CarICon:

Tempo di accelerazione di un auto

Componenti del gruppo

* Giuseppe Caggiano [MAT. 735400], g.caggiano8@studenti.uniba.it

Link progetto GitHub:

[CarICon](https://github.com/Speppo/CarICon.git)

Anno Accademico 2023-2024

**INDICE**

* *Capitolo 0 : Introduzione*
* *Capitolo 1 : Dataset e Features*
* *Capitolo 2 : Clustering (Apprendimento non supervisionato)*
* *Capitolo 3 : Knowledge Base attraverso Prolog*
* *Capitolo 4 : Regressione sull’accelerazione*
* *Capitolo 5 : KNN*
* *Capitolo 6 : AdaBoost*
* *Capitolo 7 : Random Forest*
* *Capitolo 8 : Alberi di Decisione*
* *Capitolo 9 : Risultati finali*
* *Capitolo 10: Conclusione*

**Capitolo 0: Introduzione**

L'obiettivo di questo progetto è stato analizzare ed effettuare operazioni sul tempo di accelerazione da 0 a 100 km/h di un auto tramite un dataset di Kaggle chiamato Car\_specification\_dataset\_1945\_2020. Questo dataset contiene modelli di auto con varie caratteristiche per singola auto, tra cui il target di regressione scelto, ovvero il tempo di accelerazione da 0 a 100 km/h .Tra le variabili rilevanti per l'analisi dell'accelerazione vi sono, ad esempio, la lunghezza, la larghezza, la massa, la potenza del motore e la coppia massima, che influenzano direttamente le prestazioni di accelerazione.

Ho utilizzato tecniche di :

-**apprendimento non supervisionato** per l’implementazione del k-means per individuare i cluster di appartenenza di ogni occorrenza del dataset (creando una nuova caratteristica con il cluster ottenuto per migliorare accuratezza)

-**apprendimento supervisionato** con regressione tramite KNN e Alberi di Decisione

Inoltre, il progetto utilizza **Prolog** per ottenere nuove informazioni attraverso il ragionamento automatico.

**Requisiti funzionali**

Il progetto è stato realizzato con Python versione 3.12 per la grande vastità di librerie che aiutano nell’implementazione. L’IDE che ho usato per la stesura del codice è stato PyCharm, semplice e funzionale. Le librerie che ho utilizzato sono le seguenti:

- **pandas**: per la manipolazione e analisi dei dati.

- **scikit-learn**: per il preprocessing, clustering e apprendimento supervisionato.

- **matplotlib**: per la visualizzazione dei dati attraverso i grafici

- **pyswip**: per l'integrazione con Prolog.

**Configurazioni iniziali**

Installare SWI-Prolog dal sito ufficiale <https://www.swi-prolog.org/download/stable>, fondamentale per l’utilizzo della libreria pyswip. Successivamente, dopo aver aperto il progetto con l’IDE preferito, avviare lo script install.py per scaricare tutte le librerie importanti per il corretto funzionamento del progetto.

**Capitolo 1 : Dataset e Features**

Il dataset è stato trovato sul sito di Kaggle, una sorta di archivio di dataset, e contiene modelli di auto, con annesse caratteristiche, prodotte dal ’45 al 2020. Molte colonne contengono dati mancanti o presentano un alto grado di sparsità, aspetto che ha richiesto un significativo lavoro di preprocessing per ridurre la dimensione del dataset ed eliminare valori mancanti o irrilevanti, tra cui:

* maximum\_torque\_n\_m (coppia massima del motore)
* injection\_type (tipo di iniezione del motore)
* engine\_hp (numero indicante i cavalli del motore)
* boost\_type (tipo di turbina)
* transmission (tipo di trasmissione manuale, automatica o sequenziale)

oltre ovviamente alle dimensioni del mezzo e ad altre caratteristiche.

I dati sono poi preprocessati per renderli pronti all’analisi.

Per prima cosa, come detto precedentemente, ho deciso di ‘scremare’ le caratteristiche non importanti per l’obiettivo alleggerendo il mio dataset. Così ho eliminato le colonne che non avevano impatto sul tempo di accelerazione, oppure quelle con troppi valori vuoti.

Successivamente, oltre a questo sono stati fatti alcuni aggiustamenti nelle colonne che erano fondamentali per il tempo di accelerazione, al fine di rendere più snello e leggibile il dataset, ma anche per eliminare dei problemi sintattici che davano problemi nella base di conoscenza Prolog.

Dopo queste fasi viene creato un nuovo file chiamato preprocessed.csv

**Capitolo 2 : Clustering (Apprendimento non supervisionato)**

L’apprendimento non supervisionato è una branca dell’apprendimento automatico in cui l’agente viene addestrato su un insieme di dati senza etichette.

- Clustering : tecnica di machine learning che suddivide i dati in gruppi omogenei chiamati **cluster**. L'obiettivo di questa fase è ottenere un secondo dataset che aggiunge a preprocessed.csv una nuova feature con il cluster ottenuto.

Personalmente ho applicato l’algoritmo K-Means sfruttando la distanza euclidea. Inoltre ho deciso di escludere la feature ID perché può non essere un dato importante allo scopo del mio progetto.

Ora andiamo con l’individuazione del numero ottimale di cluster utilizzabili. Per fare ciò ho implementato la tecnica della curva del gomito con un intervallo da 1 a 20.

Immagine che contiene testo, schermata, linea, diagramma

Descrizione generata automaticamente



Quindi ottenuto il numero ottimale di cluster questo viene usato per calcolare il cluster di appartenenza per ogni occorrenza, poi inserito nella caratteristica a cui appartiene.

Non siamo sicuri però che sapere il cluster di appartenenza possa aiutare nella regressione quindi otteniamo un nuovo file chiamato clustered.csv

**Capitolo 3 : Knowledge Base attraverso Prolog**

A questo punto creo una Knowledge Base sfruttando il Prolog con la libreria Pyswip di Python, per ottenere nuova conoscenza utile per creare due nuovi dataset per la valutazione dell’apprendimento automatico sfruttando ragionamento automatico e la struttura di individui e relazioni:

- **Individui**: Knowledge Base popolata con fatti del dataset preprocessato modellando il dominio, e trova la classe di individui con simbolo di funzione annesso car(C) cioè l’auto con ID C.

