Age from brain Data

Project presentation

Marco Accerenzi

Dipartimento di Fisica

Esame di Computing Methods for Exerimental Particle Physics and Data Analysis, 12 Settembre 2024

1. Introduzione
   - Cosa c'è nel codice
2. Estrazione e manipolazioni dei dati
   - Manipolazione dei dati
   - L'effetto della normalizzazione
3. Il regression model
   - Callbacks
   - Confronto dei risultati

Introduzione al progetto

L'obiettivo di questo progetto era sviluppare un modello di regressione di deep learning, addestrato per predire l'età cerebrale dei pazienti dai dati estratti dalla risonanza magnetica (MRI) dei loro cervelli.
Il modello implementato nel progetto è in grado di predire l'età dei pazienti nel set di dati con un errore quadratico medio (RMS) di circa 6 anni.
La configurazione migliore per il modello è stata un modello di deep learning con 2 hidden layers e 6 nodes per strato. Il modello si addestra e predice al meglio quando i dati sono standardizzati e normalizzati utilizzando i metodi forniti nel progetto.

Cosa c'è nel codice

Il codice sorgente di questo programma contiene diverse classi e alcuni script, al fine di fornire un framework per utilizzare i dati MRI in input dal file excel "raw" per produrre un modello addestrato in grado di fare previsioni, valutare e visualizzare tali previsioni.

Cosa c'è nel codice

Il codice sorgente di questo programma contiene diverse classi e alcuni script, al fine di fornire un framework per utilizzare i dati MRI in input dal file excel "raw" per produrre un modello addestrato in grado di fare previsioni, valutare e visualizzare tali previsioni.

- La classe ExcelData è responsabile della lettura del file excel e fornisce input adeguati per il modello.

Cosa c'è nel codice

Il codice sorgente di questo programma contiene diverse classi e alcuni script, al fine di fornire un framework per utilizzare i dati MRI in input dal file excel "raw" per produrre un modello addestrato in grado di fare previsioni, valutare e visualizzare tali previsioni.

- La classe ExcelData è responsabile della lettura del file excel e fornisce input adeguati per il modello.
- La classe RegressionModel contiene il modello stesso.

Cosa c'è nel codice

Il codice sorgente di questo programma contiene diverse classi e alcuni script, al fine di fornire un framework per utilizzare i dati MRI in input dal file excel "raw" per produrre un modello addestrato in grado di fare previsioni, valutare e visualizzare tali previsioni.

- La classe ExcelData è responsabile della lettura del file excel e fornisce input adeguati per il modello.
- La classe RegressionModel contiene il modello stesso.
- I due script "Console" forniscono un'interfaccia utente di base.
- Lo script Optimizator permette di scegliere rapidamente i valori per gli iperparametri del modello.

Cosa c'è nel codice

Il codice sorgente di questo programma contiene diverse classi e alcuni script, al fine di fornire un framework per utilizzare i dati MRI in input dal file excel "raw" per produrre un modello addestrato in grado di fare previsioni, valutare e visualizzare tali previsioni.

- La classe ExcelData è responsabile della lettura del file excel e fornisce input adeguati per il modello.
- La classe RegressionModel contiene il modello stesso.
- I due script "Console" forniscono un'interfaccia utente di base.
- Lo script Optimizator permette di scegliere rapidamente i valori per gli iperparametri del modello.
- Lo script Analysis per l'analisi.

Estrazione dei dati

I dati della risonanza magnetica per l'addestramento del modello sono stati forniti per questo progetto sotto forma di due file .xlsx, contenenti le caratteristiche di 915 pazienti (o campioni). Per ogni campione, il file contiene anche il valore reale dell'età al momento della scansione (etichetta) e il nome del gruppo che ha fornito i dati per il campione (nome del gruppo).

Estrazione dei dati

I dati della risonanza magnetica per l'addestramento del modello sono stati forniti per questo progetto sotto forma di due file .xlsx, contenenti le caratteristiche di 915 pazienti (o campioni). Per ogni campione, il file contiene anche il valore reale dell'età al momento della scansione (etichetta) e il nome del gruppo che ha fornito i dati per il campione (nome del gruppo).

