     il modello AST 2 140, e  non tutti i modelli della gamma AST
    2. Da tutto cio' si evince che, i coefficienti compressori/
    evaporatore, descrivendo solo il modello 140, risultano non
    significativi in questa analisi, perche' costanti su tutti i
    68175 record

# elimino dal dataset tutte le colonne sopra indicate piu' la
    colonna segnaposto "OUTPUT"

dataset_simulatore_VBA = dataset_simulatore_VBA.drop(["C1 R A0",
"C1 R A1", "C1 R A2" ,"C1 R A3", "C1 R A4", "C1 R A5", "C1 R A6",
"C1 R A7", "C1 R A8", "C1 R A9", "C1 P A0", "C1 P A1", "C1 P A2",
"C1 P A3", "C1 P A4", "C1 P A5", "C1 P A6", "C1 P A7", "C1 P A8",
"C1 P A9", "C2 R A0", "C2 R A1", "C2 R A2", "C2 R A3", "C2 R A4",
"C2 R A5", "C2 R A6", "C2 R A7", "C2 R A8", "C2 R A9", "C2 P A0",
"C2 P A1", "C2 P A2", "C2 P A3", "C2 P A4", "C2 P A5", "C2 P A6",
"C2 P A7", "C2 P A8", "C2 P A9", "Alfa h2o", "Alfa R410A",
"Alfa zdp", "Superficie", "OUTPUT"],  axis=1)

# uso nuovamente il metodo info() per rivedere i dati dopo il
    ridimensionamento del dataset

dataset_simulatore_VBA.info()

# info

RangeIndex: 68175 entries
Data columns (total 7 columns)

Index   Column                      Count   Dtype
0       Temperatura Water IN        68175   float64
1       Temperatura Water OUT       68175   float64
```

```
2       Temperatura ambiente       68175  float64
3       Beta                       68175  float64
4       Resa frigo C1              68175  float64
5       Potenza assorbita TOT C1   68175  float64
6       RMP ventilatori 1          68175  float64

# visualizzo questa volta, Figura 3.2, i dati del dataset attraverso istogrammi tracciati per ogni attributo numerico usando il metodo hist()

dataset_simulatore_VBA.hist(bins=50, figsize=(20,15))
plt.show()

# calcolo i coefficienti di correlazione standard fra la variabile di target "RMP ventilatori 1" e le altre variabili, utilizzando il metodo corr()

corr_matrix = dataset_simulatore_VBA.corr()
pcorr_matrix["RMP ventilatori 1"].sort_values(ascending=False)

# correlazioni

RMP ventilatori 1           1.000000
Temperatura ambiente        0.796209
Potenza assorbita TOT C1    0.533380
Resa frigo C1               0.184604
Beta                        0.121633
Temperatura Water OUT       0.108351
Temperatura Water IN        0.107025

# c'e' una forte correlazione fra il numero di rotazioni al minuto dei ventilatori e la temperatura ambiente. Questo e' corretto se aumenta la temperatura ambiente aumenta la pressione nel condensatore, e questo determina un aumento dei giri dei ventilatori.In questo modo aumenta la portata d'aria nel condensatore per permettere di smaltirne il calore accumulato ed abbassarne di conseguenza la pressione e la temperatura di lavoro

# si puo' osservare una relazione di proporzionalita' anche fra il numero di rotazioni al minuto dei ventilatori e la potenza totale assorbita, ma con un valore piu' basso 0.533380, rispetto a quello con la temperatura ambiente 0.796209.Questo e' dovuto al fatto che la potenza assorbita totale comprende
```

```
anche la potenza assorbita dei compressori, che ha un impatto maggiore rispetto a quella dei ventilatori

# inserisco nel dataset "dataset_simulatore_VBA", la caratteristica "Potenza Assorbita Ventilatori", perche' molto probabilmente la correlazione fra il numero di giri dei ventilatori e la sola potenza assorbita dagli stessi aumentera

# per fare questo inserimento, applico la stessa formula matematica che calcola gli assorbimenti dei ventilatori, inserita nel simulatore VBA

dataset_simulatore_VBA["Potenza Assorbita Ventilatori"] = 3 * 1.62 * (0.00000228 * dataset_simulatore_VBA["RMP ventilatori 1"]**2 - (0.00142294 * dataset_simulatore_VBA["RMP ventilatori 1"]) + 0.29068568)

# il valore 3, usato nella formula sopra, indica il numero di ventilatori attivi nel refrigeratore AST 2 140
# Il valore 1.62, usato nella formula sopra, indica la potenza assorbita in kW da ogni singolo ventilatore

# calcolo i coefficienti di correlazione standard fra la variabile di target "RMP ventilatori 1" e le altre variabili, utilizzando il metodo corr()

corr_matrix = dataset_simulatore_VBA.corr()
corr_matrix["RMP ventilatori 1"].sort_values(ascending=False)

# correlazioni

RMP ventilatori 1                1.000000
Potenza Assorbita Ventilatori    0.951927
Temperatura ambiente             0.796209
Potenza assorbita TOT C1         0.533380
Resa frigo C1                    0.184604
Beta                             0.121633
Temperatura Water OUT            0.108351
Temperatura Water IN             0.107025

# la correlazione fra il numero di rotazioni al minuto dei ventilatori e la potenza assorbita ventilatori e' molto alta 0.951927, e questo e' corretto se i ventilatori girano piu velocemente assorbono di piu'
```

```

# inserisco nel dataset "dataset_simulatore_VBA", la caratteristica "EER" (Energy Efficency Ratio). Indica il rapporto fra l'effetto frigorifero del refrigeratore e la {potenza meccanica scambiata. E' un parametro prestazionale che misura l'efficienza elettrica di una macchina termica mentre funziona in raffreddamento

# in questa analisi il paramentro EER serve per vedere la relazioni fra il numero di giri dei ventilatori e le prestazioni del refrigetore. EER come detto e' il rapporto fra la resa frigo e la potenza assorbita totale che sono due delle caratteristiche del dataset

# quello che si dovrebbe ottenere attraverso il calcolo dei coefficienti di correlazione, e' una relazione di "proporzionalita' inversa". Se aumento il numero di  giri dei ventilari, dovuto all'aumento della temperatura ambiente e di conseguenza della pressione di condensazione, aumentano gli assorbimenti (compressori, ventilatori). Complessivamente il valore dell'indice prestazione EER diminuisce, perche' a parita' di resa frigo (il numeratore del rapporto), aumenta la potenza assorbita totale (il denominatore del rapporto)

# aggiungo la colonna EER

dataset_simulatore_VBA["EER"] = dataset_simulatore_VBA["Resa frigo C1"] / dataset_simulatore_VBA["Potenza assorbita TOT C1"]

# calcolo i coefficienti di correlazione standard fra la variabile di target "RMP ventilatori 1" e le altre variabili, utilizzando il metodo corr()

corr_matrix = dataset_simulatore_VBA.corr()
corr_matrix["RMP ventilatori 1"].sort_values(ascending=False)

# correlazioni

RMP ventilatori 1                1.000000
Potenza Assorbita Ventilatori    0.951927
Temperatura ambiente             0.796209
Potenza assorbita TOT C1         0.533380
Resa frigo C1                    0.184604
```

```
Beta                                0.121633
Temperatura Water OUT               0.108351
Temperatura Water IN                0.107025
EER                                -0.582878

# come si puo' vedere il dato -0.582878, conferma la correlazione inversa fra il numero di giri dei ventilatori e il coefficiente prestazionale EER

