



Progetto Alstom

Gruppo 2 :

- **Alesi Nicol**
- **Borino Marco**
- **De Pietri Ludovica**
- **Lapi Klima**
- **Papalia Michele**
- **Sansossio Matteo**



Programma



Introduzione

Approccio al problema

Risoluzione proposta

- Script
- Tempi e costi

Possibili miglioramenti

- GPT 4/5 vs Ollama
- Database relazionale
- Interfaccia utente

Fattibilità



Introduzione alla challenge

**Automatizzare l'analisi dei guasti
avvenuti su un treno per capirne
le cause al fine di permettere al
cliente di gestire la situazione
presente e futura**



Approccio al problema

Input testuale, Input generico diagnostica

Input testuale, Input
diagnostica generico

Analisi input

Analisi dell'input testuale
con modelli di NLP, per
identificare parole chiave
legate ai guasti

"WC non funzionante"



- Estrazione delle **parole chiave** dal plain text
- Mappatura delle parole chiave alle **colonne** della diagnostica generale

Verifica file generico diagnostica

Verifica nel file di diagnostica
generale la presenza di
eventuali guasti, con focus
sulla colonna rilevata prima

"TOI1:WC Fuori servizio"=1



- Individuazione della **colonna** binaria **corrispondente**
- Se il **valore = 1, conferma il guasto** e passa al file dettagliato dell'impianto

Caricamento dettaglio diagnostica

Caricamento della diagnostica
specificata dell'impianto
coinvolto, così da analizzare i
parametri specifici per trovare
la causa esatta



- Identificazione delle **problematiche** nei dati associati all'impianto



- Produzione di un **output testuale** chiaro e leggibile per spiegare il problema sfruttando gli **LLM**

Risultato

Output strutturato in
linguaggio naturale basato
sui dati rilevati

*livello dell'acqua è
inferiore al 10 percento,
controllare il riempimento
del serbatoio TOI1*

Risoluzione proposta

Linguaggio : Python

Librerie principali:

Spacy

Elabora il linguaggio naturale (NLP), sfruttando modelli pre-addestrati (it_core_news_lg).
Nel nostro caso analizza la similarità semantica.

Pandas

Analisi dati strutturati, necessaria per manipolare e leggere filexlsx o csv.

tkinter

Permette la **creazione di interfacce** grafiche utente :

- Finestre
- Bottoni
- Etichette

Ollama

Consente di **eseguire modelli di intelligenza artificiale localmente**, senza dover dipendere dalle API cloud di terze parti.

Script

analisiTesto.py

- **Input:**
 - file di testo
 - file excel di diagnostica generico
- **Connessione** tra testo inserito e colonna d'errore
- **Verifica** il flag d'errore nella colonna
- Se presente un errore viene **eseguito lo script** analisiDettaglioRaffinata.py

analisiDettaglio Raffinata.py

- **Analizza il file excel** di dettaglio (ora fisso)
- **Genera un report** attraverso il modello mistral (LLM) sulle indicazioni di un prompt
- **Ricerca le colonne** dei contatori e **stampa** il valore min e max
- **Salva il report** in un file .txt e lo mostra a schermo

Script - Esempio I/O

Questo PC > Volume (D:) > Alstom > docs

Report e Contatori

Report di Stato degli Impianti

Data e Ora: [Data e ora tratta dal file]

Durante la giornata, si è verificato il seguente stato degli impianti:

1. Inizialmente, l'impianto WC era in servizio ed era libero. L'impianto di scarico dei rifiuti (sistema scarico tazza) non ha presentato problemi all'inizio.
2. Successivamente, si è verificata l'isolamento del WC, che è passato quindi in stato "fuori servizio". In contemporanea con questo evento, è avvenuto il reset dello stato del sistema scarico tazza.
3. Più avanti nella giornata, si è rilevato un'anomalia nel sistema scarico tazza per cui lo stato è passato a "occupato". Al contempo, la porta ha subito un'apparizione anomala nel sistema scarico tazze per cui lo stato è stato "reset".
4. Successivamente, si sono verificati molti avvicendamenti "e lo stato "in servizio" per il sistema scarico tazza e la porta. Sono state riscontrate anomalie nel sistema scarico tazze per cui lo stato è stato "reset".
5. Infine, si è verificata l'isolamento del WC che ha fatto tornare lo stato "fuori servizio". In contemporanea con questo evento, lo stato di scarico dei rifiuti è tornato a essere libero.

Conclusioni: Al termine della giornata, si ha una situazione complessa in esame. Il WC è stato in generale in uno stato "fuori servizio" e continuamente in uno stato anomalo, sebbene la porta abbia superato i problemi riscontrati.