- **Proprietà**: le proprietà associate a individuo car(C) corrispondono alle feature nel dataset che sono usate ai fini dell’ottenere nuove informazione nel ragionamento automatico

- engine\_hp(car(C), H)

- engine\_hp(car(C), H)

- curb\_weight\_kg(car(C), K)

- height\_mm(car(C), HeightMm)

- width\_mm(car(C), WidthMm)

- capacity\_cm3(car(C), Cap)

- number\_of\_cylinders(car(C), N)

- max\_speed\_km\_per\_h(car(C), M)

- make(car(C), Manufacturer)

- cylinder\_layout(car(C), Cylinder\_layout)

- max\_trunk\_capacity\_l(car(C), Capacity)

- model(car(C), MaxModel)

- year\_from(car(C), Year)

- **Clausole Definite**: clausole definite per effettuare ragionamento . Ad esempio:

- **Clausole rapporti:**

- calcolo coefficiente areodinamico tramite caratteristiche dell’auto e densità dell’aria

- calcolo potenza specifica con rapporto cavalli/cilindrata

- calcolo rapporto lunghezza/altezza

- calcolo rapporto lunghezza/potenza

- **Clausole di Categorizzazione**:

- classificazione potenza auto in bassa, media e alta

- classificazione tipo auto in sportiva, coupè , berlina e suv

- calcolo capacità media del bagagliaio

- calcolo modello più potente di un marchio

- **Clausole ausiliarie** : clausole per capire quale sia l’elemento più frequente di una lista o la forma dell’auto in base alle caratteristiche ‘fisiche’ dell’auto.

- **Query** : dopo le clausole, si effettuano query per il calcolo dei valori da inserire nelle occorrenze delle nuove feature, in cui va sostituita la variabile C in car(C) in ogni clausola sfruttata con ID auto. Con esse creo due nuovi dataset **prolog\_generated.csv** ( con le feature create in questa fase) e **clustered\_plus\_prolog\_generated.csv** (con feature di clustered.csv + le feature create in questa fase)

**Capitolo 4 : Regressione sull’accelerazione**

Dopo aver generato i vari dataset, procediamo col target su tempo di accelerazione.

Usiamo la libreria scikit-learn di Python per svolgere il training tramite i regressori KNN, AdaBoost, Random Forest e Alberi di decisione per ogni dataset generato.

In ogni fase di addestramento, si è cominciato con l'encoding delle variabili categoriche (differenti in base al dataset) utilizzando la classe **OrdinalEncoder**.

Successivamente, sono stati definiti gli iperparametri da ottimizzare per ciascun tipo di regressore, insieme ai rispettivi valori e intervalli da testare.

L'addestramento dei modelli è stato condotto con un approccio di **10-fold cross-validation,** applicato per ogni combinazione di iperparametri, e le metriche di valutazione riportate rappresentano la media dei risultati ottenuti sui 10 modelli addestrati per ciascuna combinazione. Per garantire un comportamento deterministico, dove possibile (nei modelli AdaBoost, Decision Tree e Random Forest), il parametro *random\_state* è stato fissato a 0.

Le metriche di valutazione selezionate sono le seguenti:

* *neg\_mean\_absolute\_error*
* *neg\_mean\_squared\_error*
* *max\_error*

Infine, per ottenere una comprensione più approfondita dei fattori che influenzano la variabile obiettivo calcola la **feature importance** sul modello migliore per ogni regressore e per ogni dataset usato.

**Capitolo 5 : KNN**

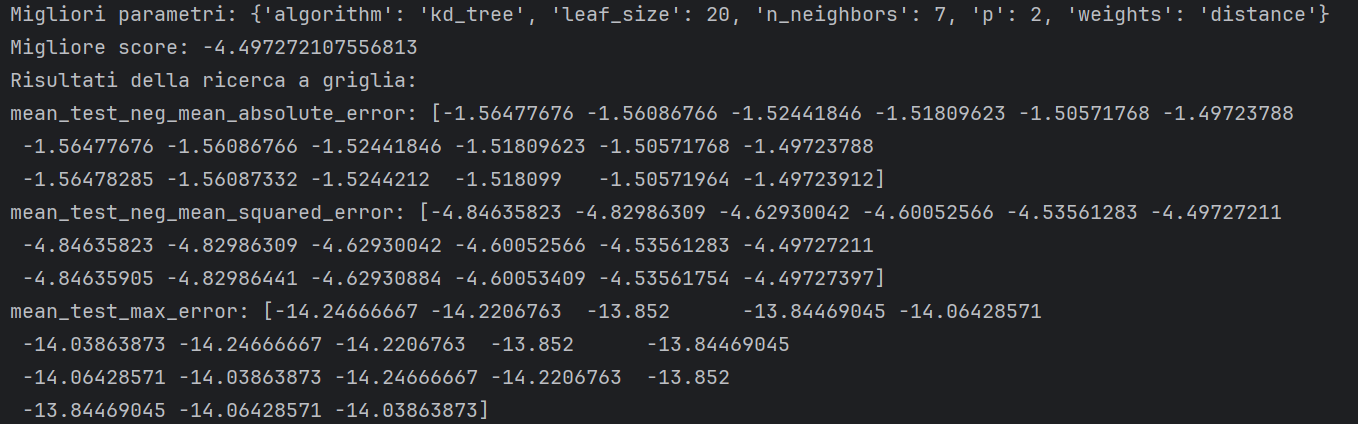
Il primo modello di regressione analizzato sui quattro dataset precedentemente generati è il K-Nearest Neighbors (KNN). Le prestazioni del KNN sono state valutate utilizzando la *10-fold Cross-Validation*, testando tutte le combinazioni dei seguenti iperparametri:

* **n\_neighbors**: sono stati testati i valori 3, 5 e 7, rappresentanti il valore di kkk.
* **weights**: sono stati valutati i metodi *uniform* e *distance* per il calcolo delle distanze. Con *uniform*, tutti i vicini hanno lo stesso peso, mentre *distance* assegna un peso maggiore ai vicini più prossimi.
* **algorithm**: è stato testato *kd-tree* come tipo di algoritmo per la ricerca dei kkk punti più vicini.
* **p**: è stato utilizzato il valore 2, specificando l’uso della distanza euclidea per identificare i vicini.
* **leaf\_size**: sono stati provati i valori 20, 30 e 40, i quali definiscono la dimensione massima delle foglie dell’albero kd-tree.