- Utilizzando la funzione *read_excel* fornita dalla libreria *Pandas*, la classe crea un dataframe che rappresenta i dati nel file di input, comprese le etichette delle colonne.

Estrazione dei dati

I dati della risonanza magnetica per l'addestramento del modello sono stati forniti per questo progetto sotto forma di due file .xlsx, contenenti le caratteristiche di 915 pazienti (o campioni). Per ogni campione, il file contiene anche il valore reale dell'età al momento della scansione (etichetta) e il nome del gruppo che ha fornito i dati per il campione (nome del gruppo).

- Utilizzando la funzione *read_excel* fornita dalla libreria *Pandas*, la classe crea un dataframe che rappresenta i dati nel file di input, comprese le etichette delle colonne.
- Il metodo *to_numpy* del dataframe viene poi utilizzato per estrarre i dati numerici da fornire al modello e viene mantenuto come proprietà dell'istanza ExcelData.

Estrazione dei dati

I dati della risonanza magnetica per l'addestramento del modello sono stati forniti per questo progetto sotto forma di due file .xlsx, contenenti le caratteristiche di 915 pazienti (o campioni). Per ogni campione, il file contiene anche il valore reale dell'età al momento della scansione (etichetta) e il nome del gruppo che ha fornito i dati per il campione (nome del gruppo).

- Utilizzando la funzione *read_excel* fornita dalla libreria *Pandas*, la classe crea un dataframe che rappresenta i dati nel file di input, comprese le etichette delle colonne.
- Il metodo *to_numpy* del dataframe viene poi utilizzato per estrarre i dati numerici da fornire al modello e viene mantenuto come proprietà dell'istanza ExcelData.

Tutte le manipolazioni sui dati vengono eseguite su copie dell'array *numpy*. I campioni possono essere mescolati prima dell'addestramento per prevenire l'overfitting, consentendo di produrre previsioni genuine sugli stessi dati di test e validazione utilizzati nell'addestramento.

Manipolazione dei dati

I dati "grezzi" contengono diverse celle vuote e le varie caratteristiche hanno intervalli di valori molto differenti. Per affrontare questo problema, i dati vengono standardizzati e normalizzati.

Manipolazione dei dati

I dati "grezzi" contengono diverse celle vuote e le varie caratteristiche hanno intervalli di valori molto differenti. Per affrontare questo problema, i dati vengono standardizzati e normalizzati.

- Durante la lettura del file excel, ogni cella vuota viene riempita con il valore -9999.

Manipolazione dei dati

I dati "grezzi" contengono diverse celle vuote e le varie caratteristiche hanno intervalli di valori molto differenti. Per affrontare questo problema, i dati vengono standardizzati e normalizzati.

- Durante la lettura del file excel, ogni cella vuota viene riempita con il valore -9999.
- Tutte le colonne dell'array vengono normalizzate: ogni cella viene divisa per il valore massimo di ciascuna colonna. In questo calcolo, le celle contenenti il valore -9999 vengono escluse.

Manipolazione dei dati

I dati "grezzi" contengono diverse celle vuote e le varie caratteristiche hanno intervalli di valori molto differenti. Per affrontare questo problema, i dati vengono standardizzati e normalizzati.

- Durante la lettura del file excel, ogni cella vuota viene riempita con il valore -9999.
- Tutte le colonne dell'array vengono normalizzate: ogni cella viene divisa per il valore massimo di ciascuna colonna. In questo calcolo, le celle contenenti il valore -9999 vengono escluse.
- Il modello si allenerà per ignorare i valori -9999. Questo consente di utilizzare un set di dati più ampio rispetto a quello ottenuto eliminando tutti i campioni con una caratteristica mancante.

Manipolazione dei dati

I dati "grezzi" contengono diverse celle vuote e le varie caratteristiche hanno intervalli di valori molto differenti. Per affrontare questo problema, i dati vengono standardizzati e normalizzati.

