

UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE,
INGEGNERIA E MATEMATICA APPLICATA (DIEM)

CORSO DI LAUREA IN INGEGNERIA INFORMATICA



ELABORATO FINALE

UN SISTEMA INTERATTIVO PER LA PREDIZIONE DEL CONSUMO DI CARBURANTE IN FLOTTE AZIENDALI TRAMITE MACHINE LEARNING

Relatore

Prof. Antonio Parziale

Candidato

Alessio Napoli

Matr. 0612705164

Tirocinio svolto presso: Rebitral SRL

Tutor Aziendale: Dott. Elena Cuomo

Anno Accademico 2024 – 2025

INDICE

ABSTRACT.....	1
CAPITOLO 1 - DESCRIZIONE DEL PROBLEMA E FONDAMENTI TEORICI.....	2
1.1 – CONTESTO OPERATIVO.....	2
Perché prevedere i consumi?	3
1.2 - CONCETTI BASE DEL MACHINE LEARNING.....	4
Apprendimento supervisionato.....	5
Apprendimento non supervisionato.....	6
Apprendimento per rinforzo	6
Valutazione dei modelli.....	6
1.3 - PROBLEMATICHE COMUNI NEL MACHINE LEARNING.....	7
Overfitting e underfitting.....	7
Overfitting: quando il modello impara troppo.....	7
Underfitting: quando il modello impara troppo poco	8
Data Quality	9
CAPITOLO 2 – PROGETTAZIONE DEL SISTEMA.....	11
2.1 – ANALISI DEI REQUISITI.....	12
2.2 – PRE-PROCESSING DEI DATI.....	14
2.3 - CREAZIONE DEL MODELLO.....	16
Training e Valutazione dei modelli	16
Mean Squared Error	16
Coefficiente di determinazione R^2	17
Mean Absolute Error	17
Random Forest	18
XGBoost.....	18
Scelta degli iperparametri.....	19
2.4 – AMBIENTE DI SVILUPPO.....	20
2.5 – IMPLEMENTAZIONE DELLA DASHBOARD	21
CAPITOLO 3 – DASHBOARD INTERATTIVA	22
3.1 – PAGINA DI ACCESSO	22
3.2 – PAGINA DI SELEZIONE E VISUALIZZAZIONE DATI.....	23
3.3 – PAGINA DI ADDESTRAMENTO DEL MODELLO	25
3.4 – PAGINA DI VALUTAZIONE DEI MODELLI	26
CAPITOLO 4 – CONCLUSIONI E SVILUPPI FUTURI	28
Sviluppi Futuri.....	29
APPENDICE – MODULI DI PROGRAMMAZIONE SVILUPPATI	30
BIBLIOGRAFIA E SITOGRADIA	33
RINGRAZIAMENTI.....	35

ABSTRACT

DESCRIZIONE DEL PROBLEMA AFFRONTATO

Per le aziende di trasporto e logistica, la sostenibilità economica delle flotte di veicoli passa attraverso il controllo e l'ottimizzazione dei consumi di carburante, che rappresentano la principale fonte di costo operativo. In questo contesto, la capacità di prevedere con accuratezza i consumi riveste un ruolo strategico. L'uso di modelli predittivi consente di ridurre sprechi, migliorare l'efficienza e supportare strategie orientate alla sostenibilità ambientale.

INQUADRAMENTO DELL'ELABORATO NELLO SCENARIO TECNICO CONTEMPORANEO

La stima del consumo di carburante è al centro di numerosi studi e sperimentazioni recenti che sfruttano i dati acquistati dai sensori a bordo dei veicoli. Inoltre, l'utilizzo di modelli di machine learning rappresenta oggi l'approccio più efficace per affrontare questo tipo di problema, grazie alla capacità di apprendere dai dati e di adattarsi a contesti differenti.

CONTRIBUTO PERSONALE DEL CANDIDATO ALLA SOLUZIONE DEL PROBLEMA

Questo elaborato consiste nella progettazione e realizzazione di un sistema interattivo per la previsione del consumo di carburante, basato su dati telemetrici raccolti da una flotta aziendale. Il sistema integra tecniche di machine learning, con particolare attenzione all'utilizzo e al confronto degli algoritmi Random Forest e XGBoost, e include lo sviluppo di una dashboard in Python, tramite libreria Streamlit, per l'analisi e la valutazione dei modelli.

DESCRIZIONE DEI CONTENUTI APPLICATIVI/Sperimentali dell'elaborato

La dashboard si articola in più pagine. La prima pagina consente di estrarre dal database un insieme di dati e di visualizzare le distribuzioni delle feature. La seconda pagina consente la selezione e l'addestramento di algoritmi di apprendimento supervisionato. Infine, l'ultima pagina riassume le prestazioni dei modelli in termini di errore quadratico medio, errore assoluto medio e coefficiente di determinazione. Il funzionamento del sistema è stato testato su un ampio dataset di dati telemetrici. Le migliori prestazioni sono state ottenute con Random Forest (MSE pari a 0.0017, R^2 pari a 0.9362).

CAPITOLO 1 - DESCRIZIONE DEL PROBLEMA E FONDAMENTI TEORICI

1.1 – CONTESTO OPERATIVO

Il presente elaborato nasce dall'esperienza di tirocinio curricolare svolta presso l'azienda ***Rebitral Srl***, operante nel settore dello sviluppo software per il fleet management e per la sicurezza.

Il progetto in questione è la realizzazione di un sistema predittivo, basato su tecniche di machine learning, e volto a stimare lo stile di guida e il **consumo di carburante** di una flotta aziendale. I dati provengono da sensori installati sui veicoli e includono informazioni relative a posizione GPS, velocità, accelerazione, frenate e altri parametri di guida. Nello specifico, le attività di progettazione e sviluppo descritte in questo elaborato hanno riguardato la realizzazione di un sistema per la predizione del consumo di carburante da dati telemetrici. Il sistema, per predire il consumo di carburante, utilizza algoritmi di machine learning, tra cui **Random Forest** e **XGBoost**.

Il progetto ha previsto anche la realizzazione di una **dashboard interattiva**, sviluppata tramite la libreria **Streamlit** di Python, per consentire agli utenti di caricare dati, addestrare e valutare le prestazioni dei modelli, visualizzando i risultati in modo intuitivo e dinamico.

PERCHÉ PREVEDERE I CONSUMI?

Nella gestione di una flotta aziendale, uno degli aspetti più rilevanti è il monitoraggio e l'ottimizzazione dei consumi di carburante. In un'azienda di trasporto e/o logistica, il carburante è una delle principali fonti di spesa. Secondo numerosi studi da parte dell'International Energy Agency (IEA) [1] e dell'European Environment Agency (EEA) [2], la spesa per il rifornimento può rappresentare fino al 30–40% dei costi totali di gestione di un veicolo commerciale, e variazioni anche minime nell'efficienza di consumo possono tradursi in risparmi considerevoli su larga scala. Con un modello predittivo come quello realizzato nel corso del tirocinio curriculare e descritto in questo elaborato, è possibile rispondere a questa esigenza aziendale. Un modello del genere è in grado, infatti, di stimare il consumo di carburante mediante l'analisi di dati telemetrici, ricavati dai veicoli. Questo può tornare utile anche per determinare eventuali anomalie all'interno del veicolo, oltre ad ottenere diversi benefici, quali:

- **Riduzione dei costi:** poter stimare i consumi consente alle aziende di pianificare con maggiore accuratezza le spese e di identificare eventuali inefficienze operative. Ad esempio, come suggerito prima, veicoli con anomalie nei consumi possono essere rapidamente individuati e sottoposti a manutenzione preventiva.
- **Ottimizzazione della pianificazione logistica:** conoscere i consumi previsti per un determinato tragitto consente di scegliere i percorsi più efficienti, minimizzando i rifornimenti e riducendo i tempi di inattività.
- **Sostenibilità ambientale:** la riduzione dei consumi ha un impatto diretto sulle emissioni di anidride carbonica e altri gas serra. In un contesto in cui le normative europee e internazionali spingono verso una maggiore sostenibilità, disporre di sistemi in grado di prevedere e ridurre i consumi diventa un vantaggio competitivo.
- **Supporto alle decisioni manageriali:** i dati derivanti dai modelli predittivi possono essere integrati in dashboard aziendali, fornendo supporto ai manager per la gestione della flotta aziendale.