# rinomino con il metodo rename() e riordino con il metodo reindex() le etichette delle colonne del dataset "dataset_simulatore_VBA", per preparalo alla creazione dei set di training. Per i set di test usero' il dataset "dataset_piano_prove_TR3470", ottenuto durante il collaudo funzionale del refrigeratore. I due dataset hanno la stessa forma, ma diversi record

# rename()

dataset_simulatore_VBA.rename(columns={'Resa frigo C1': 'Resa frigo', 'Potenza assorbita TOT C1': 'Potenza Assorbita Totale', 'RMP ventilatori 1': 'RMP ventilatori'}, inplace=True)

# reindex()

dataset_simulatore_VBA = dataset_simulatore_VBA.reindex(columns=['Temperatura ambiente','Temperatura Water IN','Temperatura Water OUT', 'Beta', 'Resa frigo', 'Potenza Assorbita Totale', 'Potenza Assorbita Ventilatori', 'EER', 'RMP ventilatori'])


# visulizzo graficamente le correlazioni fra i dati attraverso una matrice Heatmap, che sfrutta una libreria Python chiamata seaborn che ho importato all'inizio del codice

corr = dataset_simulatore_VBA.corr()
plt.subplots(figsize = (15,10))
sb.heatmap(corr, xticklabels = corr.columns, yticklabels = corr.columns, annot = True, cmap = sb.diverging_palette (220, 20, as_cmap = True))
plt.show()