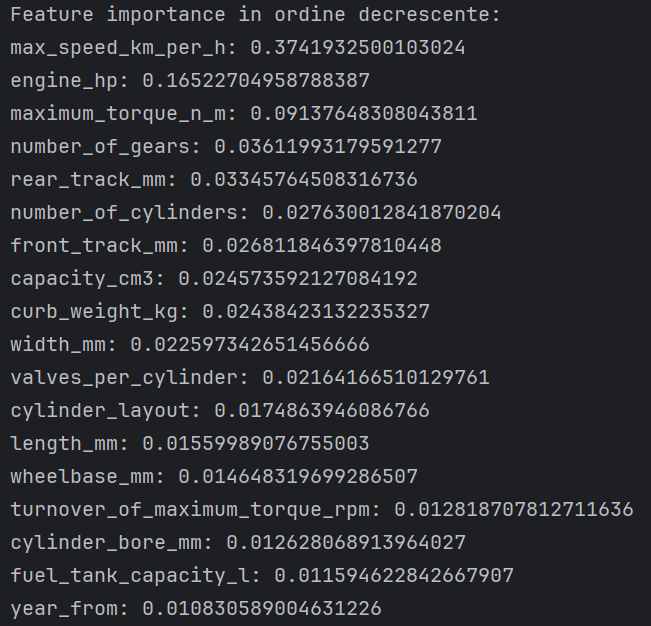
Questa configurazione ha permesso di ottimizzare il modello in modo completo, esplorando le diverse combinazioni di parametri per individuare la configurazione migliore.

**Valutazioni dei dataset**

-preprocessed.csv



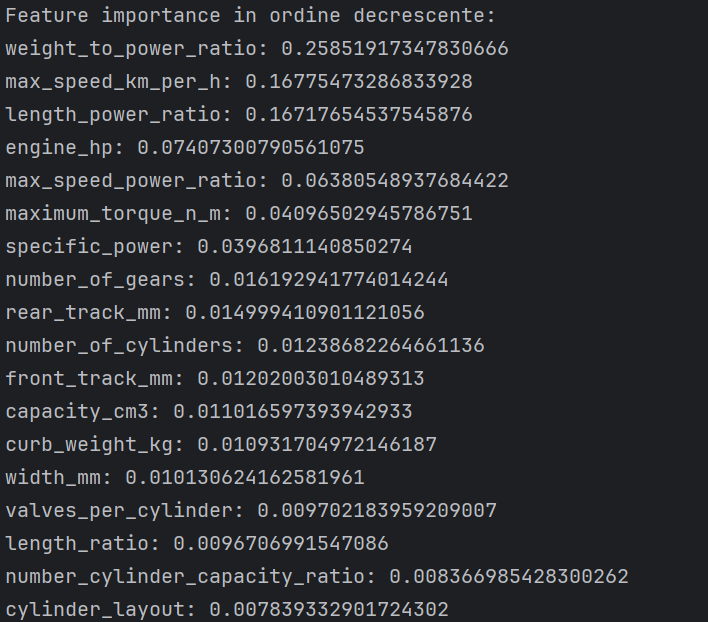
Feature importance:



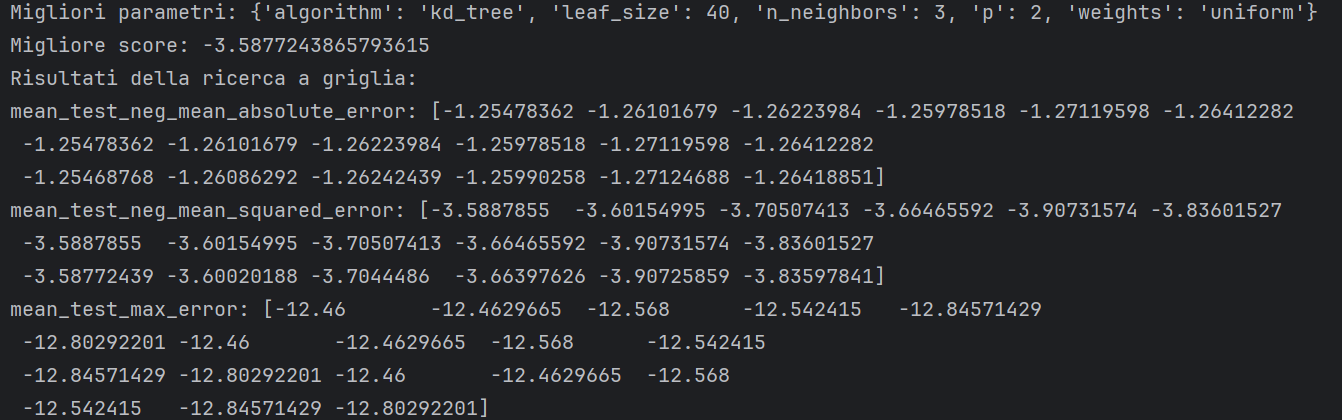
-clustered\_plus\_prolog\_generated.csvImmagine che contiene testo, schermata, Carattere

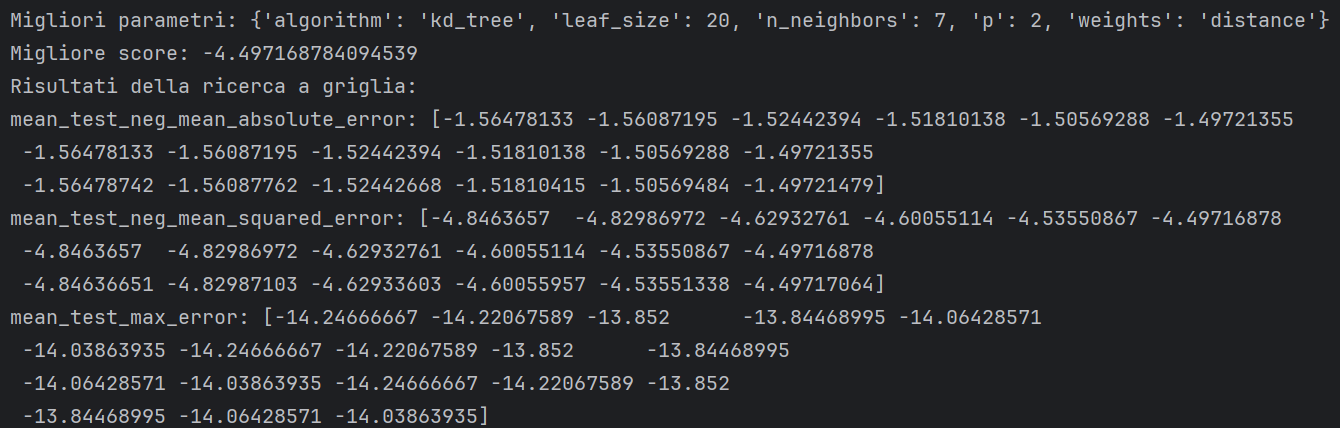
Descrizione generata automaticamente

Feature importance:



-prolog\_generated.csv



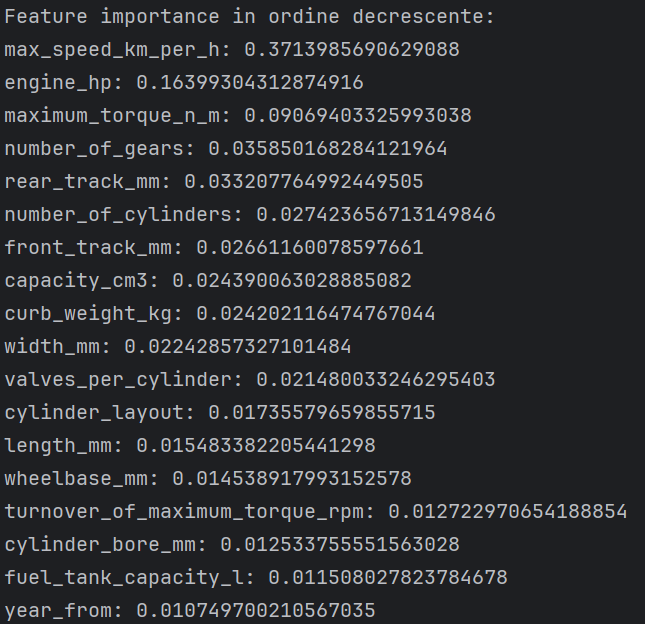
Feature importance:

-clustered.csv

Immagine che contiene testo, schermata, Carattere, design

Descrizione generata automaticamente

Feature importance:

****

**Capitolo 6: AdaBoost**

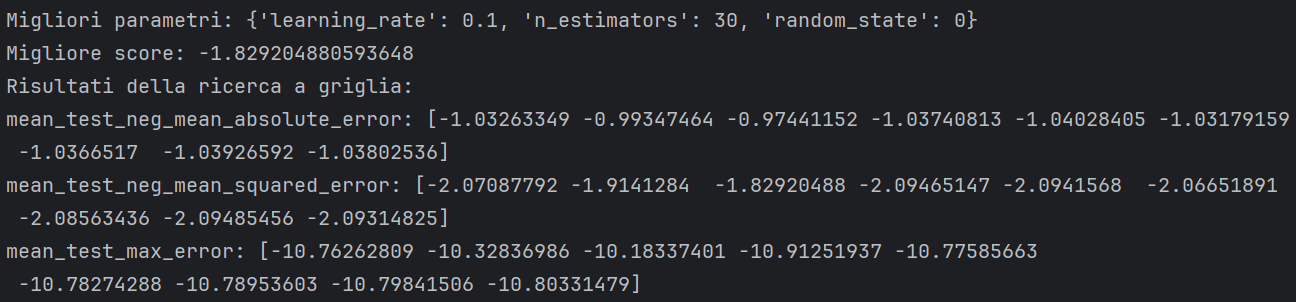
Il secondo modello di regressione analizzato sui quattro dataset è l’AdaBoost. Le prestazioni sono state valutate tramite *10-fold Cross-Validation*, testando tutte le combinazioni dei seguenti iperparametri:

* n\_estimators: i valori testati sono 10, 20 e 30, e rappresentano il numero di “weak learners” utilizzati per costruire il modello di boosting.
* learning\_rate: sono stati provati i valori 0.1, 0.01 e 0.001; questo parametro controlla di quanto viene aggiornato il peso di ogni weak learner durante l'addestramento.

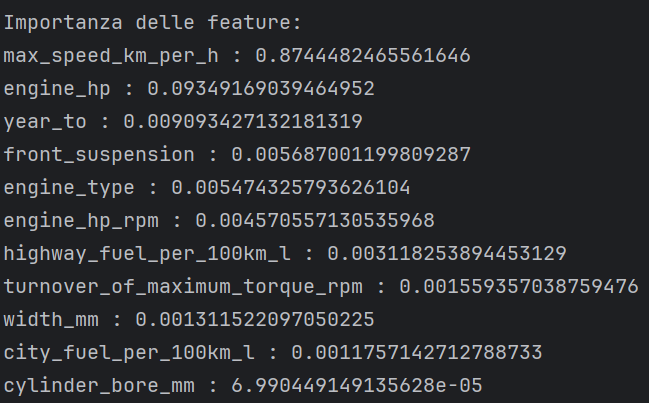
Questa configurazione ha permesso di esplorare e ottimizzare le performance del modello, individuando la combinazione di iperparametri più efficace per ciascun dataset.

**Valutazioni dei dataset**

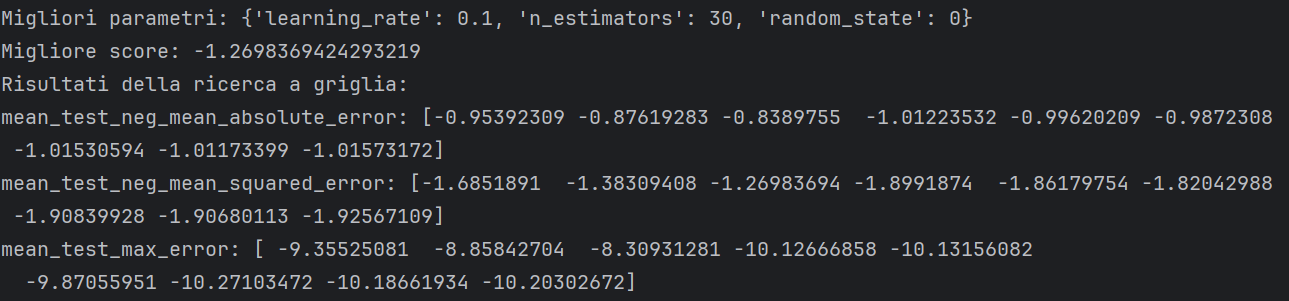
-preprocessed.csv

****

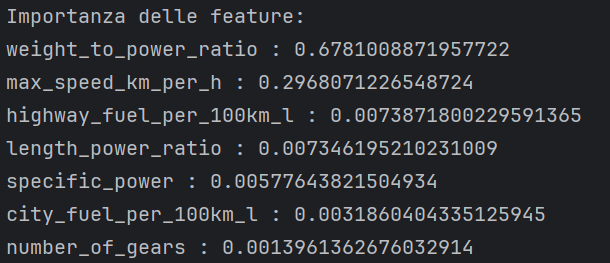
Feature importance:

****

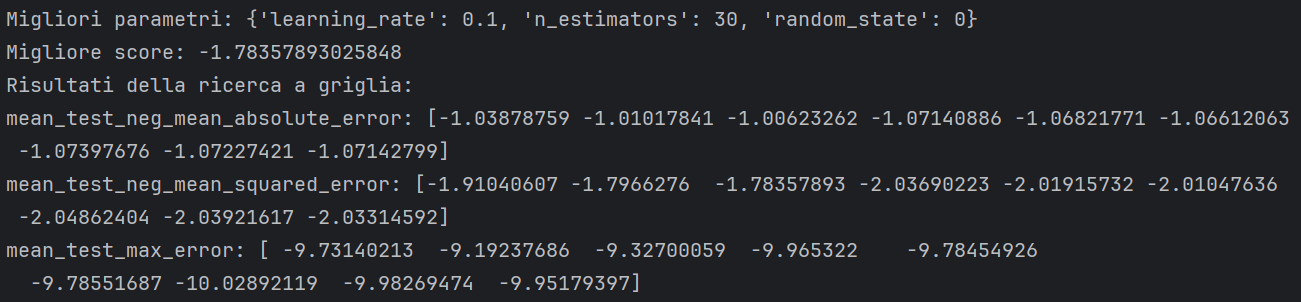
-clustered\_plus\_prolog\_generated.csv



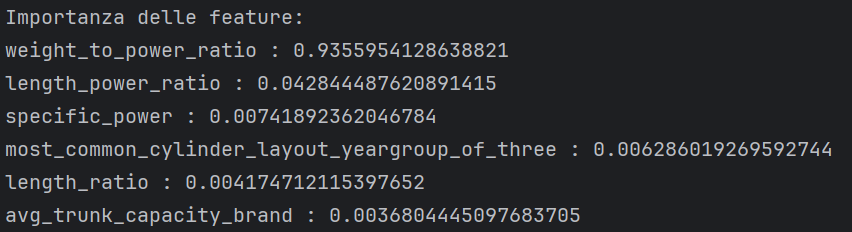
Feature importance:

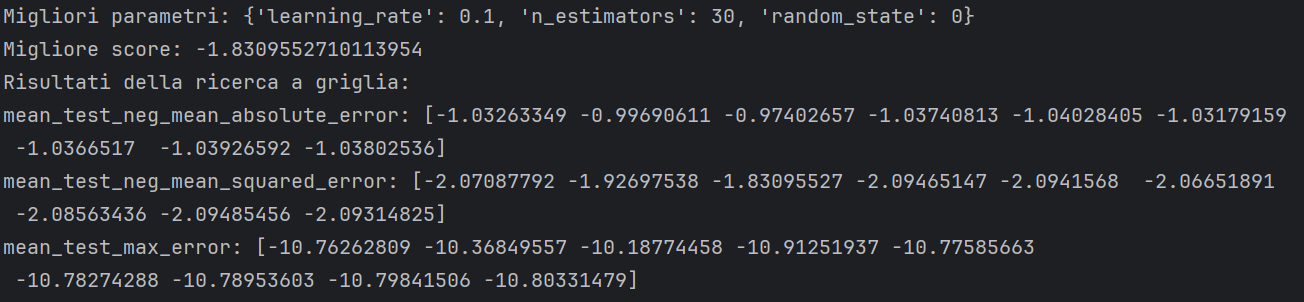
****

-prolog\_generated.csv

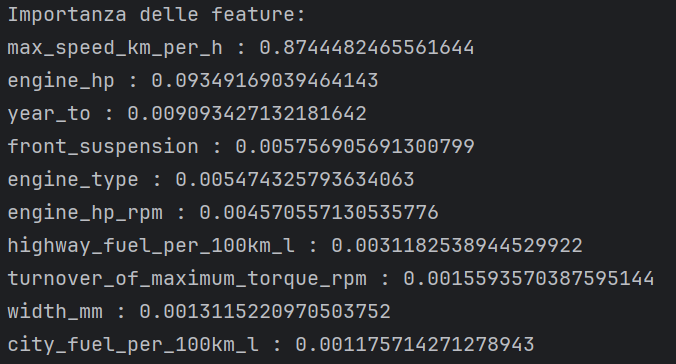
****

Feature importance:

****

-clustered.csv

Feature importance:

****

**Capitolo 7: Random Forest**

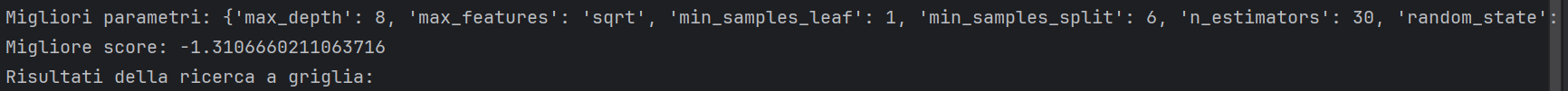
Il terzo modello di regressione analizzato sui quattro dataset è il Random Forest. Le prestazioni di questo modello sono state valutate utilizzando la *10-fold Cross-Validation*, testando tutte le combinazioni dei seguenti iperparametri:

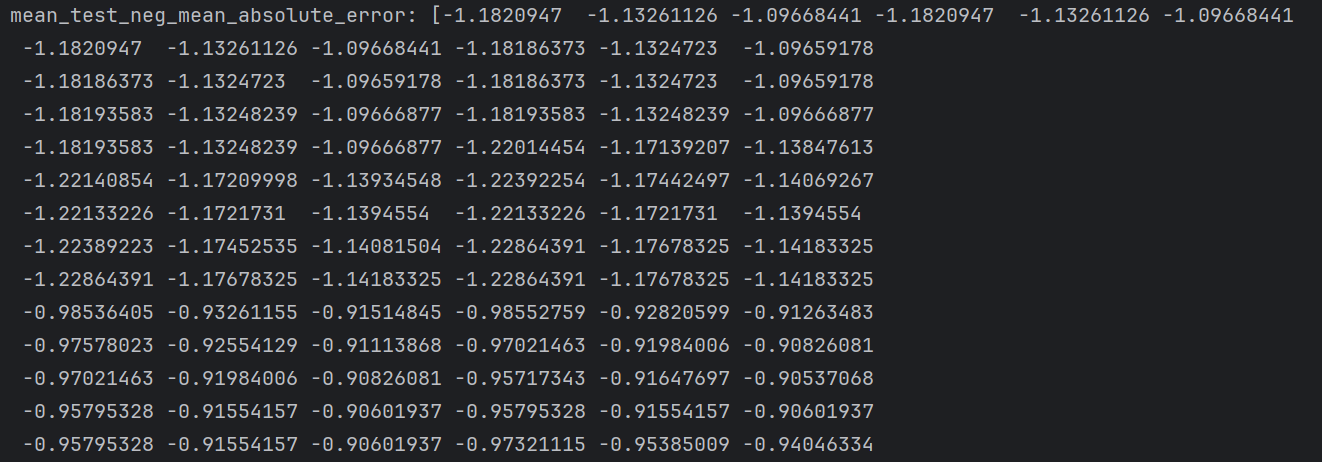
* n\_estimators: i valori testati sono 10, 20 e 30, che rappresentano il numero di alberi decisionali generati nel modello.
* max\_depth: sono stati testati i valori 4, 6 e 8, indicando la profondità massima consentita per ciascun albero decisionale.
* min\_samples\_split: con valori di 2, 4 e 6, questo parametro stabilisce il numero minimo di campioni necessari per suddividere un nodo interno.
* min\_samples\_leaf: i valori testati sono 1, 2 e 4, indicando il numero minimo di campioni richiesti per una foglia.
* max\_features: i valori testati sono *sqrt* e *log2*. Con *sqrt*, il numero di feature selezionate per ciascun albero è la radice quadrata del totale, mentre *log2* seleziona un numero di feature pari al logaritmo in base 2 del totale.