1.2 - CONCETTI BASE DEL MACHINE LEARNING

Il **Machine Learning (ML)** è una branca dell'intelligenza artificiale (IA) che si occupa dello sviluppo di algoritmi in grado di apprendere automaticamente dai dati e di migliorare le proprie prestazioni nel tempo senza la necessità di essere esplicitamente programmati per ogni singolo compito [3]. I principali componenti di un algoritmo di ML sono:

1. **Dati:** Essi costituiscono la base del processo di apprendimento. Possono essere strutturati (come tabelle) o non strutturati (come testo, immagini, audio, video). La qualità, quantità e rappresentatività dei dati influenzano profondamente le prestazioni del modello.
2. **Algoritmo di apprendimento:** L'algoritmo è il metodo matematico che consente al sistema di identificare schemi e regolarità nei dati. Esistono molteplici algoritmi, ciascuno con specifici punti di forza a seconda della natura del problema.
3. **Modello:** Una volta completata la fase di addestramento, il modello rappresenta il risultato dell'applicazione dell'algoritmo ai dati. Esso racchiude la "conoscenza" acquisita e viene utilizzato per effettuare previsioni o decisioni.
4. **Funzione di perdita:** La funzione di perdita quantifica lo scostamento tra le previsioni del modello e i valori reali. Durante l'addestramento, l'algoritmo cerca di minimizzare questa funzione per migliorare l'accuratezza.
5. **Ottimizzazione:** L'ottimizzazione è il processo matematico con cui si modificano i parametri del modello al fine di ridurre la funzione di perdita e migliorare progressivamente le prestazioni.

I sistemi di ML rientrano in una o più delle seguenti categorie in base a come imparano a fare previsioni o generare contenuti:

- Apprendimento supervisionato
- Apprendimento non supervisionato
- Apprendimento per rinforzo

APPRENDIMENTO SUPERVISIONATO

I modelli di apprendimento supervisionato sono in grado di effettuare previsioni dopo essere stati addestrati su un insieme di dati etichettati, ovvero contenenti gli input e le relative risposte corrette. Quindi, attraverso questa fase di addestramento, il modello apprende le relazioni tra le variabili di input e quelle di output.

Esistono due categorie di apprendimento supervisionato: la **classificazione** e la **regressione**.

Un modello di regressione ha l'obiettivo di stimare un valore numerico continuo a partire da un insieme di valori indipendenti, denominati *features*. Esempi pratici sono la previsione della quantità di pioggia in millimetri in base a parametri meteorologici (temperatura, umidità, pressione atmosferica), oppure, come nel caso del problema affrontato in questo elaborato, la previsione del consumo di carburante di un veicolo sulla base di velocità, accelerazioni e giri motore.

I modelli di classificazione, invece, hanno l'obiettivo di prevedere a quale categoria discreta appartenga un elemento, sempre descritto in termini di features. In questo caso, l'output non è un valore continuo ma una classe predefinita. Esempi pratici sono determinare se un'email è spam oppure non spam, oppure classificare uno stile di guida come prudente, normale o aggressivo.

I modelli di classificazione non si limitano a fornire la classe più probabile, ma spesso restituiscono una probabilità associata a ciascuna classe. Ad esempio, un sistema di riconoscimento di immagini potrebbe restituire, data una certa immagine: 70% gatto, 25% cane, 5% altro.

I modelli di classificazione sono suddivisi in due gruppi: classificazione binaria e classificazione multiclasse. Nel primo caso le classi possibili sono solo due, mentre nel secondo caso le classi attribuibili sono più di due.

APPRENDIMENTO NON SUPERVISIONATO

I modelli di apprendimento non supervisionato lavorano su dati privi di etichette o risposte corrette. L'obiettivo principale è individuare schemi e relazioni significative, così che il modello possa riconoscere autonomamente strutture e somiglianze all'interno dei dati. Un tipo di apprendimento non supervisionato è il **clustering**, nel quale si raggruppano per elementi simili i dati in modo da creare dei gruppi naturalmente delimitati, come mostrato in Figura 1.

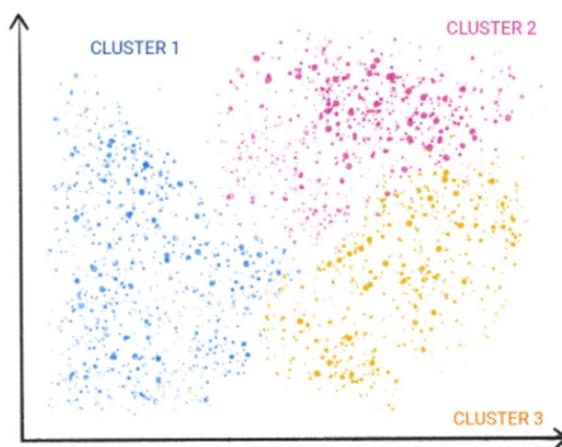


Figura 1: Un esempio di cluster

APPRENDIMENTO PER RINFORZO

I modelli di questo tipo fanno previsioni ricevendo ricompense o penalizzazioni in base alle azioni compiute. Il modello impara quindi a prendere decisioni interagendo con l'ambiente e aggiornando la propria strategia in funzione dei feedback ricevuti. Le ricompense e penalità variano in base al contesto applicativo: ad esempio, in robotica una ricompensa può derivare da riuscire a sollevare un oggetto, mentre una penalità può essere una caduta o una collisione.

VALUTAZIONE DEI MODELLI

Per misurare l'efficacia di un modello si utilizzano metriche diverse a seconda del tipo di problema:

- Regressione: MSE, RMSE, MAE, R^2 .
- Classificazione: Accuracy, Precision, Recall, F1-score, AUC-ROC.
- Clustering: Silhouette score, Davies-Bouldin index.

1.3 - PROBLEATICHE COMUNI NEL MACHINE LEARNING

OVERFITTING E UNDERFITTING

Uno degli aspetti di cui tenere conto nel processo di addestramento di un modello di Machine Learning riguarda la sua capacità di generalizzare, ossia produrre buone previsioni non solo sui dati utilizzati durante l'addestramento, ma anche su dati nuovi e mai osservati [4]. Due problematiche comuni che possono compromettere questa capacità sono l'**overfitting** e l'**underfitting**. Entrambe rappresentano situazioni in cui il modello non riesce a catturare in modo adeguato i pattern presenti nei dati.