# Esporto il dataset in formato csv, pronto per essere utilizzato nel training dei modelli di machine learning
```

```
dataset_simulatore_VBA.to_csv('/content/gdrive/My Drive/TESI/
    Dataset/dataset_simulatore_VBA.csv', sep=";", index=False)
```

Listing 3.1: Python Code Pre-processing Dataset Simulatore VBA

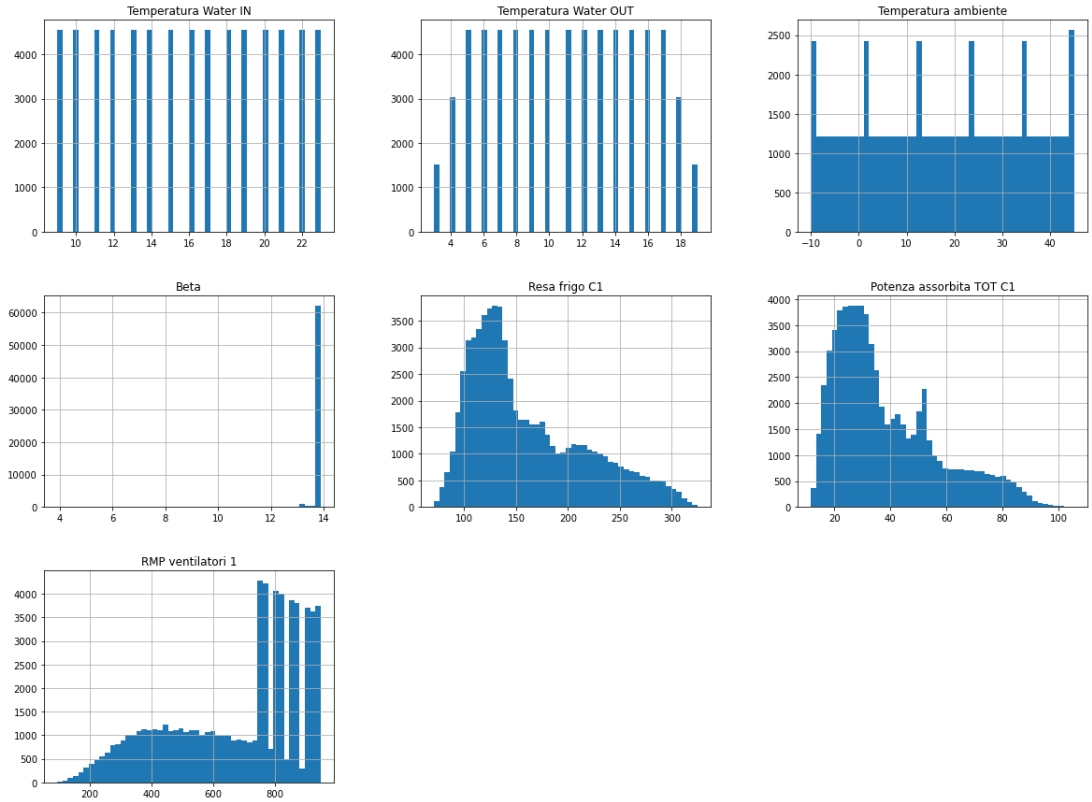


Figura 3.2: Istogrammi dati ottenuti con il metodo hist()

Una matrice *Heatmap*, indicata in figura 3.3, permette di visualizzare in forma grafica, le stesse correlazioni ottenute precedentemente con il metodo *corr()*, ma con un confronto uno ad uno fra tutte le caratteristiche rappresentate.

In colore rosso marcato si possono notare le correlazioni *dirette*, dove fra le più significative e di interesse per questa tesi, vi sono quelle fra "RMP ventilatori" e "Temperatura ambiente", "Potenza Assorbita Ventilatori", "Potenza Assorbita Totale". In colore ciano marcato si possono vedere le correlazioni *indirette*. La più significativa per questo elaborato di tesi quella fra "RMP ventilatori" ed "EER". Importanti sono anche le correlazioni inverse fra "EER" e le caratteristiche "Potenza Assorbita Totale" (-0.63) e "Potenza Assorbita Ventilatori" (-0.62). In HP Flottante attiva, con temperature ambiente di lavoro più basse, gli assorbimenti dei ventilatori proporzionalmente incidono di più rispetto a quelli dei compressori per quanto riguarda la potenza assorbita totale. Questo indica ancora una volta

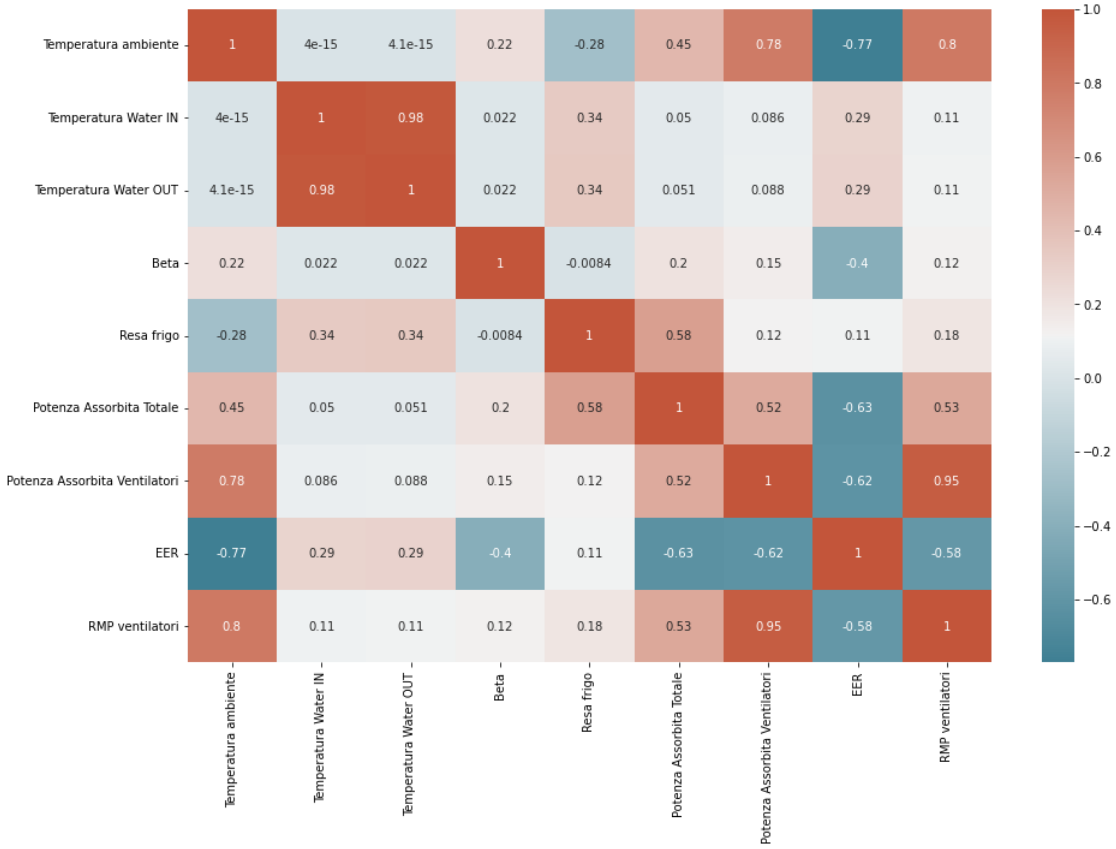


Figura 3.3: Heatmap correlazioni dei dati simulati in VBA

la bontà della soluzione HP Flottante dove può essere abilitata. Una regolazione fine del numero dei giri dei ventilatori può dare grandi vantaggi, riducendo gli assorbimenti di energia elettrica e portando i chiller a performance prestazionali molto più alte.

Concludendo questa analisi di *pre-processing*, posso affermare che i dati ottenuti dal simulatore VBA, hanno confermato tutte quelle che sono le correlazioni fondamentali alla base della soluzione *HP Flottante*.

La regolazione del numero di giri al minuto dei ventilatori elettronici, in condizioni di HP Flottante attiva durante il funzionamento del refrigeratore, è in relazione *diretta* e con una forte dipendenza, alla temperatura ambiente e all'assorbimento dei ventilatori, che a loro volta sono due fattori fortemente correlati fra loro. La stessa regolazione ventilatori è in relazione *indiretta* con le prestazioni del refrigeratore (indice EER). Questo si evince dal fatto che se aumenta la temperatura ambiente, a seguire aumenta la pressione di condensazione, che è la condizione sfavorevole per HP Flottante, il tutto porta ad un aumento della

potenza assorbita, che alla fine determina il calo delle prestazioni del refrigeratore.

3.2.3 Training e Test

Il *Training* rappresenta la parte centrale del processo, in quanto vengono ricercati e generalizzati i *pattern* partendo dai dati che descrivono l'evento che si sta analizzando.

Per questo elaborato di tesi ho a disposizione i due dataset, quello dei dati simulati "*dataset_simulatore_VBA*" e quello dei dati reali "*dataset_piano_prove_TR3470*".

In generale quando si applicano modelli di machine learning supervisionati si dispone solo del dataset dei dati già etichettati. Per ottenere quindi un set di dati su cui effettuare l'addestramento e il successivo test dei modelli, il dataset di cui si dispone viene diviso in due set, uno per la fase di train e l'altro per la fase di test. Queste due partizioni devono essere estratte casualmente, in modo da ottenere campioni rappresentativi. Inoltre per ottenere i due insiemi, il dataset non viene diviso in modo uguali, ma è preferibili destinare un maggior numero di record al set di allenamento, così che i modelli possano apprendere quanto più possibile. Un rischio a cui si può andare incontro durante il training è l'*overfitting*. Esso si verifica quando il modello su cui si sta effettuando l'addestramento, si specializza eccessivamente sull'insieme dei dati, adattandosi così a caratteristiche specifiche del trainSet. Di conseguenza aumenteranno le prestazioni sul set di allenamento, ma diminuiranno sul set di test.

Una tecnica che si può adottare per evitare il sovradattamento è la *Cross-validation*. Con questa tecnica il dataset viene diviso in k sottocampioni, le *fold* del metodo Cross-validation, tutti della stessa dimensione. Ad ogni ripetizione, una partizione viene conservata per la convalida, mentre le altre vengono utilizzate per l'addestramento. Dopo aver effettuato le k iterazioni viene restituito il risultato, ossia il valore dell'errore medio dei campioni di test.