- Durante la lettura del file excel, ogni cella vuota viene riempita con il valore -9999.
- Tutte le colonne dell'array vengono normalizzate: ogni cella viene divisa per il valore massimo di ciascuna colonna. In questo calcolo, le celle contenenti il valore -9999 vengono escluse.
- Il modello si allenerà per ignorare i valori -9999. Questo consente di utilizzare un set di dati più ampio rispetto a quello ottenuto eliminando tutti i campioni con una caratteristica mancante.
- La normalizzazione e la standardizzazione riducono notevolmente il tempo necessario per l'addestramento. Le istanze del modello addestrate con dati normalizzati fanno previsioni mediamente più accurate sui dati rispetto a quelle addestrate senza normalizzazione.

L'effetto della normalizzazione 1

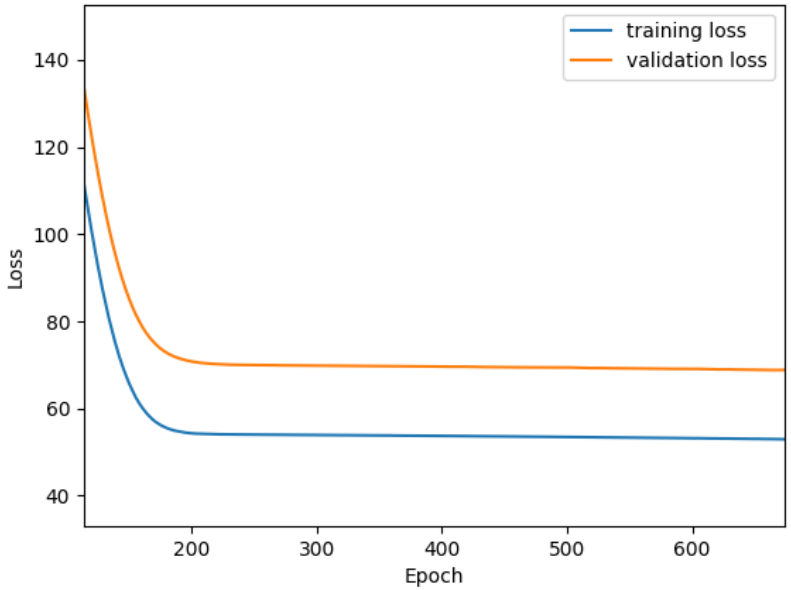


Figure: Grafico delle prime epoche di addestramento utilizzando dati normalizzati.

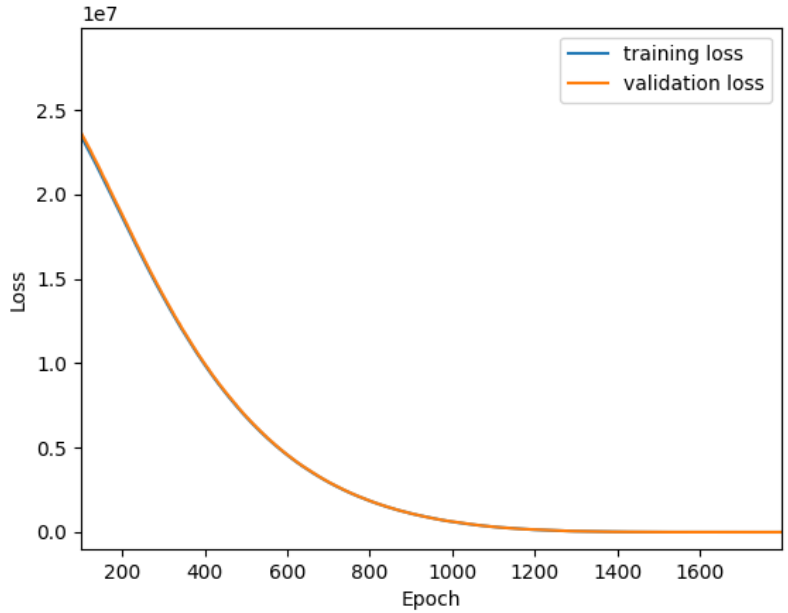


Figure: Grafico delle prime epoche di addestramento utilizzando dati grezzi.