OVERFITTING: QUANDO IL MODELLO IMPARA TROPPO

L'overfitting si verifica quando un modello apprende **troppi dettagli e rumore** dai dati di addestramento, arrivando a "memorizzare" anche informazioni irrilevanti o casuali. In questo caso, il modello mostra prestazioni eccellenti sui dati su cui è stato addestrato, ma risulta **molto meno accurato** su dati nuovi o mai visti prima [4]. È come se un alunno imparasse a memoria tutte le risposte di un compito senza comprendere realmente le regole generali che stanno alla base del problema oggetto della verifica: appena cambia la domanda, sbaglia.

I segnali tipici dell'overfitting sono:

- Errore molto basso sul training set;
- Errore alto sul validation o test set;
- Curve di apprendimento divergenti, con la perdita (loss) che continua a calare sul training set ma si stabilizza o peggiora sul validation set.

Le cause più comuni includono:

- Modelli troppo complessi, con troppi parametri rispetto alla quantità di dati disponibili;
- Addestramento troppo prolungato, che porta il modello a "memorizzare" i dati;

- Presenza di rumore nei dati, come valori anomali o informazioni non rappresentative.

Per contrastare l'overfitting si possono adottare varie strategie:

- Regolarizzazione: tecniche come L1 (lasso) e L2 (ridge) aggiungono un termine di penalità alla funzione di perdita per evitare che i pesi assumano valori troppo grandi;
- Early stopping: interrompe l'addestramento non appena le prestazioni sul validation set iniziano a peggiorare;
- Data augmentation: in particolare nel campo dell'elaborazione di immagini, consente di generare nuove istanze modificando leggermente quelle esistenti;
- Cross-validation: validazione incrociata per verificare la stabilità delle prestazioni del modello su diverse suddivisioni dei dati.

UNDERFITTING: QUANDO IL MODELLO IMPARA TROPPO POCO

All'estremo opposto si trova l'**underfitting**, che si verifica quando il modello è **tropo semplice** o non sufficientemente addestrato per riconoscere i pattern significativi presenti nei dati [4]. In questo caso, il modello ottiene prestazioni scarse sia sui dati di addestramento che su quelli di test, perché non riesce a catturare la complessità della relazione tra input e output.

Le cause principali dell'underfitting includono:

- Modelli con bassa capacità espressiva, come un modello lineare per dati che seguono una relazione non lineare;
- Training interrotto troppo presto, che non permette al modello di apprendere a sufficienza;
- Feature non adeguate o una rappresentazione dei dati troppo povera.

Alcuni segnali tipici dell'underfitting sono la presenza di un errore elevato sia sul training set che sul test set, oppure curve di apprendimento piatte, senza miglioramenti significativi nelle metriche di accuratezza.

Per correggere l'underfitting si possono seguire vari approcci:

- Utilizzare modelli **più complessi**, con maggiore capacità di apprendimento;
- Prolungare il tempo di addestramento;
- Migliorare la **rappresentazione dei dati** con feature engineering più accurato;
- Rimuovere tecniche di regolarizzazione troppo restrittive.

DATA QUALITY

La **qualità dei dati** è un elemento fondamentale per il successo di qualunque progetto di machine learning. Modelli predittivi e sistemi di analisi traggono infatti la loro efficacia non soltanto dalla scelta dell'algoritmo, ma soprattutto dalla completezza e affidabilità dei dati utilizzati per l'addestramento. Dati rumorosi, incompleti o non rappresentativi possono compromettere significativamente la capacità del modello di generalizzare e, di conseguenza, ridurre la validità delle previsioni ottenute.

Per essere qualitativamente ottimi, i dati devono essere:

- **Completi**: i dataset devono contenere un numero sufficiente di osservazioni e di variabili rilevanti per descrivere in maniera accurata il caso di studio.
- **Accurati**: i dati devono riflettere in maniera corretta la realtà osservata, riducendo al minimo eventuali errori di registrazione e/o anomalie.
- **Consistenti**: i valori devono essere coerenti tra le diverse fonti e nel tempo, evitando duplicazioni e contraddizioni.
- **Rappresentativi**: il campione deve includere tutte le categorie di interesse, così da permettere al modello di generalizzare correttamente anche su nuovi dati.

All'interno del data quality, un aspetto particolarmente rilevante è costituito dal **bias**. In ambito di machine learning, si riferisce a una distorsione sistematica che porta un modello a generare previsioni non corrette o non imparziali. Dunque, il bias si manifesta quando i risultati prodotti dall'algoritmo deviano dalla realtà a causa di fattori legati ai dati, alle scelte progettuali o al processo di addestramento.

La comunità scientifica ha sviluppato numerose tecniche per identificare e mitigare il bias nei sistemi di Machine Learning. Alcuni approcci rilevanti includono:

- **Controllo della rappresentatività:** assicurarsi che il dataset rifletta in modo equo la popolazione reale, attraverso la raccolta di dati eterogenei e bilanciati.
- **Bilanciamento dei dati:** applicare metodi di oversampling o undersampling per correggere gli sbilanciamenti nelle classi.
- **Adozione di metriche etiche:** oltre alla classica accuratezza, si considerano metriche come fairness, demographic parity, equal opportunity, che misurano l'equità del modello rispetto a determinate variabili sensibili.
- **Modelli interpretabili (XAI):** l'uso di tecniche di Explainable AI consente di visualizzare, tracciare e spiegare le decisioni del modello, rendendo più semplice il rilevamento di eventuali comportamenti problematici.

CAPITOLO 2 – PROGETTAZIONE DEL SISTEMA

La realizzazione del sistema predittivo ha seguito una pipeline ben strutturata, articolata in più fasi che vanno dalla raccolta dei dati grezzi fino alla visualizzazione dei risultati tramite una dashboard interattiva. La Figura 2 sintetizza il workflow di progetto ed evidenzia le fasi più significative dell’attività svolta: pre-processing dei dati, selezione delle feature, addestramento dei modelli, validazione dei risultati e creazione di una interfaccia utente per la gestione del sistema.

Nelle sezioni che seguono verrà analizzato nel dettaglio ciascuna fase, descrivendo il ruolo e le scelte implementative adottate durante lo sviluppo del progetto.