In questo elaborato di tesi, essendo disponibile un campione di set di dati reali proveniente dalla prove di collaudo, non è necessario applicare *Cross-validation*, ma posso utilizzare tutto il dataset dei dati ottenuti dal simulatore VBA per il training.

Per valutare le *prestazioni* dei quattro modelli per i problemi di regressione selezionati per questo elaborato, viene effettuata la misura con la metrica *Root Mean Square Error (RMSE).*

Questa misura da un'idea di quanti errori il sistema fa tipicamente nelle sue previsioni, con un peso maggiore per errori di grandi dimensioni.

L'equazione della formula matematica per calcolare *RMSE* è la seguente:

$$RMSE(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2} \tag{3.1}$$

m è il numero di istanze nel set di dati su cui si sta misurando *RSME.*

$\mathbf{x}^{(i)}$ è il vettore di tutti i valori delle caratteristiche (esclusa l'etichetta) dell'i-esima istanza nel set di dati e $y^{(i)}$ è la sua etichetta (il valore di output desiderato per quell'istanza).

$\mathbf{X}$ è una matrice contenente tutti i valori delle caratteristiche (escluse le etichette) di tutte le istanze nel set di dati.

h è la funzione di predizione del sistema, chiamata anche *ipotesi.* Quando al sistema viene assegnato il vettore di caratteristiche di un'istanza $\mathbf{x}^{(i)}$, restituisce un valore predetto $\hat{y}^{(i)} = h(\mathbf{x}^{(i)})$.

Per l'attendibilità delle performance dei modelli viene calcolato il coefficiente di determinazione R^2.

L'equazione della formula matematica per calcolare R^2 è la seguente:

$$R^2 = 1 - \sum \frac{(y_i - \hat{y}_i)^2}{(y_i - \bar{y}_i)^2} \tag{3.2}$$

y_i sono i dati osservati

$\bar{y}_i$ è la loro media

$\hat{y}_i$ sono i dati stimati dal modello

Infine mostrerò anche il valore massimo del residuo per ognuno dei modelli applicati.

I quattro modelli di *machine learning* in analisi sono:

- *Decision Tree*
- *Random Forest*
- *AdaBoost*
- *XGBoost*

Dopo il loro addestramento sul dataset "*dataset_ simulatore_ VBA*", saranno collaudati nella fase di *testing*, nel dataset "*dataset_ piano_ prove_ TR3470*", che come detto più volte contiene dati reali derivanti dal collaudo del refrigeratore AST 2 140.

In questo elaborato di tesi, ho scelto di testare i modelli di ML, senza impostare nessuno degli *Iperparametri* disponibili per ogni algoritmo. i vari set di iperparametri rimangono nei valori impostati di default.

Come ultima cosa segnalo un ulteriore confronto. Come detto nel capitolo 2, il dataset "*dataset_ simulatore_ VBA*" è stato adattato a simulare la curva di lavoro dei ventilatori, come se stessero lavorando in HP flottante. Il software originale VBA in realtà non prevede l'opzione HP flottante, perché come detto precedentemente questa viene abilitata direttamente dal controllo del refrigeratore senza essere simulata. Il simulatore VBA quindi è in grado di simulare solamente le curve di lavoro dei ventilatori nelle condizioni "standard".

Per questo motivo ho creato un ulteriore dataset "*dataset_ VBAreale_prove_ TR3470*". Il nuovo dataset è stato costruito prendendo il simulatore originale VBA della gamma AST 2, impostando in ingresso i valori misurati nelle 32 prova raccolte dai test, ad esclusione della variabile di target, ossia il numero di giri dei ventilatori. Da queste 32 simulazioni ho estratto 32 valori di giri ventilatori, uno per prova, che sono poi stati confrontati e misurati sempre con la metrica *RMSE*, con il valore reale della corrispondente prova.

Nelle conclusioni di tesi saranno riportati i confronti delle simulazioni *machine learning - dati reali* VS *simulatore reale VBA - dati reali.*

3.3 Implementazione

In questo paragrafo verrà mostrato il lavoro svolto in Python, facendo riferimento al processo spiegato nel paragrafo capitolo.

3.3.1 Pre-processing

I dataset su cui ho lavorato sono relativi alla simulazione fatta con il software in linguaggio VBA, alle prove di collaudo con l'opzione HP flottante abilitata, realizzate sul refrigeratore AST 2 140 e alla simulazione con il simulatore reale della gamma AST 2 mentre simulava i dati delle prove.

Nella fase di pre-precessing vista nel capitolo precedente, sono intervento solo nel dataset dei dati simulati in VBA, dove ho ridimensionato le caratteristiche, ne ho aggiunte di nuove, di interesse per le valutazioni di tesi ed infine ho passato il dataset in un file csv.

I dataset dei dati reali ed il dataset del simulatore reale, erano già stati creati in file csv, e strutturati con le stesse colonne del dataset VBA.