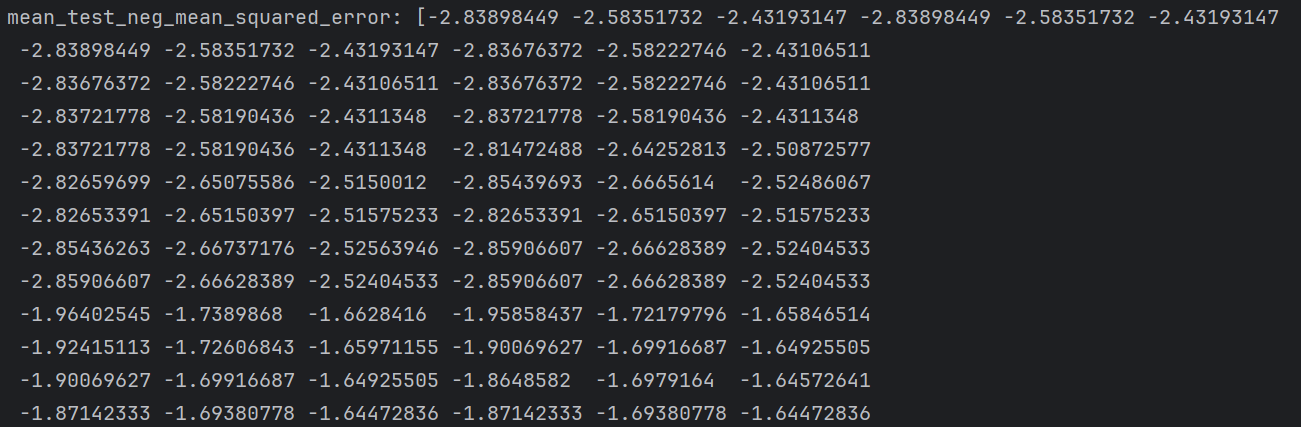
Per ciascun dataset, sono state selezionate e riportate le cinque migliori combinazioni di iperparametri, poiché presentare tutte e diciotto le configurazioni risulterebbe poco leggibile.