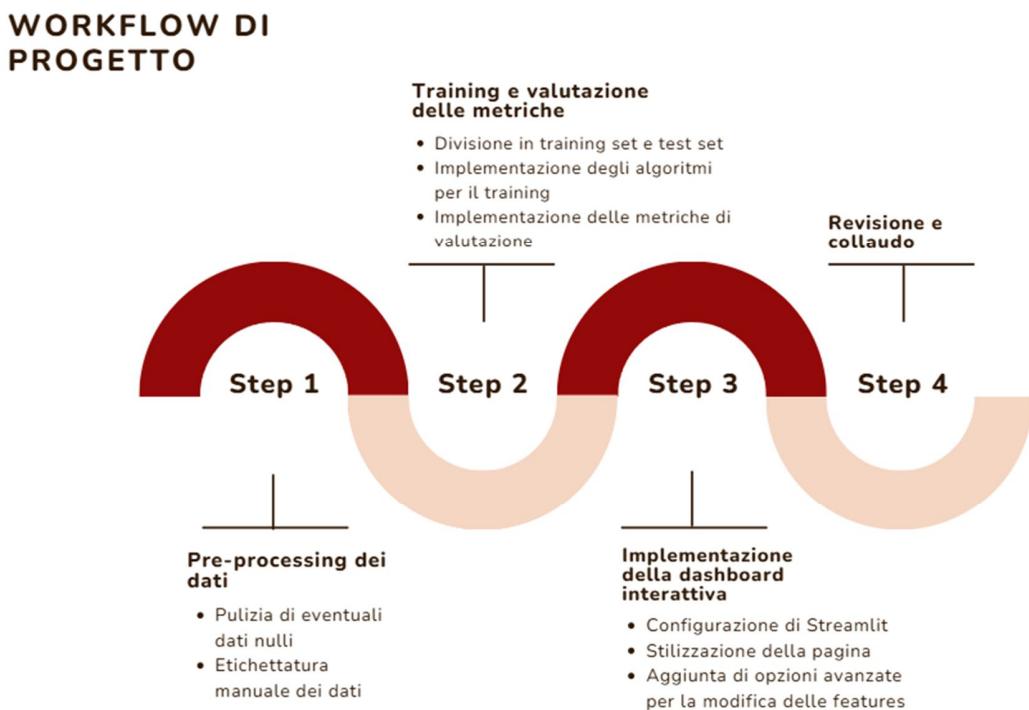


Figura 2: Workflow di progetto

2.1 – ANALISI DEI REQUISITI

L’analisi dei requisiti è importante per garantire che il sistema sviluppato risponda in maniera adeguata alle esigenze degli obiettivi progettuali designati dall’azienda.

I requisiti definiti dall’azienda per la realizzazione di un sistema dotato di dashboard per la predizione del consumo di carburante da dati telemetrici archiviati in un database sono riassunti di seguito in Tabella 1.

Requisito Funzionale	Descrizione
Database Relazionale	Il sistema deve essere dotato di un DB relazionale per il salvataggio dei dati telemetrici
Visualizzazione dei dati	Il sistema deve consentire la visualizzazione dei dati archiviati nel database e che verranno utilizzati per addestrare il modello di ML.
Pre-processing dei dati	Il sistema deve consentire il filtraggio, la pulizia e la normalizzazione dei dati estratti dal dataset aziendale.
Selezione e addestramento dei modelli	Il sistema deve consentire all’utente di selezionare uno o più algoritmi di ML, di settarne gli iperparametri e di avviare l’addestramento.
Valutazione dei modelli	Calcolare le varie metriche di valutazione, come MSE, MAE, R^2 .
Identificazione del modello migliore	Evidenziare in automatico l’algoritmo con le prestazioni più elevate.
Ambiente di sviluppo predefinito e librerie	Utilizzo di ambiente Python e librerie dedicate allo sviluppo ML.

Tabella 1: Tabella dei requisiti funzionali

La Figura 3 mostra uno **schema a blocchi** che sintetizza l'architettura del sistema con le principali componenti del processo e le loro interconnessioni logiche e funzionali.



Figura 3: Schema a blocchi del progetto svolto

2.2 – PRE-PROCESSING DEI DATI

Come evidenziato nel *Machine Learning Crash Course* di Google [3], un dataset con valori incoerenti con il resto dei dati presenti oppure nulli può portare a risultati fuorvianti e modelli poco generalizzabili, a prescindere dalla complessità o potenza dell'algoritmo impiegato.

Per quanto riguarda il progetto, il dataset iniziale era costituito da circa **700.000 righe**, in cui erano memorizzati i dati acquisiti dai sensori installati sui veicoli della flotta aziendale. In Tabella 2 sono riportati i dati acquisiti, con relativa descrizione.

L'operazione di pulizia e trasformazione del dataset è stata condotta mediante l'utilizzo di query SQL eseguite tramite codice Python. Questo approccio ha permesso di operare direttamente sulla struttura tabellare del database, automatizzando la rimozione dei dati inconsistenti. In particolare, sono stati eliminati:

- I record contenenti **valori null** o mancanti;
- I record contenenti **valori pari a zero** per attributi in cui tale valore risulta anomalo. Per esempio, un veicolo non può avere velocità = 130 km/h e RPM del motore = 0;
- Gli attributi ritenuti **non rilevanti** ai fini dell'analisi predittiva, in base a considerazioni sia statistiche sia di dominio.

Tipo di dato acquisito	Descrizione
Vehicle_speed	Indica la velocità a cui va il veicolo al momento della registrazione.
Can_fuel_consumed	Indica il consumo di carburante al momento della registrazione.
Position_latitude, Position_altitude, Position_longitude	Indica le coordinate del veicolo al momento della registrazione.
Position_direction	Indica la direzione (nord, sud, est, ovest) del veicolo al momento della registrazione.
Can_engine_status	Indica se il motore è acceso o spento al momento della registrazione
Can_engine_rpm	Indica il numero di giri del motore al momento della registrazione.
Vehicle_mileage	Indica il numero di chilometri percorsi al momento della registrazione.

Tabella 2: Tabella dei dati acquisiti dal veicolo

L'attributo *can_fuel_consumed*, che rappresenta il valore cumulativo del carburante consumato dal veicolo dall'accensione, è stato utilizzato per calcolare il consumo istantaneo del veicolo *fuel_delta* ed etichettare i dati che saranno poi utilizzati come training set per addestrare il modello di machine learning selezionato dall'utente del sistema. In particolare, per ottenere una stima del consumo istantaneo, è stata calcolata la differenza del valore di *can_fuel_consumed* tra due istanti consecutivi. Per ridurre l'impatto del rumore e delle fluttuazioni locali nei dati, si è applicata una media mobile esponenziale (EWM), con finestra parametrizzata.

Questa fase di pre-processing ha permesso di trasformare un insieme eterogeneo di dati grezzi in un dataset coerente, compatto e strutturato, pronto per essere utilizzato nelle successive fasi di addestramento dei modelli di machine learning.

2.3 - CREAZIONE DEL MODELLO

Il modello sviluppato si articola secondo una classica pipeline di apprendimento supervisionato, in cui il consumo istantaneo di carburante è trattato come **variabile dipendente continua**, da prevedere sulla base di un insieme di **variabili indipendenti** (feature) derivate dai sensori veicolari. Sono stati implementati due algoritmi: **Random Forest** e **XGBoost**, scelti per la loro efficacia nella gestione di dataset complessi, eterogenei e non perfettamente lineari.

TRAINING E VALUTAZIONE DEI MODELLI

Una volta che i dati di interesse sono stati selezionati ed estratti dal database, il sistema li suddivide in training set e test set per consentire l'addestramento e la validazione del modello di ML che verrà poi selezionato dall'utente. Il sistema ti permette inoltre di regolare la ripartizione di test set e training set in un range predefinito.

Inoltre, il sistema implementa una serie di metriche che consentono di valutare e confrontare i modelli selezionati dall'utente. In particolare, i modelli sono stati valutati tramite le metriche **Mean Squared Error (MSE)**, **R² (coefficiente di determinazione)** e **Mean Absolute Error (MAE)**.

Infine, il sistema sintetizza i risultati mediante grafici a barre, al fine di facilitare la comparazione dei vari modelli addestrati.

MEAN SQUARED ERROR

Il **Mean Squared Error (MSE)** è una delle metriche più utilizzate per valutare le prestazioni di un modello di regressione. [3] Misura la media dei quadrati degli errori, ossia delle differenze tra i valori osservati e valori predetti dal modello. La sua formula è:

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

Equazione 1: Formula generale per il calcolo dell'MSE

dove y_i rappresenta il valore reale, \hat{y}_i il valore stimato dal modello, e n il numero delle osservazioni.

Il vantaggio principale di MSE è la forte sensibilità agli errori grandi. Difatti, gli scarti maggiori vengono amplificati dal quadrato dell'equazione, fornendo un segnale forte sulla presenza di predizioni particolarmente imprecise. Tuttavia, proprio questa caratteristica può renderlo meno robusto in presenza di outlier. In generale, valori più bassi di MSE corrispondono a modelli più accurati.