```
# importo i tre dataset

dataset_simulatore_VBA = pd.read_csv('/content/
gdrive/My Drive/TESI/Dataset/dataset_simulatore_VBA.csv',
sep=";")
```

Column Non-Null Count Dtype
— ——— ———— ———

0 Temperatura ambiente 68175 non-null float64
1 Temperatura Water IN 68175 non-null float64
2 Temperatura Water OUT 68175 non-null float64
3 Beta 68175 non-null float64
4 Resa frigo 68175 non-null float64
5 Potenza Assorbita Totale 68175 non-null float64

6 Potenza Assorbita Ventilatori 68175 non-null float64
7 EER 68175 non-null float64
8 RMP ventilatori 68175 non-null float64

```
dataset_piano_prove_TR3470 = pd.read_csv('/content/
gdrive/My Drive/TESI/Dataset/dataset_piano_prove_TR3470.csv',
sep=";")
```

Column Non-Null Count Dtype
— ——— ————— ———
0 Temperatura ambiente 32 non-null float64
1 Temperatura Water IN 32 non-null float64
2 Temperatura Water OUT 32 non-null float64
3 Beta 32 non-null float64
4 Resa frigo 32 non-null float64
5 Potenza Assorbita Totale 32 non-null float64
6 Potenza Assorbita Ventilatori 32 non-null float64
7 EER 32 non-null float64
8 RMP ventilatori 32 non-null float64

```
dataset_VBAreale__prove_TR3470 = pd.read_csv('/content/
gdrive/My Drive/TESI/Dataset/dataset_VBAreale_prove_TR3470.csv',
sep=";")
```

Column Non-Null Count Dtype
— ——— ————— ———
0 Temperatura ambiente 32 non-null float64
1 Temperatura Water IN 32 non-null float64
2 Temperatura Water OUT 32 non-null float64
3 Beta 32 non-null float64
4 Resa frigo 32 non-null float64
5 Potenza Assorbita Totale 32 non-null float64
6 Potenza Assorbita Ventilatori 32 non-null float64
7 EER 32 non-null float64
8 RMP ventilatori 32 non-null float64

```
# creo i set per il train dal dataset "dataset_simulatore_VBA.csv"

#creo le colonne di input del set di Training
X_train = dataset_simulatore_VBA.iloc[:,:-1]

#creo la colonna di target del set di Training
y_train = dataset_simulatore_VBA.iloc[:,-1]

# creo i set per il test dal dataset "dataset_piano_prove_TR3470.csv"

#creo le colonne di input del set di Testing
X_test = dataset_piano_prove_TR3470.iloc[:,:-1]

#creo la colonna di target del set di Testing
y_test = dataset_piano_prove_TR3470.iloc[:,-1]
```

3.3.2 Training e Test

In questo progetto di tesi si è scelto di applicare algoritmi di bagging e di boosting, che vengono utilizzati spesso per affrontare una varietà di problemi di apprendimento.
Ogni algoritmo è stato utilizzato nella sua forma base, con i suoi parametri di default.

3.3.2.1 Decision Tree

```
#importo le librerie
import pandas as pd
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import max_error
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

#creo il modello XGBRegressor
reg = DecisionTreeRegressor()

```

```
#fitting
reg.fit(X_train, y_train)

#training score
score = reg.score(X_train, y_train)
print("Training score: ", score)

#predict
y_pred = reg.predict(X_test)

#mean square error
mse = mean_squared_error(y_test, y_pred)
print("MSE: %.2f" % mse)

#root mean square error
print("RMSE: %.2f" % (mse**(1/2.0)))

#R2 score
r2score = r2_score(y_test,y_pred)
print("R2_score: %.3f" % r2score)

#max error
maxerror = max_error(y_test, y_pred)
print("max_error: %.2f" % maxerror)

#plot dei risultati
x_ax = range(len(y_test))
plt.plot(x_ax, y_test, label="real data")
plt.plot(x_ax, y_pred, label="DecisionTreeRegressor predicted")
plt.title("real data VS DecisionTreeRegressor")
plt.legend()
plt.show()
```

Listing 3.2: DecisionTree

3.3.2.2 Random Forest

```
#importo le librerie
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import max_error
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

```

```
#creo il modello XGBRegressor
rand_reg = RandomForestRegressor()

#fitting
rand_reg.fit(X_train, y_train)

#training score
score = rand_reg.score(X_train, y_train)
print("Training score: ", score)

#predict
y_pred = rand_reg.predict(X_test)

#mean square error
mse = mean_squared_error(y_test, y_pred)
print("MSE: %.2f" % mse)

#root mean square error
print("RMSE: %.2f" % (mse**(1/2.0)))

#R2 score
r2score = r2_score(y_test,y_pred)
print("R2_score: %.3f" % r2score)

#max error
maxerror = max_error(y_test, y_pred)
print("max_error: %.2f" % maxerror)

#plot dei risultati
x_ax = range(len(y_test))
plt.plot(x_ax, y_test, label="real data")
plt.plot(x_ax, y_pred, label="RandomForestRegressor predicted")
plt.title("real data VS RandomForestRegressor")
plt.legend()
plt.show()
```

Listing 3.3: andomForest

3.3.2.3 AdaBoost

```
#importo le librerie
import pandas as pd
from sklearn.ensemble import AdaBoostRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import max_error
```

```
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

#creo il modello XGBRegressor
abr = AdaBoostRegressor()

#fitting
abr.fit(X_train, y_train)

#training score
score = abr.score(X_train, y_train)
print("Training score: ", score)

#predict
y_pred = abr.predict(X_test)

#mean square error
mse = mean_squared_error(y_test, y_pred)
print("MSE: %.2f" % mse)

#root mean square error
print("RMSE: %.2f" % (mse**(1/2.0)))

#R2 score
r2score = r2_score(y_test,y_pred)
print("R2_score: %.3f" % r2score)

#max error
maxerror = max_error(y_test, y_pred)
print("max_error: %.2f" % maxerror)

#plot dei risultati
x_ax = range(len(y_test))
plt.plot(x_ax, y_test, label="real data")
plt.plot(x_ax, y_pred, label="AdaBoostRegressor predicted")
plt.title("real data VS AdaBoostRegressor")
plt.legend()
plt.show()
```

Listing 3.4: AdaBoost

3.3.2.4 XGBoost

```
#importo le librerie
import xgboost as xgb
```

```
import pandas as pd
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.metrics import max_error
import matplotlib.pyplot as plt