**Valutazioni dei dataset**

-preprocessed.csv

****

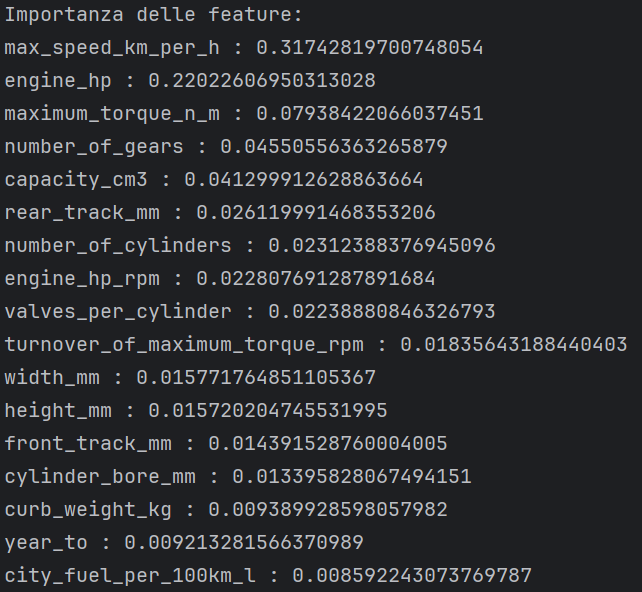
****

****

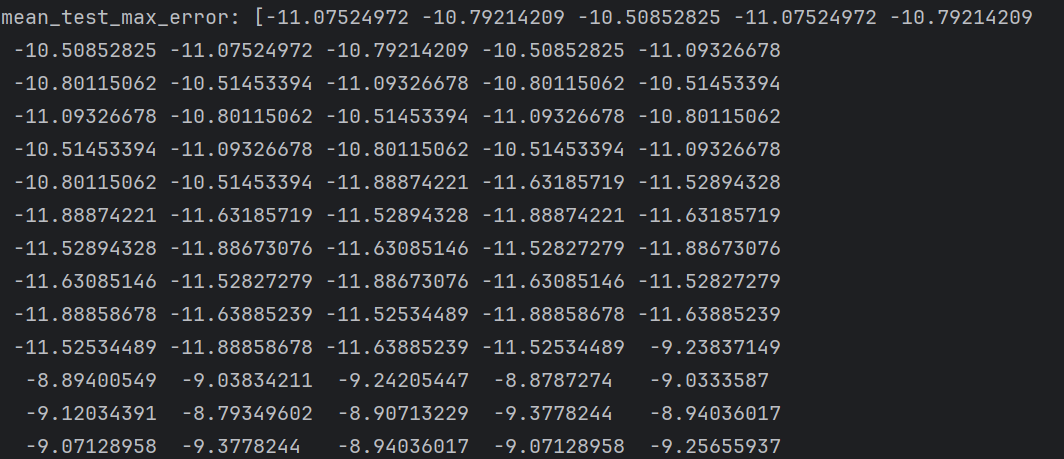
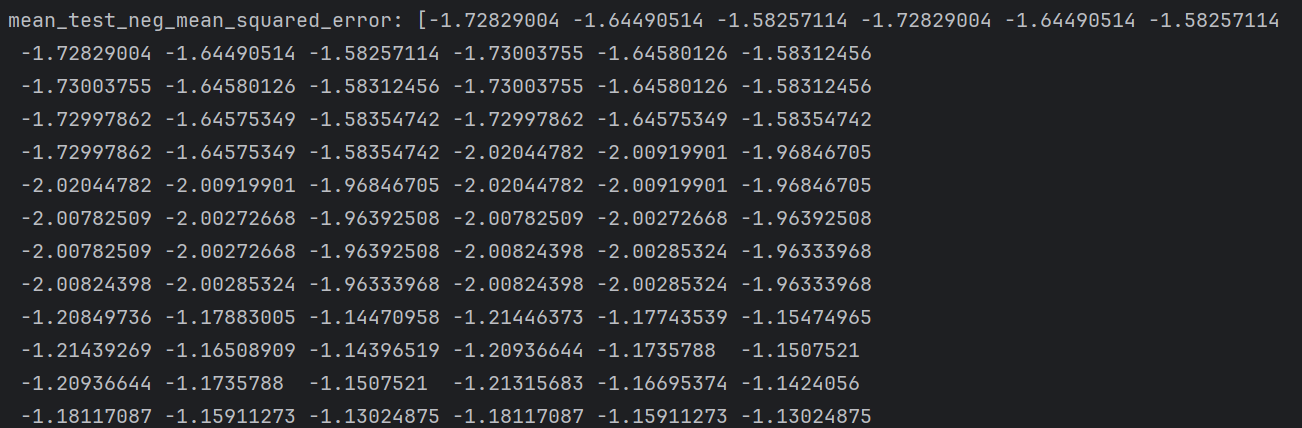
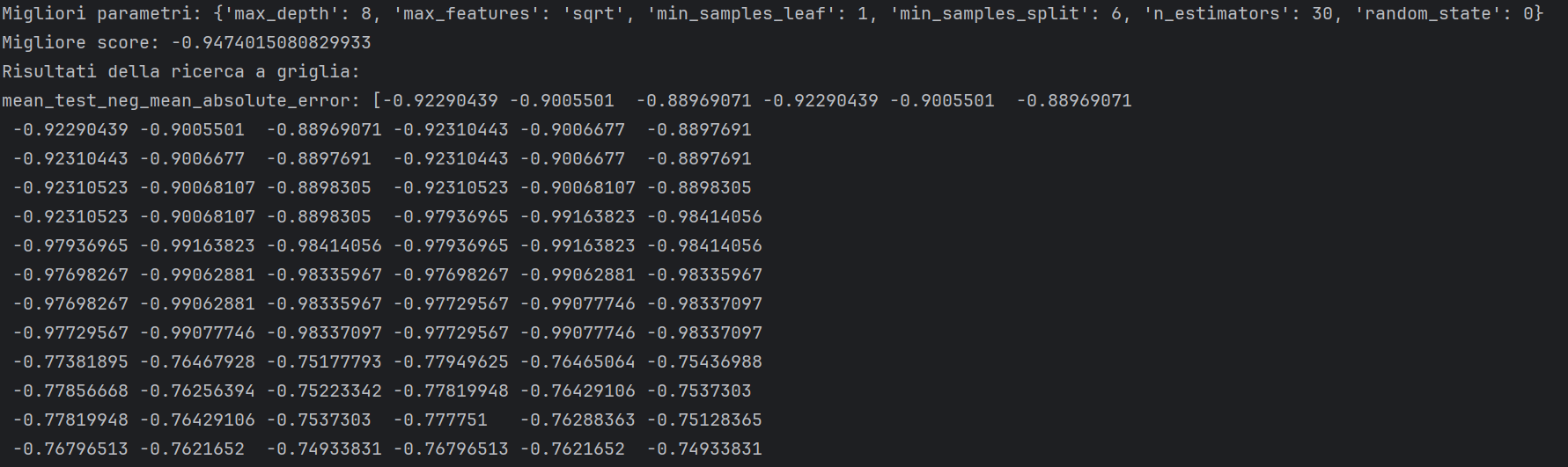
**Immagine che contiene testo, schermata

Descrizione generata automaticamente**

Feature importance:



-clustered\_plus\_prolog\_generated.csv



Feature importance:

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

-prolog\_generated.csv

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Feature importance:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

-clustered.csvImmagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Feature Importance:

Immagine che contiene testo, schermata, menu, Carattere

Descrizione generata automaticamente

**Capitolo 8: Alberi di Decisione**

Il quarto modello di regressione analizzato sui quattro dataset è il Decision Tree. Le prestazioni di questo modello sono state valutate attraverso la *10-fold Cross-Validation*, testando tutte le combinazioni dei seguenti iperparametri:

* max\_depth: i valori testati sono *None* e un intervallo da 3 a 8, che rappresentano la profondità massima consentita per l'albero.
* min\_samples\_split: con valori di 2, 5 e 10, questo parametro indica il numero minimo di campioni necessari per suddividere un nodo interno.
* min\_samples\_leaf: i valori testati sono 1, 2 e 4, indicando il numero minimo di campioni richiesti per una foglia dell'albero.
* max\_features: sono stati provati i valori 0.5, *sqrt* e *log2*. Con 0.5, il modello considera circa il 50% delle feature per ciascuna suddivisione; *sqrt* seleziona il numero di feature pari alla radice quadrata del totale, e *log2* seleziona il numero di feature pari al logaritmo in base 2 del totale.

**Valutazioni dei dataset**

-preprocessed.csv Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Feature importance:

Immagine che contiene testo, schermata, menu, Carattere

Descrizione generata automaticamente

-clustered\_plus\_prolog\_generated.csv

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Feature importance:

Immagine che contiene testo, schermata, Carattere, menu

Descrizione generata automaticamente

-prolog\_generated.csv

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Feature importance:

Immagine che contiene testo, schermata, Carattere, design

Descrizione generata automaticamente

- clustered.csv

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

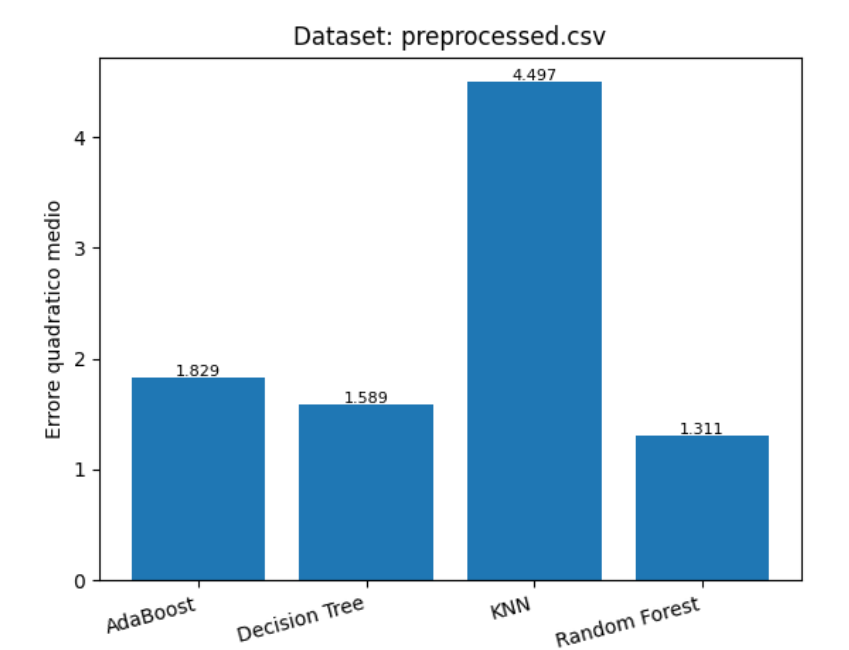
Feature importance:

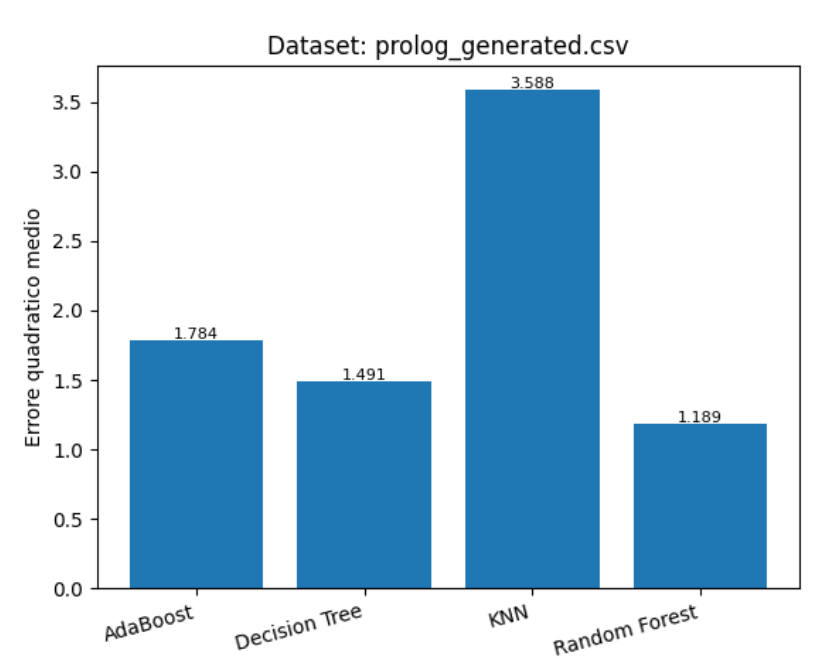
Immagine che contiene testo, schermata, Carattere, menu

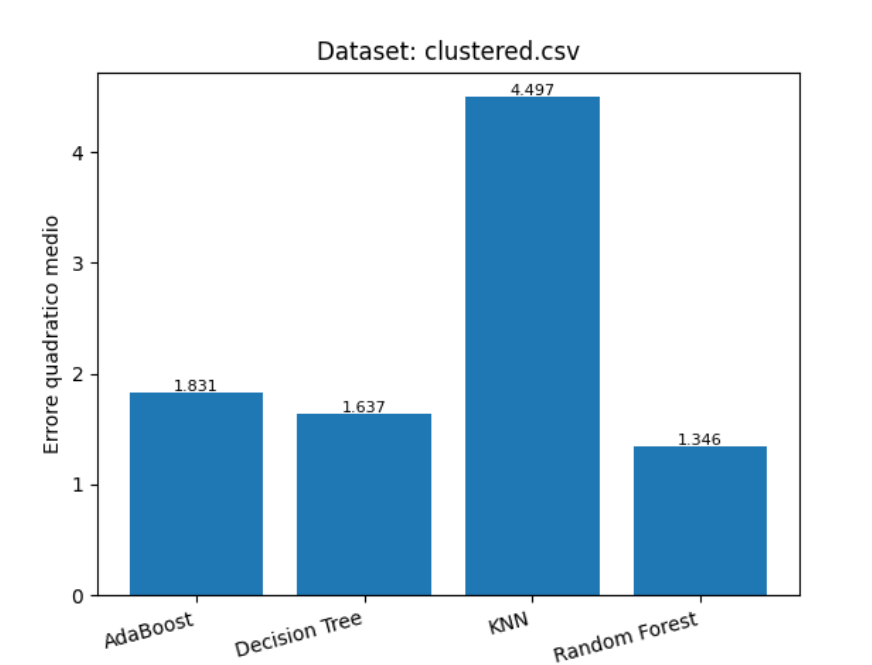
Descrizione generata automaticamente

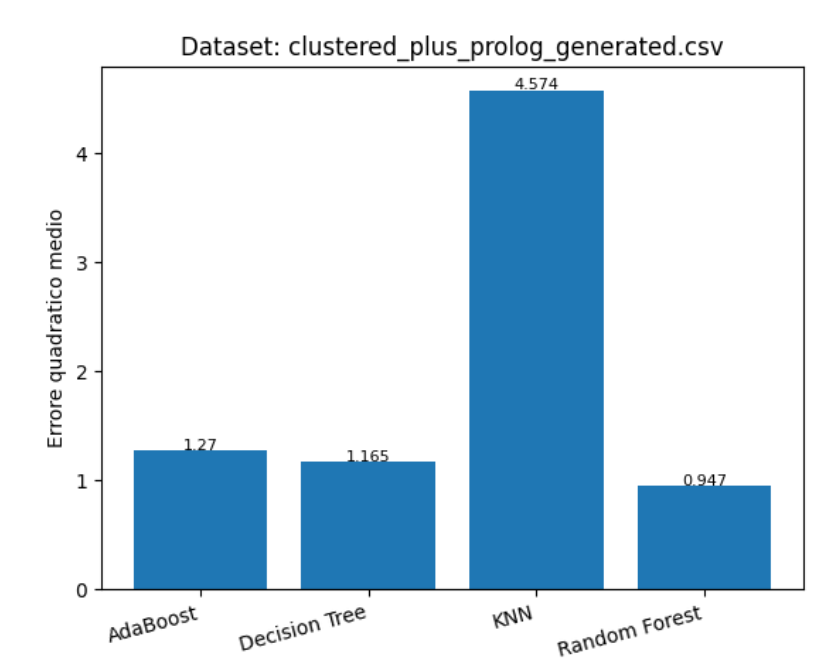
**Capitolo 9 : Risultati Finali**

Possiamo confrontare adesso i vari risultati ottenuti confrontando tutti i modelli









Come possiamo vedere dai risultati ottenuti il miglior regressore è Random Forest sul dataset clustered\_plus\_prolog\_generated.csv

**Capitolo 10 : Conclusioni e Sviluppi Futuri**

In questo progetto, il framework costruito ha dimostrato l’efficacia nell'analisi delle prestazioni di accelerazione, integrando algoritmi di machine learning .Le operazioni di preprocessing hanno permesso di selezionare solo le variabili più significative , migliorando così l’accuratezza delle predizioni dei regressori. I modelli hanno mostrato l'importanza di alcune variabili come la potenza del motore e la velocità massima, fondamentali per il tempo di accelerazione

Come sviluppi futuri, si potrebbe pensare di ampliare il dataset, magari con dei parametri ambientali che possono condizionare il tempo di accelerazione