COEFFICIENTE DI DETERMINAZIONE R^2

Il **coefficiente di determinazione** (R^2), è una metrica che quantifica la proporzione della varianza della variabile dipendente che è spiegata dal modello. [3] È definito come:

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

Equazione 2: Formula generale per il calcolo del Coefficiente di Determinazione

dove y_i sono i valori osservati, \hat{y}_i quelli predetti, e \bar{y} la media dei valori reali.

Un valore di R^2 vicino a 1 indica che il modello è in grado di spiegare quasi tutta la variabilità dei dati, mentre un valore prossimo a 0 segnala che il modello non è più informativo di una semplice media.

MEAN ABSOLUTE ERROR

Il **Mean Absolute Error (MAE)** misura l'errore medio assoluto tra i valori reali e le predizioni del modello, senza tenere conto del segno della differenza [3]. La formula è:

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

Equazione 3: Formula generale per il calcolo del MAE

dove y_i è il valore osservato, \hat{y}_i è il valore predetto e n è il numero di osservazioni.

RANDOM FOREST

Random Forest è un algoritmo di apprendimento supervisionato basato su un insieme di **alberi decisionali (decision trees)**, che operano in maniera aggregata secondo il principio del **bagging (bootstrap aggregating)** [5]. L'idea centrale è quella di costruire molteplici alberi su sottocampioni del dataset e di aggregare i risultati mediante una media nel caso di regressione, o votazione nel caso di classificazione.

Ogni albero viene costruito su un sottoinsieme casuale dei dati e delle feature, rendendo il modello meno sensibile al rumore e alle anomalie nei dati. Questo approccio consente di ottenere modelli robusti e generalmente poco affetti da overfitting, specialmente se il numero di alberi è sufficientemente elevato.

L'utilizzo di modelli come Random Forest migliora la generalizzazione del modello, combinando i punti di forza di modelli più semplici e riducendo la varianza. Nel presente progetto, è stato implementato attraverso la libreria scikit-learn, con una fase di tuning dei parametri iper-ottimizzata tramite tecniche di cross-validation.

XGBOOST

XGBoost è un algoritmo la cui tecnica consiste nell'addestrare sequenzialmente una serie di modelli, in cui ciascun nuovo modello cerca di correggere gli errori commessi dai modelli precedenti [6]. A differenza del bagging, che riduce la varianza aggregando modelli indipendenti, il boosting lavora per ridurre il bias attraverso una correzione iterativa degli errori.

Questo algoritmo è una delle implementazioni più efficienti e ottimizzate del **gradient boosting**, progettata per ottenere prestazioni elevate sia in termini predittivi che computazionali. La sua architettura prevede l'utilizzo di funzioni di costo regolarizzate, strategie di pruning per evitare overfitting, e ottimizzazioni come la gestione dei missing values e la parallelizzazione del training.

SCELTA DEGLI IPERPARAMETRI

Gli iperparametri, sono valori definiti manualmente prima dell’addestramento, che controllano il comportamento dell’algoritmo di apprendimento. Quindi, a differenza dei parametri, che sono valori interni che un modello apprende automaticamente durante il training, questi ultimi vengono scelti dal progettista per guiderne il processo. Nella sua versione attuale, il sistema consente all’utente di regolare gli iperparametri riportati in Tabella 3, mentre per gli altri attribuisce il valore di default [7].

Iperparametro	Descrizione
N_estimators	Numero di alberi nella foresta. In generale più alberi riducono la varianza ma aumentano il tempo di calcolo.
N_jobs	Numero di core CPU da utilizzare in parallelo.
Random_state	Valore numerico iniziale (seed) per rendere i risultati replicabili.

Tabella 3: Tabella degli iperparametri utilizzati

2.4 – AMBIENTE DI SVILUPPO

La fase iniziale del progetto ha previsto l’allestimento di un ambiente di sviluppo Python isolato, al fine di garantire una gestione modulare e controllata dei pacchetti necessari. A tal proposito, è stato creato un **ambiente virtuale** utilizzando Python 3.11, così da assicurare la compatibilità con le librerie impiegate e ridurre eventuali conflitti tra dipendenze. Le librerie utilizzate sono elencate di seguito con una breve descrizione delle loro funzionalità principali:

- **NumPy**: fondamentale per la gestione efficiente di array multidimensionali e per l’esecuzione di operazioni matematiche e statistiche di base.
- **Pandas**: utilizzata per la manipolazione di strutture tabellari (DataFrame), fornisce strumenti avanzati per la gestione, trasformazione e filtraggio dei dati.
- **Scikit-learn**: principale libreria di machine learning in ambiente Python, impiegata per l’addestramento e la valutazione dei modelli predittivi. Include algoritmi, metriche e strumenti per la validazione incrociata [8].
- **Matplotlib e Seaborn**: impiegate per la visualizzazione dei dati e dei risultati, offrono supporto per grafici statici e dinamici, essenziali per l’analisi esplorativa e la presentazione dei modelli.
- **psycopg2**: libreria di interfaccia tra Python e PostgreSQL, utilizzata per effettuare query SQL direttamente dal codice Python, e per caricare i risultati in oggetti Pandas DataFrame.

Una volta definito l’ambiente, si è proceduto all’interrogazione della base di dati. L’obiettivo era ottenere un sottoinsieme di righe affidabili, corrispondenti a condizioni di guida reali e prive di anomalie. Questa selezione mirava a garantire la coerenza semantica dei dati, filtrando solo i record in cui il motore fosse acceso, la velocità maggiore di zero e i valori geografici rientrassero in intervalli plausibili. In tal modo, si è potuto operare su un dataset privo di **outlier** macroscopici e pronto per la fase di elaborazione.

2.5 – IMPLEMENTAZIONE DELLA DASHBOARD

Al fine di rendere accessibile e fruibile il sistema sviluppato anche a utenti non esperti di programmazione, è stata realizzata un’interfaccia grafica interattiva tramite la libreria **Streamlit** [9], uno strumento open-source basato su Python, progettato per facilitare la creazione rapida di applicazioni web orientate all’analisi dei dati e al machine learning.

La dashboard interattiva sviluppata si articola in più sezioni. La prima schermata è dedicata alla configurazione della connessione al database, in cui l’utente è deve inserire i parametri di accesso principali: indirizzo dell’host, porta di comunicazione, nome utente, password e nome del database. Questa scelta permette di mantenere il sistema flessibile e riutilizzabile in ambienti diversi, senza dover modificare il codice sorgente.

Una volta stabilita la connessione, si accede a una sezione dedicata alla visualizzazione dei dati, in cui vengono mostrati sia grafici riepilogativi relativi alla distribuzione delle principali variabili del dataset (istogrammi, boxplot, scatter plot), sia estratti tabellari contenenti un campione dei dati acquisiti. Questa fase ha lo scopo di fornire un primo livello di esplorazione visiva e di verifica della qualità dei dati prima dell’avvio delle fasi di addestramento.

Dopodiché è presente una sezione dedicata al training dei modelli, in cui è possibile selezionare tra diversi algoritmi di apprendimento supervisionato. Al termine del processo di addestramento, la dashboard presenta una pagina riepilogativa contenente i risultati dell’analisi, includendo metriche di valutazione come l’**errore quadratico medio (MSE)**, l’**errore assoluto medio (MAE)** e il **coefficiente di determinazione R^2** .