#creo il modello XGBRegressor
xgbr = xgb.XGBRegressor()

#fitting
xgbr.fit(X_train, y_train)

#training score
score = xgbr.score(X_train, y_train)
print("Training score: ", score)

#predict
y_pred = xgbr.predict(X_test)
print(y_pred)

#mean square error
mse = mean_squared_error(y_test, y_pred)
print("MSE: %.2f" % mse)

#root mean square error
print("RMSE: %.2f" % (mse**(1/2.0)))

#R2 score
r2score = r2_score(y_test,y_pred)
print("R2_score: %.3f" % r2score)

#max error
maxerror = max_error(y_test, y_pred)
print("max_error: %.2f" % maxerror)

#plot dei risultati
x_ax = range(len(y_test))
plt.plot(x_ax, y_test, label="real data")
plt.plot(x_ax, y_pred, label="XGBoost predicted")
plt.title("real data VS XGBoost")
plt.legend()
plt.show()
```

Listing 3.5: XGBoost

3.4 Risultati Ottenuti

In questo paragrafo verranno mostrati i risultati ottenuti con i quattro modelli applicati per la previsione del numero di rotazioni al minuto dei ventilatori con regolazione elettronica, in modalità *HP Flottante* attiva.

3.4.1 Accuratezza

Nei seguenti grafici e tabelle, sarà possibile valutare l'accuratezza dei modelli utilizzati. Si farà riferimento al valore di RMSE (root mean square error), al coefficiente di determinazione R^2 ed al valore massimo del residuo.

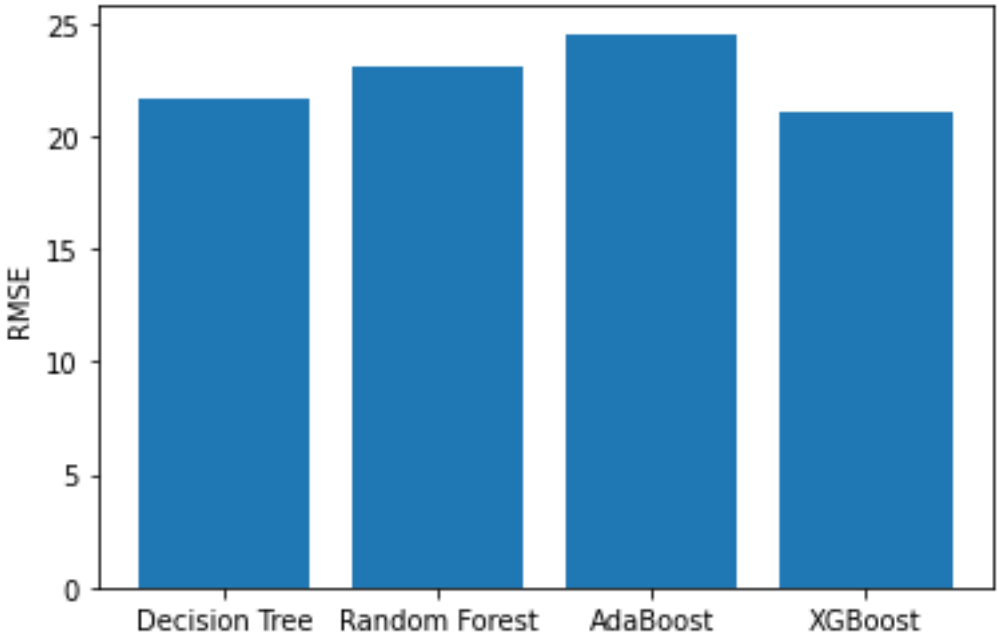


Figura 3.4: RMSE modelli

Models	RMSE	R^2 %	max error
Decision Tree	21.65	97.00	35.00
Random Forest	23.15	95.00	97.80
AdaBoost	24.56	96.10	64.34
XGBoost	21.14	97.10	40.51

Nei grafici 3.4, 3.5 e 3.6, viene riportata l'accuratezza dei modelli espressa in termini di RMSE, di coefficiente di determinazione R^2 e di massimo valore del residuo. Nella tabella i valori delle tre metriche analizzate per ogni modello.

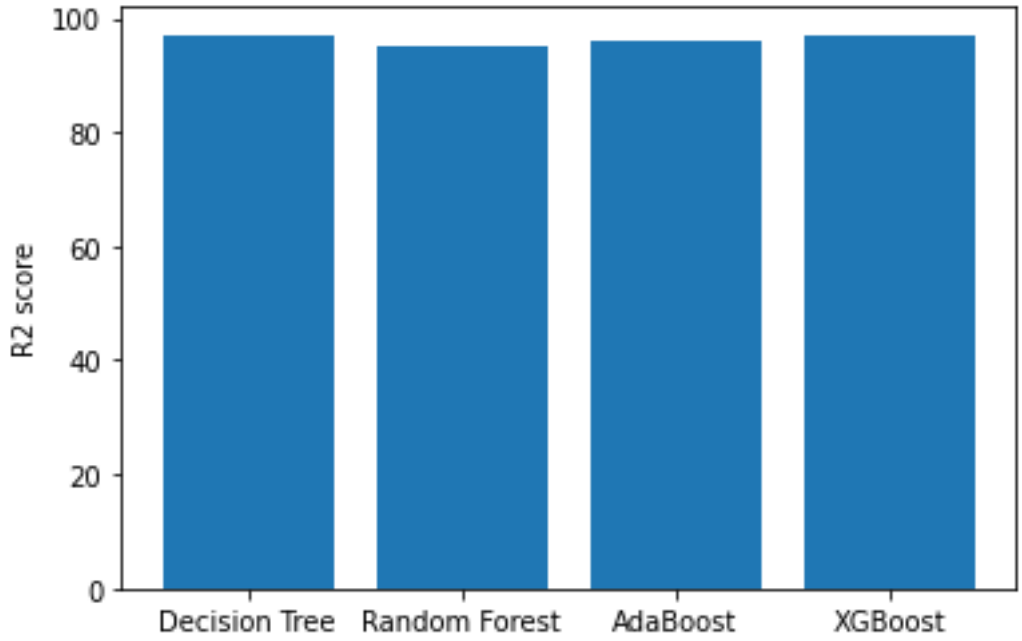


Figura 3.5: Coefficiente di determinazione R^2

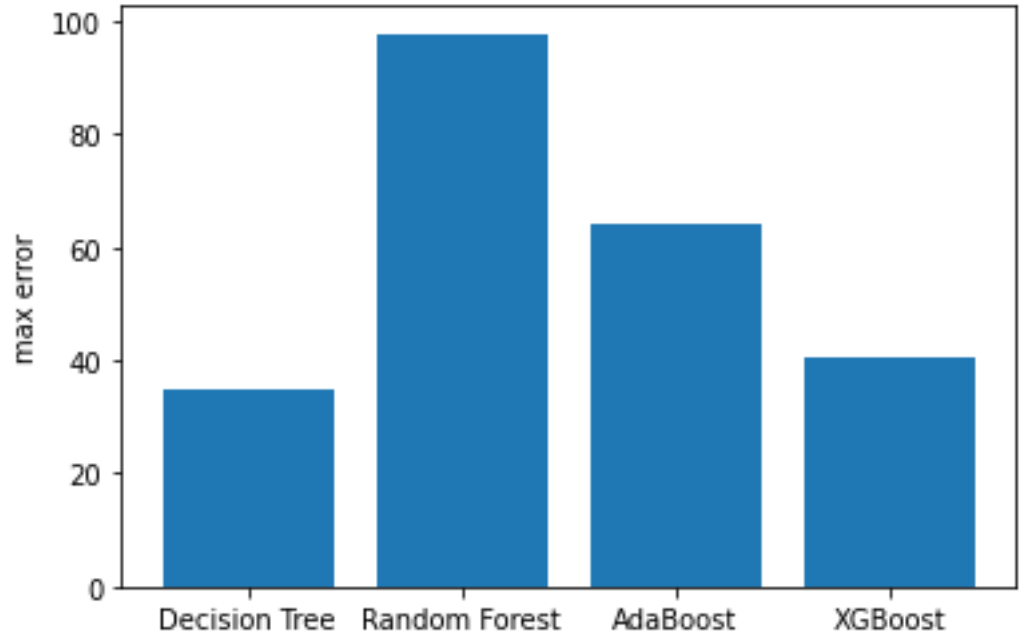


Figura 3.6: Massimo valore del residuo

L'RMSE è un indice di errore, ma non necessariamente un modello che riporta un RMSE basso è in grado di fornire previsioni più attendibili. Per una visione più completa e dettagliata è necessario effettuare un confronto sulla base del coefficiente di determinazione R^2. In generale su un dataset di pochi dati reali significativi, il valore massimo del residuo può essere considerato di supporto, ma

non di riferimento per l'analisi.

Le performance dei quattro modelli sono molto simili, ed essendo riferite ad una base dati di soli 32 campioni, ottenute dai collaudi, non è possibile apprezzare i vantaggi di un modello rispetto agli atri.

I valori dei coefficienti di determinazione R^2 sono buoni, tutti sopra al 95%, i modelli si sono adattati bene ai dati, e possono quindi garantire delle buone performance per quanto riguarda l'attendibilità delle previsioni.

Per quando riguarda il massimo valore dei residui, si passa dalla migliori prestazione di *Decision Tree* con un valore di circa 35 giri al minuto massimo di errore, a quella più scarsa di *Random Forest* con un valore di circa 98 giri al minuto massimo di errore.

Nei prossimi grafici si vedrà il confronto fra il dato reale (rotazioni al minuto dei ventilatori EC) di ogni singola prova e il risultato predetto dal modello in analisi.

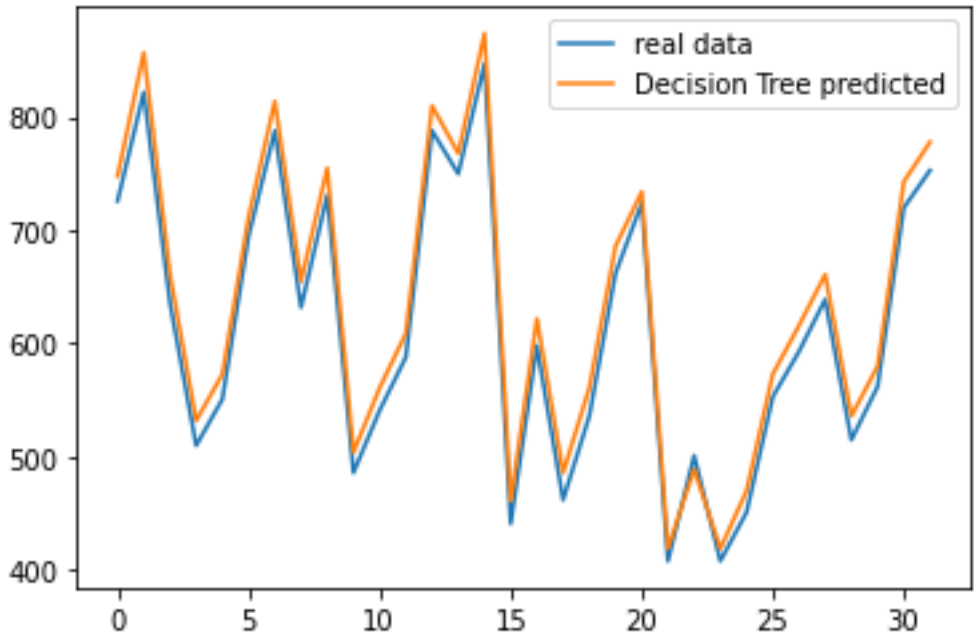


Figura 3.7: Real data VS Decision Tree

Come si può vedere, in generale ogni modello si comporta molto bene su quasi tutti i punti di lavoro, con predizioni che in alcuni casi sono corrispondenti al valore reale.

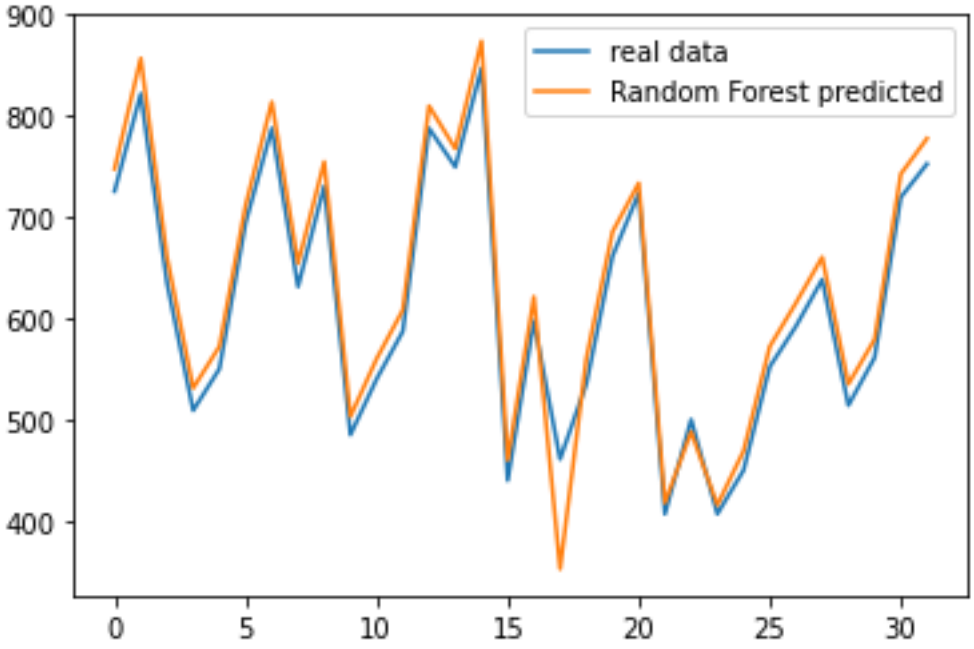


Figura 3.8: Real data VS Random Forest

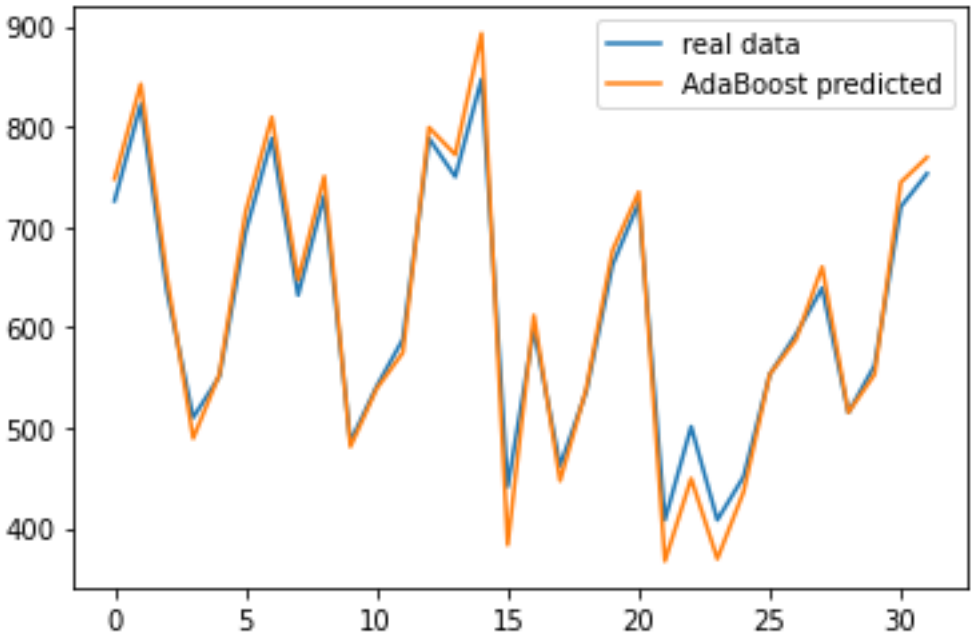


Figura 3.9: Real data VS AdaBoost

Nell'ultimo grafico che propongo, il confronto fra i dati reali ottenuti dai test del refrigeratore AST 2 140 con HP Flottante abilitata, contro i risultati dati dal simulatore reale della gamma AST 2. Come detto infatti nel paragrafo 2.2.4, il dataset per il training *dataset_ simulatore_ VBA* è stato strutturato per simulare l'opzione HP Flottante, con MTD costante, partendo dal simulatore reale, che

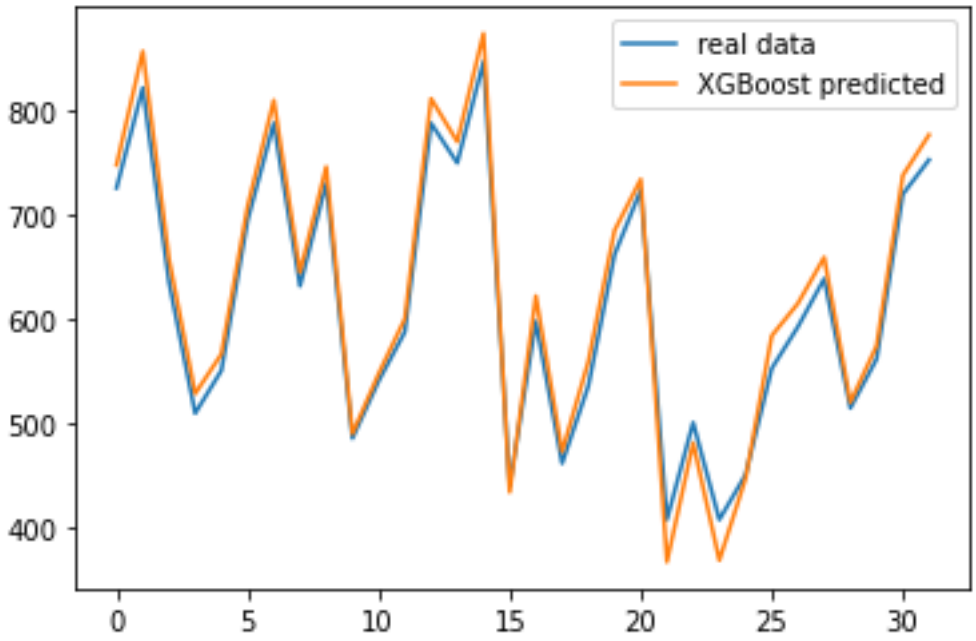


Figura 3.10: Real data VS XGBoost

invece non prevede HP flottante e simula con MTD variabile.

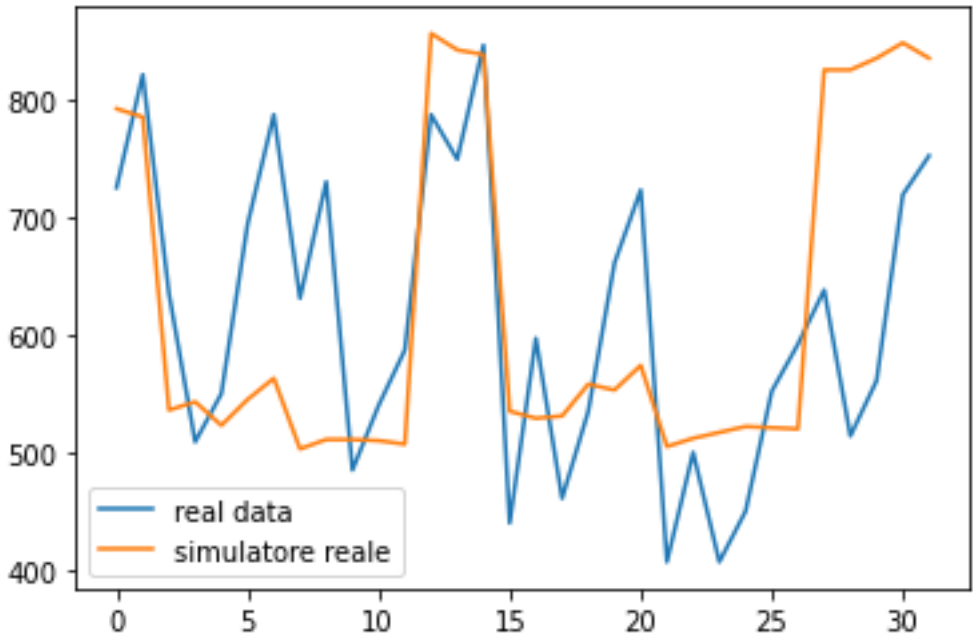


Figura 3.11: Real data VS simulatore reale

Il valore di *RMSE* è 124.00, cinque volte più alto di quello dato dai modelli di *machine learning*.