TOI:Contatore Attivazioni Sciacquone -> Min: 6415.0, Max: 6485.0
TOI:Contatore Attivazioni Rubinetto -> Min: 3932.0, Max: 3972.0
TOI:Contatore Operazioni Lavandino -> Min: 4390.0, Max: 4408.0
TOI:Contatore Attivazioni Asciugami -> Min: 1166.0, Max: 1171.0
TOI:Contatore Operazioni Pompa -> Min: 6243.0, Max: 6272.0
TOI:Contatore Porta Chiusa e Bloccata -> Min: 3768.0, Max: 3810.0
TOI2:Contatore Attivazioni Sciacquone -> Min: 10603.0, Max: 10603.0
TOI2:Contatore Operazioni Lavandino -> Min: 8541.0, Max: 8543.0
TOI2:Contatore Operazioni Pompa -> Min: 5826.0, Max: 5932.0
TOI2:Contatore Porta Chiusa e Bloccata -> Min: 6127.0, Max: 6127.0

Inserisci il testo: .Treno A viaggiava in accoppiata con Treno B

Esegui Analisi

Analisi Dati WC Fuori Servizio

Inserisci il testo: Non mi interessa niente

Esegui Analisi

OK

OK

Errore

Nessuna corrispondenza trovata nel testo.

OK

Tempi e costi

Ollama fornisce diversi modelli preaddestrati di LLM, verticalizzabili, ma con caratteristiche di base diverse.

Mistral:

Minimo 8gb RAM

Deepserk:

Minimo 31gb RAM

Ambiente di test:

- 16GB RAM
- i5-12500H
- Windows 11
- NVIDIA GeForce RTX 4050 Laptop
- Ollama 0.6.2

Output:

38.936 tokens/s

Caso d'uso:

622 righe non nulle → 15 token per frase

$$\text{Tempo} = \frac{9330}{38,936} \sim 239,62 \text{ s}$$

token

token/s

Possibili miglioramenti

Database relazionale

Possibili miglioramenti applicati al database, per avere una **collezione e gestione di dati più efficiente.**

Ollama vs ChatGPT 4/5

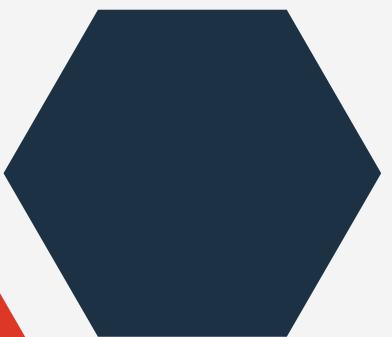
Confronto tra il metodo impiegato nel codice proposto ed **I.A. maggiormente addestrate.**

Interfaccia utente

Miglioramento proposto ai fini di ottenere un'**interfaccia più user friendly** ed esteticamente migliore.

Database relazionale

Linguaggio SQL e Python



Analisi dei passaggi:

Tipicamente i file vengono gestiti seguendo i passaggi di:

- Input testuale e Input diagnostica
- Analisi input
- Verifica file diagnostica
- Caricamento diagnostica
- Risultato

Creazione di filtri appositi in SQL

Che cos'è un filtro in un Database?

Un filtro programmato in SQL **è un tool appositamente creato dai gestori e ideatori del Database per filtrare, raggruppare e gestire i dati contenuti in esso**, in qualsiasi loro forma o dimensione.

Ad oggi i filtri in ambito della gestione dati sono ancora una delle risorse più preziose che i programmati di strutture logiche relazionali utilizzano.

Database relazionale

Linguaggio SQL e Python

Possibili impieghi:

In un'azienda in cui vi sono ingenti quantità di dati giornalieri come Alstom questi tool permetterebbero di **snellire** quelle che sono alcune **procedure nell'analisi e gestione dei dati**. Alcune pratiche applicazioni sono:

Creazione di dataset “puliti”

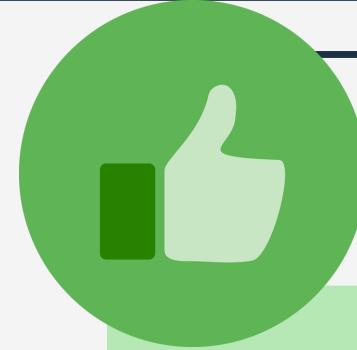
Gestione avanzata del Database e dei dati

Efficientamento nella ricerca dei dati

Separazione delle tipologie di dati

Queste quattro applicazioni **efficienterebbero in modo consistente quelli che sono i tempi di output** e di addestramento anche dell'Intelligenza Artificiale prevista per il progetto (previa integrazione dell'Algoritmo/Intelligenza con il DB con apposita libreria come psycopg 2 nel caso d'uso di SQL).

Pro e contro per l'utilizzo dei filtri nelle strutture logiche



PRO

Migliora le performance: Si riducono i dati a quelli realmente necessari → meno memoria e meno tempo di elaborazione



CONTRO

Rischio di perdita di informazioni utili: Filtri troppo restrittivi → potresti escludere dati importanti senza accorgertene.