Il sistema, inoltre, confronta in tempo reale le prestazioni degli algoritmi addestrati e identifica quello con la migliore performance in base alle metriche selezionate, evidenziandolo all’interno della visualizzazione.

CAPITOLO 3 – DASHBOARD INTERATTIVA

Al fine di offrire un’interfaccia utente intuitiva e accessibile per l’analisi dei dati, è stata sviluppata una dashboard interattiva mediante l’utilizzo della libreria open-source **Streamlit**. Quest’ultima si è rivelata particolarmente adatta per applicazioni di Data Science e Machine Learning grazie alla sua capacità di trasformare script Python in applicazioni web interattive.

Streamlit supporta nativamente la visualizzazione di dataframe, grafici (inclusi quelli generati con Matplotlib, Plotly e Seaborn), selettori dinamici, caselle di testo, slider e altro, rendendolo uno strumento altamente versatile per lo sviluppo rapido di prototipi analitici.

L’implementazione ha avuto inizio con l’importazione delle principali librerie Python utilizzate nel progetto. Contestualmente, è stato importato anche il modulo Python dedicato al modello del consumo di carburante, al fine di integrare la logica predittiva direttamente nella dashboard.

3.1 – PAGINA DI ACCESSO

La pagina iniziale dell’interfaccia accoglie l’utente mostrando il nome del progetto in evidenza e richiedendo l’inserimento di una serie di parametri necessari per stabilire la connessione con il database.

I campi obbligatori includono:

- **Host**
- **Port**
- **Database**
- **User**
- **Password** (qualora fosse presente)

Questa sezione è fondamentale per garantire la flessibilità e la portabilità dell'applicazione, permettendone l'utilizzo con differenti istanze di database.

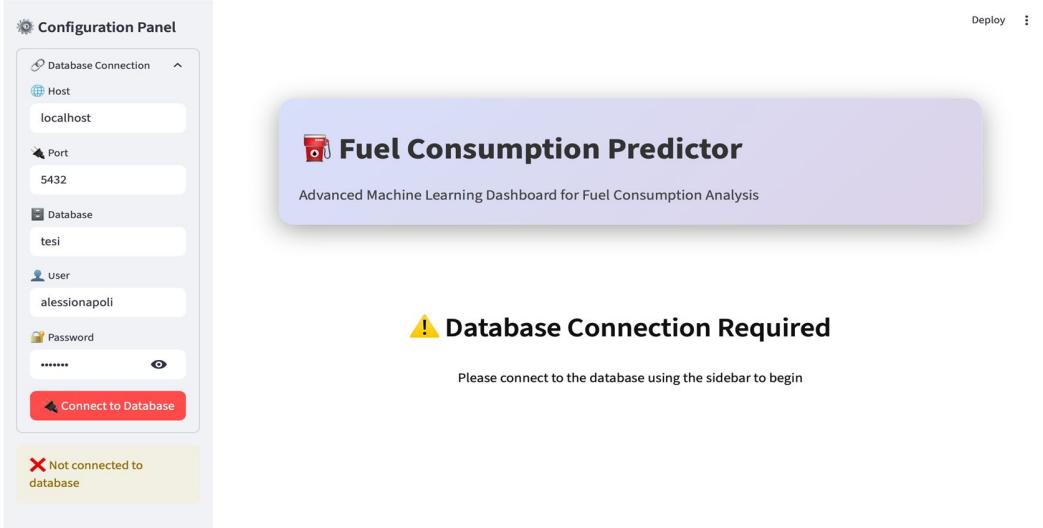


Figura 4: Pagina di connessione al database del progetto

3.2 – PAGINA DI SELEZIONE E VISUALIZZAZIONE DATI

Una volta stabilita la connessione al database e caricati i dati, l'utente viene indirizzato alla pagina di visualizzazione. In questa sezione è possibile selezionare la quantità di record da visualizzare, al fine di gestire il carico computazionale.

In Figura 5 è mostrata la sezione “*Data Loading & Exploration*” della dashboard. L'utente può estrarre e visualizzare i dati di un particolare veicolo della flotta aziendale indicandolo l'identificativo del veicolo (**vehicle_id**).

Una volta cliccato su “*load data*”, sarà possibile visualizzare due grafici di distribuzione, uno relativo al consumo di carburante in un determinato periodo di tempo e uno comparativo di velocità e carburante. La Figura 6 mostra la schermata che sintetizza i dati estratti dal database per un particolare veicolo.