Come si può vedere il confronto fra dati reali con ventilatori che lavorano in HP Flottante ad MTD costante e dati simulati con simulatore reale dove i ventilatori lavorano con MTD variabile, mette bene in evidenza la differenza fra i vari modelli matematici che simulano le curve di lavoro dei ventilatori.

Capitolo 4

Conclusioni

Il progetto svolto durante il tirocinio in MTA S.p.a., aveva l'obiettivo di creare un simulatore alternativo a quello attualmente disponibile, scitto in linguaggio VBA, valutando le performance degli algoritmi di *bagging* e *boosting* proposti in questa analisi.

Si sono visti quattro modelli: *Decision Tree, Random Forest, AdaBoost* e *XGBoost*, che sono di natura diversa, ma in generale hanno riportato gli stessi risultati.

Decision Tree utilizza tecniche di *discesa ad albero*. *Random Forest* è un *ensemble* di decision tree. *AdaBoost* appartiene alla famiglia degli algoritmi di *gradient boosting*, che sfruttano la *discesa del gradiente* per ridurre al minimo l'errore sulle previsioni. Infine *XGBoost* il più completo dei quattro. E' un *metodo di ensemble* che si fonda sull'unione delle tecniche di *bagging* e *boosting*.

Per *migliorare* le prestazioni dei quattro algoritmi è stata fatta una fase iniziale di pre-processing, per ridurre il numero di *pattern* del dataset di addestramento.

La scelta iniziale, vista la *disponibilità* di un dataset di *dati reali* è stata quella di **non** procedere con l'*ottimizzazione* dei parametri dei modelli di machine learning, ma di eseguire il *training* di ognuno di essi con le loro impostazioni di *default*.

E' stato fatto infine, il test dei modelli sul dataset ottenuto dai collaudi del refrigeratore.

In conclusione, dopo le analisi, si può dire che i quattro modelli hanno fornito dei **buoni** risultati.
La loro *accuratezza* espressa in termini di *RMSE* si equivale, con un valore minino ottenuto da *XGBoost* - *RMSE* = 21.14, e un valore massimo ottenuto da *AdaBoost* - *RMSE* = 24.56.

Considerando che i ventilatori arrivano ad un massimo di 950 giri al minuto, e che nella realtà si può considerare ininfluente uno scostamento di ± 10% sul valore dei giri al minuto misurati, una performance dei modelli di ML con un valore massimo di RMSE pari a 25 è molto buona.

Tutti gli algoritmi testati raggiungono coefficienti di determinazione R^2 superiori al 95%, e possono essere utilizzati per effettuare previsioni attendibili.

Osservando i grafici 3.7, 3.8, 3.9, 3.10, dove sono riportati i confronti fra la predizione fatta da ogni modello di *ML*, e il dato reale, in ognuno dei 32 campioni disponibili, si può vedere che *Decision Tree* è quello che *replica* più fedelmente di tutti, la "spezzata" che congiunge i 32 valori dati dalle prove di collaudo.
Infine, nel grafico 3.11, è riportato il confronto fra i valori restituiti dal *simulatore reale MTA*, non parametrizzato alle condizioni di lavoro in *HP Flottante*, e i 32 punti dati dalle prove. Il grafico mostra risultati *scadenti* con un valore di *RMSE* pari a 124.00.

Da quanto detto poco fa si possono trarre le prime conclusioni. Partendo dal modello matematico che viene caricato come programma nel controllo del refrigeratore, che si occupa della gestione dell'*HP Flottante*, si è modificato il software originale del simulatore MTA, scritto in linguaggio *VBA*.
Sostanzialmente cambiandone i parametri secondo le logiche previste nel controllo, in modo da impostare il valore dell'*MTD* a *costante*, si è ottenuto un nuovo simulatore che ha restituito un dataset adattato per la fase di *training* dei modelli.
I quattro modelli in analisi hanno dato buone performance sui dati reali, dopo essersi *addestrati* sul dataset dato dal simulatore modificato.

Questo potrebbe portare a risvolti differenti:

1. Valutare una modifica agli *script VBA* dell'attuale simulatore, inserendo una nuova *curva di lavoro dei ventilatori*, fra le scelte disponibili, aggiun-

gendola alle due curve già disponibili *SEER* e *SEPR*, che può essere selezionata solo se si settano i ventilatori di tipo *EC* e fra i parametri in ingresso e obbligatorio impostare un valore costante dell'*MTD*. L'algoritmo sfrutterebbe come base di partenza gli script fatti per le curve *SEER*, *SEPR*, ma dovrebbe in qualche modo essere modificato per prendere in ingresso un valore di *MTD costante*.
In sintesi un nuovo script per simulare *HP Flottante* e una nuova variabile fra gli *input* del simulatore, l'*MTD*.