Maggiore chiarezza del dato: Filtri ben scritti rendono il risultato più leggibile, pulito e coerente con l'obiettivo dell'analisi.

Complessità nella manutenzione: Se i filtri sono molti o complicati, aggiornare o modificare il codice può diventare difficile e rischioso.

Automazione facile: Una volta definiti i filtri, questi possono essere riutilizzati o schedulati senza intervento manuale.

Errori logici o sintattici: Un errore nei filtri (es. condizioni sbagliate) può portare a risultati completamente errati senza segnali evidenti.

Esempi pratici di filtri

```
SELECT *  
FROM clienti  
WHERE città = 'Milano';
```

E' importante sottolineare che **i filtri** programmati in SQL, possono essere **arricchiti da una serie di correlatori logici** che li possono rendere ancora **più specifici**, tra i più importanti si sottolineano:

- And
- If
- Or
- Else

Dati esemplificativi

```
clienti = [  
    {"id": 1, "nome": "Luca", "città": "Milano", "età": 30},  
    {"id": 2, "nome": "Anna", "città": "Roma", "età": 25},  
    {"id": 3, "nome": "Marco", "città": "Milano", "età": 40},  
]
```

```
milano_clienti = [c for c in clienti if c["città"] == "Milano"]  
  
print(milano_clienti)
```

Confronto tra Ollama e GPT-4/5 per l'implementazione aziendale di un LLM



OLLAMA

CHATGPT4/5

Esecuzione

Locale: il modello gira direttamente **su PC o server aziendale**, senza necessità di connessione Internet.

Cloud: il modello è accessibile **via Internet**, su server OpenAI, attraverso API o interfacce web.

Privacy

Massimo controllo: i dati restano **sempre all'interno dell'azienda**, fondamentale per la sicurezza industriale.

I dati viaggiano verso **server esterni**, anche se cifrati.
Meno controllo sulla riservatezza.

Costi

Gratuito: non richiede costi di utilizzo, salvo quelli hardware (RAM, GPU). Ideale per soluzioni interne.

A pagamento: il costo è calcolato in base ai token elaborati. Può diventare oneroso su grandi volumi.

Velocità di risposta

Molto veloce in locale, senza latenza di rete. Prestazioni dipendono dall'**hardware disponibile**.

Molto veloce, ma **dipende da rete e server esterni**. Può subire **rallentamenti in caso di alto traffico**.

Addestrabilità

Alta flessibilità: è possibile effettuare **fine-tuning** o adattare il modello ai **dati specifici** dell'azienda.

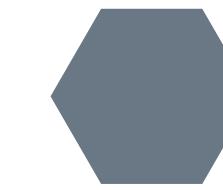
Limitato: non consente fine-tuning diretto, solo possibili **personalizzazioni tramite prompt o GPT personalizzati**.

Potenza del modello

Modelli leggeri (es. Mistral, LLaMA): ottimi per task specifici, **meno potenti per ragionamenti complessi**.

Modello estremamente potente: eccelle in compiti complessi, ragionamenti articolati e linguaggio naturale.

Fattibilità economica



COSTI HARDWARE	
PC di sviluppo (già presente)	0 €
Eventuale upgrade RAM a 32 GB	~120 €
SSD aggiuntivo (per spazio modelli/dati)	~80 €
Totale potenziale hardware	~200 €

COSTI SOFTWARE	
Librerie open-source	0 €
Ollama + modelli open (Mistral)	0 €
Eventuale GPU cloud	0-50 €/mese
Totale potenziale software	~0 €

Miglioramento con ChatGPT	
9330 token	\$ 0,01

Fattibilità organizzativa

Requisiti formativi minimi:

L'interfaccia user-friendly permette a personale non tecnico di utilizzare il sistema.

Integrazione nel processo:

Può essere integrato facilmente come tool di supporto alle decisioni tecniche.

Scalabilità:

Sì, con opportuna parametrizzazione degli input e gestione

Conformità privacy/dati:

Essendo in locale, il sistema è GDPR-compliant.

- **Tecnicamente realizzabile** su macchine con 16 GB di RAM.
- **Economicamente sostenibile** (costi bassi, no licenze, no cloud).
- **Organizzativamente utile** (impatto positivo sui tempi e chiarezza dei report)

Interfaccia grafica



UPLOAD DEL FILE EXCEL

Carica il file Excel che contiene l'elenco dei guasti del treno. Il sistema analizzerà i dati per aiutarti a trovare le cause e le soluzioni.

BARRA DI RICERCA

Scrivi una parola chiave, un codice guasto o una descrizione del problema per ottenere le possibili cause o soluzioni dai dati caricati.



Grazie per l'attenzione!