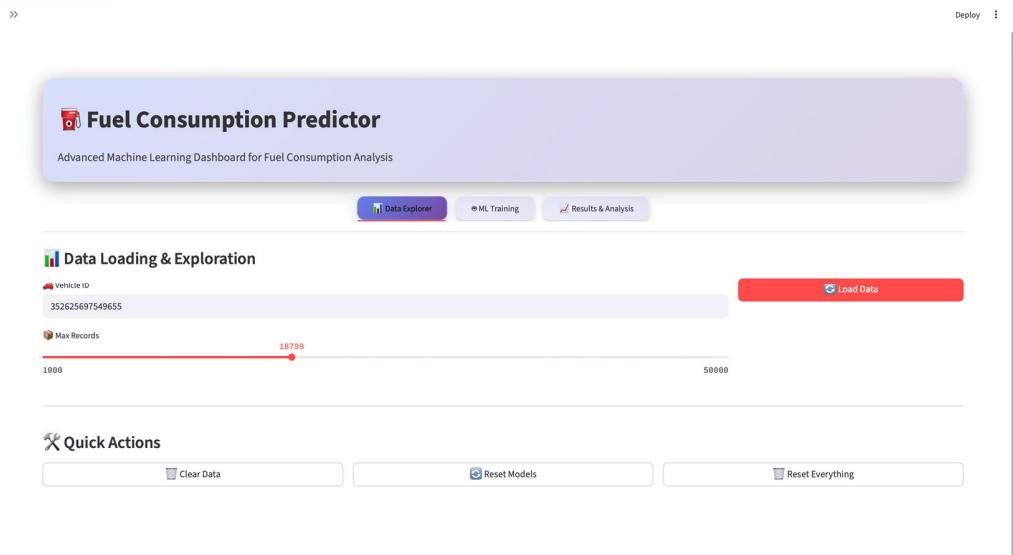


Figura 5: Schermata di scelta dei parametri per la visualizzazione dei dati

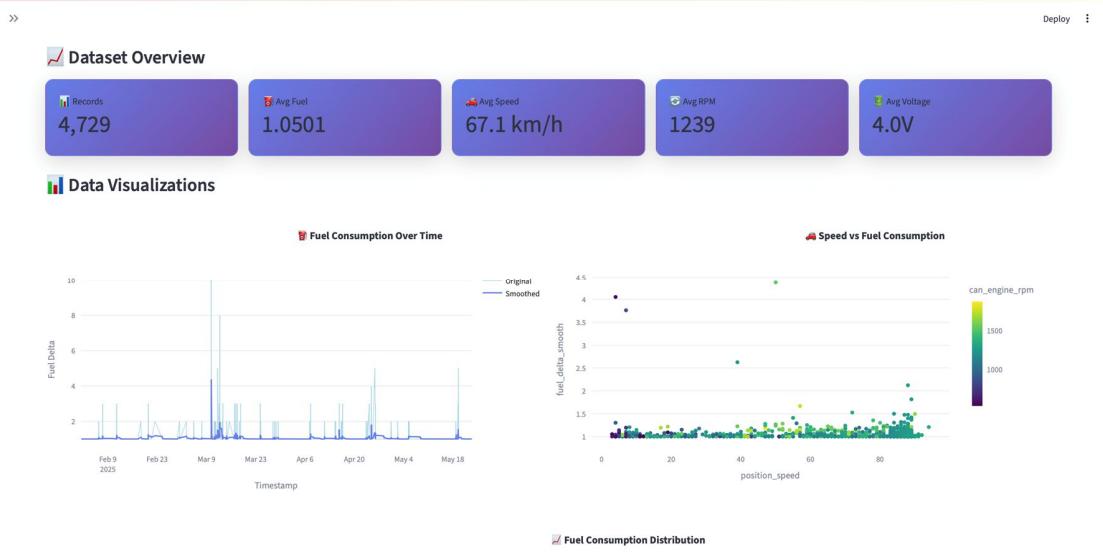


Figura 6: Visualizzazione dei grafici e principali dati di raccolta

3.3 – PAGINA DI ADDESTRAMENTO DEL MODELLO

La terza sezione della dashboard, mostrata in Figura 7, è dedicata all’addestramento dei modelli predittivi. Qui è possibile configurare i parametri principali del training:

- **Test Size:** proporzione dei dati da riservare al test set.
- **Random State:** seme casuale per la riproducibilità della ripartizione dei dati tra training e test set.
- **Selezione degli algoritmi:** l’utente può selezionare uno o più algoritmi da includere nel processo di training, tra cui Random Forest, XGBoost, Linear Regression e Gradient Boosting.

Questa sezione è stata progettata per offrire un alto grado di personalizzazione, mantenendo però un’interfaccia semplice e accessibile.

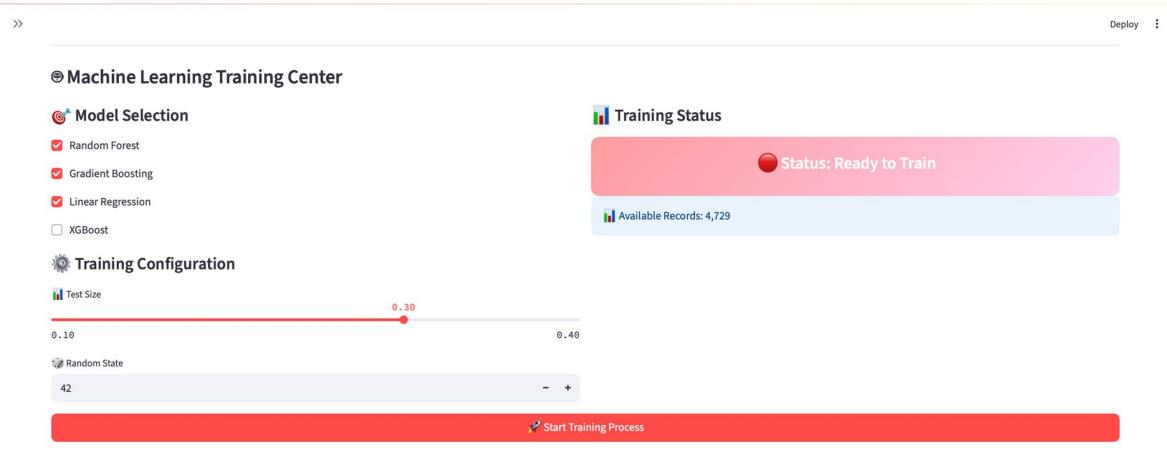


Figura 7: Selezione dei parametri per il training del modello

3.4 – PAGINA DI VALUTAZIONE DEI MODELLI

Una volta completato l'addestramento, l'utente accede alla sezione di **valutazione dei modelli**, in cui vengono mostrati per prima cosa i grafici comparativi tra i modelli allenati.

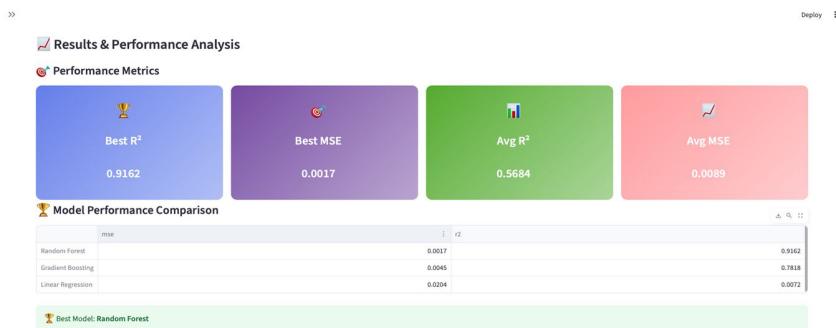


Figura 8: Analisi delle performance

La pagina di valutazione dei modelli, in Figura 8, mostra i punteggi di ogni metrica, constatando il miglior algoritmo. I principali indicatori di valutazione utilizzati includono il Mean Squared Error (MSE) e il Coefficiente R^2 .

Il funzionamento dell'intero sistema è stato testato per diverse configurazioni di dati, algoritmi ed iperparametri. In particolare, nella configurazione che prevedeva l'utilizzo del 80% del dataset come training set e del rimanente 20% come test set e scegliendo come iperparametri *Random_State* pari a 42, *N_Estimators* pari a 100 e *N_Jobs* pari a 25, il modello migliore è risultato essere il Random Forest che ha raggiunto un MSE di 0.0017 e un R^2 di 0.9362.

Infine, la Figura 9 mostra la pagina che visualizza la distribuzione delle feature, utile per interpretare l'importanza relativa delle variabili predittive.



Figura 9: Comparazione dei punteggi e importanza delle feature

CAPITOLO 4 – CONCLUSIONI E SVILUPPI FUTURI

Il presente elaborato ha affrontato il tema della predizione del consumo di carburante in una flotta di veicoli aziendali, attraverso l'applicazione di tecniche di Machine Learning su dati raccolti in tempo reale e successivamente elaborati tramite una pipeline strutturata. L'intero progetto ha avuto origine da un'esperienza di tirocinio presso l'azienda *Rebitral Srl*, nel corso della quale sono stati sviluppati modelli predittivi e strumenti interattivi finalizzati all'analisi e alla valutazione dei dati.

La realizzazione del progetto ha seguito diverse fasi fondamentali:

- **Pre-processing e pulizia del dataset**, in cui vengono gestiti i valori mancanti e normalizzati i dati.
- **Costruzione e valutazione dei modelli di Machine Learning**, in cui viene programmato il modello per il consumo di carburante, utilizzando quattro diversi algoritmi, tra cui Random Forest e XGBoost.
- **Implementazione della dashboard interattiva**, grazie a Streamlit, con la quale è possibile utilizzare il modello predittivo in maniera facile.

L'approccio adottato ha mostrato come strumenti moderni di Machine Learning, se ben integrati in ambienti interattivi, possano supportare efficacemente processi decisionali orientati all'**ottimizzazione dei consumi** e alla **gestione predittiva delle risorse**. In più, grazie a metriche quali MSE, MAE e R², è stata possibile un'analisi accurata dei risultati, evidenziando le varie differenze tra gli algoritmi impiegati.

SVILUPPI FUTURI

Nonostante i risultati ottenuti siano soddisfacenti, sono emerse numerose opportunità per l'ampliamento e il perfezionamento del progetto. Si potrebbe per esempio estendere la base di dati, avendo altre informazioni come le condizioni meteorologiche. Un possibile sviluppo è la realizzazione di un modello che, a partire dai dati telemetrici, predice lo stile di guida e usa quest'informazione per migliorare il modello di consumo di carburante.