2. La seconda strada, che è anche quella percorsa da questo elaborato di tesi, è quella di poter modificare la *logica del controllo* in modo che l'opzione *HP Flottante* se abilitata, abbia un controllo di tipo **data driven**. Questo si potrebbe ottenere caricando nella *RAM* del controllo refrigeratore, un dataset con migliaia di record di simulazioni "predette" dai modelli di *ML*. Periodicamente attraverso nuove acquisizioni, ottenute magari dai report inviati ad MTA dai clienti, o da nuovi record validati in azienda, si potrebbero *ri-addestrare* i modelli di *ML* o addestrarne di nuovi, per migliorare la precisione delle simulazioni del dataset caricato nella RAM del controllo.

3. Si potrebbero valutare anche installazioni di impianti, dove il controllo ha caricato nella sua memoria sia un dataset di partenza, sia il modello di *machine learning*. Quest'ultimo attraverso mini addestramenti periodici in *batch*, migliora continuamente in base all'*esperienza*.

Bisogna precisare che, all'inizio di questa analisi, sono state fatte delle scelte tecniche per raggiungere un buon *trade-off* fra la complessità del simulare l'intera termodinamica di un refrigeratore e il numero di *pattern* del dataset ottenuto.

Inoltre è stato simulato un solo modello della gamma AST 2, e non tutti i modelli.

Concludo quindi dicendo che saranno fatti ulteriori esperimenti, per migliorare gli attuali modelli di *machine learning* o valutarne di nuovi, e dall'altro lato si dovrà aumentare la *complessità* del sistema che si vuole *astrarre*, cercando di eliminare tutte quelle scelte *restrittive* fatte in prima analisi.

Bibliografia

[1] Stuart Russell, Peter Norvig, *Artificial Intelligence. A Modern Approach*, Fourth Edition, 2021. Pearson.

[2] Anna Cretì, Nathaly Cruz, *Certificats d'Économie d'Énergie: l'esperienza francese*, 2018. Université Paris Dauphine e Université Paris Nanterre. https://rienergia.staffettaonline.com/articolo/33027/Certificats+d%E2%80%99%C3%89conomie+d%E2%80%99%C3%89nergie:+l%E2%80%99esperienza+francese/Anna+Cret%C3%AC+e+Nathaly+Cruz

[3] Loris Nanni, *Corso di Intelligienza Artificiale*, 2019. DEI, Università di Padova.

[4] MTA Spa, *Corso di Refrigerazione. Tecnica del Freddo*, 2012. MTA Spa.

[5] Christophe Bolein, *Économie d'énergie dans les centrales frigorifiques: La haute pression flottante - white paper*, 2010. Schneider Electric. https://issuu.com/schneider-electric-france/docs/economies-energies-centrales-frigorifiques

[6] Joel Grus, *Data Science from Scratch: First Principles With Python*, 2019. O°REILLY. https://github.com/joelgrus/data-science-from-scratch

[7] Luciano Ramalho, *Fluent Python. Clear, Concise and Effective Programming*, 2nd Edition, 2022. O°REILLY. https://github.com/fluentpython/example-code-2e

[8] Stefanie Molin, Ken Jee, *Hands-On Data Analysis with Pandas: A Python data science handbook for data collection, wrangling, analysis, and visualization*, 2nd Edition, 2021. Packt Publishing. https://github.com/stefmolin/Hands-On-Data-Analysis-with-Pandas-2nd-edition

[9] Corey Wade, Kevin Glynn, *Hands-on Gradient Boosting with XGBoost and Scikit-Learn. Perform accessible machine learning and extreme gradient boosting with python*, 2020. Packt Publishing. `https://github.com/PacktPublishing/Hands-On-Gradient-Boosting-with-XGBoost-and-Scikit-learn`

[10] Aurelién Géron, *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow. Concept, Tools and Techniques to Build Intelligent Systems*, 2nd Edition, 2019. O°REILLY. `https://github.com/ageron/handson-ml2`

[11] Cyrille Rossant, *IPython Interactive Computing and Visualization Cookbook. Over 100 hands-on recipes to sharpen your skills in high-performance numerical computing and data science in the Jupyter Notebook*, Second Edition, 2018. Packt Publishing. `https://cyrille.rossant.net/`

[12] Sebastian Raschka, Vahid Mirjalili, *Machine Learning con Python. Costruire algoritmi per generare conoscenza*, 2nd Edition, 2020. APOGEO. `https://github.com/rasbt/machine-learning-book`

[13] Andrea Loreggia, *Linear Regression in Python*, 2019. DEI, Università di Padova.

[14] Andrea Loreggia, *Machine Learning*, 2019. DEI, Università di Padova.

[15] Felix Zumstein, *Python for Excel: A Modern Environment for Automation and Data Analysis*, 2021. O°REILLY. `https://github.com/fzumstein/python-for-excel`, `https://www.xlwings.org/team`

[16] Peter Bruce, Andrew Bruce, Peter Gedeck, *Pratical Statistics for Data Scientists. 50+ Essential Concepts Using R and Python*, 2nd Edition, 2020. O°REILLY. `https://github.com/gedeck/practical-statistics-for-data-scientists`