Un altro possibile sviluppo potrebbe essere l'implementazione di algoritmi di Deep Learning, utilizzando reti neurali, anche se questo porterebbe a un significativo aumento della complessità computazionale.

Ancora, sarebbe possibile integrare la dashboard in un Docker o in piattaforme cloud (ad esempio Google Cloud Platform, Azure, AWS ecc...), al fine di renderla fruibile a più utenti e su diversi dispositivi.

APPENDICE – MODULI DI PROGRAMMAZIONE SVILUPPATI

```
SELECT * FROM telemetry_temp
WHERE
    ident = '352625697549655'
    AND can_fuel_consumed IS NOT NULL
    AND din_1 = true
    AND engine_ignition_status = true
    AND position_speed > 0
    AND movement_status = true
    AND can_engine_rpm BETWEEN 1 AND 8000
    AND position_latitude BETWEEN -90 AND 90
    AND position_longitude BETWEEN -180 AND 180
    AND position_altitude BETWEEN -500 AND 10000
    AND position_direction BETWEEN 1 AND 359
    AND position_satellites > 4
    AND can_vehicle_mileage < 2000000;
```

Figura 10: Interrogazione SQL sul codice

```
df = pd.read_sql_query(query, conn)
conn.close()
```

Figura 11: Chiusura connessione al database con salvataggio dati

```
df = df.sort_values('timestamp')
df['timestamp'] = pd.to_datetime(df['timestamp'])
df = df.set_index('timestamp')
df['fuel_delta'] = df['can_fuel_consumed'].diff()
df = df[df['fuel_delta'] > 0] # Scarto i valori negativi o nulli
df['fuel_delta_smooth'] =
df['fuel_delta'].ewm(span=20, adjust=False).mean()
```

Figura 12: Applicazione di una media mobile esponenziale (EWM) per normalizzare i risultati

```

feature_cols = [
    'battery_voltage',
    'can_engine_rpm',
    'can_pedal_brake_status',
    'can_throttle_pedal_level',
    'position_altitude',
    'position_direction',
    'position_speed',
    'vehicle_mileage'
]

target_col = 'fuel_delta_smooth'
df['veicle_mileage'] = np.sqrt(df['vehicle_mileage']) * 0.01

```

Figura 13: Selezione delle features da utilizzare

```

df_model = df[feature_cols + [target_col]].dropna()
X = df_model[feature_cols]
y = df_model[target_col]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

```

Figura 14: Addestramento dei modelli

```

# A. Random Forest
evaluate_model("Random Forest",
               RandomForestRegressor(n_estimators=100, random_state=42, n_jobs=-1),
               X_train, y_train, X_test, y_test)

# B. Gradient Boosting
evaluate_model("Gradient Boosting",
               HistGradientBoostingRegressor(random_state=42), X_train, y_train,
               X_test, y_test)

# C. Linear Regression
evaluate_model("Linear Regression", LinearRegression(), X_train,
               y_train, X_test, y_test)

# D. XGBoost
evaluate_model("XGBoost", XGBRegressor(n_estimators=100,
                                         random_state=42, n_jobs=-1),
               X_train, y_train, X_test, y_test)

```

Figura 15: Valutazione dei modelli

```

models = list(results.keys())
mses = [results[m] ['mse'] for m in models]
r2s = [results[m] ['r2'] for m in models]

plt.figure(figsize=(8, 4))
plt.bar(models, r2s, color='skyblue')
plt.title("Confronto R2 (più alto è meglio)")
plt.ylabel("R2")
plt.ylim(min(r2s) - 0.05, max(r2s) + 0.05)
plt.axhline(0, color='gray', linestyle='--')
plt.grid(axis='y', linestyle=':', alpha=0.6)
plt.tight_layout()
plt.show()

plt.figure(figsize=(8, 4))
plt.bar(models, mses, color='salmon')
plt.title("Confronto MSE (più basso è meglio)")
plt.ylabel("Mean Squared Error")
plt.ylim(0, max(mses) + 0.01)
plt.grid(axis='y', linestyle=':', alpha=0.6)
plt.tight_layout()
plt.show()

```

Figura 16: Visualizzazione dei risultati tramite grafici MatPlotLib

BIBLIOGRAFIA E SITOGRADIA

- [1] International Energy Agency (IEA), «Fuel Economy in Major Car Markets,» [Online]. Available: <https://www.iea.org/reports/fuel-economy-in-major-car-markets>.
- [2] European Energy Agency (EEA), «Greenhouse Gas Emissions from Transport,» [Online]. Available: <https://www.eea.europa.eu/en/analysis/indicators/greenhouse-gas-emissions-from-transport>.
- [3] G. Developers, «Machine Learning Crash Course.,» Google, [Online]. Available: <https://developers.google.com/machine-learning/crash-course>.
- [4] G. W. D. H. T. & T. R. James, An Introduction to Statistical Learning: with Applications in R. Springer., 2021.
- [5] L. Breiman, Random Forests. Machine Learning., 2001.
- [6] T. & G. C. Chen, XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining., 2016.
- [7] «XGBoost Documentation.,» [Online]. Available: <https://xgboost.readthedocs.io/>.
- [8] «Sci-kit Learn. User Guide.,» [Online]. Available: https://scikit-learn.org/stable/user_guide.html?utm_source=chatgpt.com.
- [9] «Streamlit Documentation.,» [Online]. Available: <https://docs.streamlit.io/>.
- [10] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2^a edizione, O'Reilly Media, 2019.

Indice delle figure

Figura 1: Un esempio di cluster	6
Figura 2: Workflow di progetto	11
Figura 3: Schema a blocchi del progetto svolto	13
Figura 4: Pagina di connessione al database del progetto	23
Figura 5: Schermata di scelta dei parametri per la visualizzazione dei dati.....	24
Figura 6: Visualizzazione dei grafici e principali dati di raccolta.....	24
Figura 7: Selezione dei parametri per il training del modello	25
Figura 8: Analisi delle performance.....	26
Figura 9: Comparazione dei punteggi e importanza delle feature.....	27
Figura 10: Interrogazione SQL sul codice	30
Figura 11: Chiusura connessione al database con salvataggio dati.....	30
Figura 12: Applicazione di una media mobile esponenziale (EWM) per normalizzare i risultati	30
Figura 13: Selezione delle features da utilizzare.....	31
Figura 14: Addestramento dei modelli.....	31
Figura 15: Valutazione dei modelli.....	31
Figura 16: Visualizzazione dei risultati tramite grafici MatPlotLib.....	32

RINGRAZIAMENTI

Desidero ringraziare il mio relatore e Tutor Accademico, il Prof. Antonio Parziale, per il supporto e la disponibilità dimostrati durante l'intero percorso di tirocinio e nella stesura della presente tesi.

Un sentito ringraziamento va all'azienda Rebitral Srl, che mi ha accolto con professionalità e disponibilità, consentendomi di partecipare attivamente a un progetto stimolante e formativo.

Ringrazio la mia famiglia e la mia ragazza, per la pazienza nell'aiutarmi ad affrontare le mie difficoltà durante questo percorso.

Infine, desidero ringraziare me stesso, per la determinazione con cui ho affrontato ogni sfida e momento buio, credendo sempre nelle mie capacità e nei miei obiettivi.

Che questo sia solo un passo del sentiero che sto percorrendo.

Grazie.