Il regression model

La classe del modello di regressione utilizza i moduli *models*, *layers* e *callbacks* della libreria *keras*. Implementa diversi metodi:

Il regression model

La classe del modello di regressione utilizza i moduli *models*, *layers* e *callbacks* della libreria *keras*. Implementa diversi metodi:

- Il metodo Compile_Model definisce l'istanza del modello, imposta i livelli e i nodi del modello e stabilisce la forma dei dati di input. Dopo la chiamata del metodo Compile_Model, il modello viene compilato ed è pronto per l'addestramento.

Il regression model

La classe del modello di regressione utilizza i moduli *models*, *layers* e *callbacks* della libreria *keras*. Implementa diversi metodi:

- Il metodo Compile_Model definisce l'istanza del modello, imposta i livelli e i nodi del modello e stabilisce la forma dei dati di input. Dopo la chiamata del metodo Compile_Model, il modello viene compilato ed è pronto per l'addestramento.
- Il metodo Start_Training allena il modello su alcuni dati, utilizzando l'algoritmo di apprendimento supervisionato fornito dalla libreria *keras*. Il numero di epoche e altri parametri dell'addestramento possono essere forniti dall'utente.

Il regression model

La classe del modello di regressione utilizza i moduli *models*, *layers* e *callbacks* della libreria *keras*. Implementa diversi metodi:

- Il metodo Compile_Model definisce l'istanza del modello, imposta i livelli e i nodi del modello e stabilisce la forma dei dati di input. Dopo la chiamata del metodo Compile_Model, il modello viene compilato ed è pronto per l'addestramento.
- Il metodo Start_Training allena il modello su alcuni dati, utilizzando l'algoritmo di apprendimento supervisionato fornito dalla libreria *keras*. Il numero di epoche e altri parametri dell'addestramento possono essere forniti dall'utente.
- I metodi Plot_History e Save_Model sono pensati per essere utilizzati dopo Start_Training, per salvare la cronologia dell'addestramento e il modello stesso.

Il regression model

La classe del modello di regressione utilizza i moduli *models*, *layers* e *callbacks* della libreria *keras*. Implementa diversi metodi:

- Il metodo Compile_Model definisce l'istanza del modello, imposta i livelli e i nodi del modello e stabilisce la forma dei dati di input. Dopo la chiamata del metodo Compile_Model, il modello viene compilato ed è pronto per l'addestramento.
- Il metodo Start_Training allena il modello su alcuni dati, utilizzando l'algoritmo di apprendimento supervisionato fornito dalla libreria *keras*. Il numero di epoche e altri parametri dell'addestramento possono essere forniti dall'utente.
- I metodi Plot_History e Save_Model sono pensati per essere utilizzati dopo Start_Training, per salvare la cronologia dell'addestramento e il modello stesso.
- Durante l'inizializzazione di un'istanza del modello è possibile utilizzare la funzione *load_model* per caricare un modello già addestrato: questo modello può essere utilizzato solo per l'inferenza e

Callbacks

Per ottimizzare la fase di addestramento vengono utilizzati diversi callback:

Callbacks

Per ottimizzare la fase di addestramento vengono utilizzati diversi callback:

- Early stopping: Questo callback interrompe l'addestramento se, dopo un numero di epoche successive impostato dall'utente, la funzione di perdita non è diminuita. Permette di fermare l'addestramento prima di incorrere in overfitting. Dopo l'interruzione, ripristina i pesi con le migliori prestazioni.

Callbacks

Per ottimizzare la fase di addestramento vengono utilizzati diversi callback:

- Early stopping: Questo callback interrompe l'addestramento se, dopo un numero di epoche successive impostato dall'utente, la funzione di perdita non è diminuita. Permette di fermare l'addestramento prima di incorrere in overfitting. Dopo l'interruzione, ripristina i pesi con le migliori prestazioni.
- ModelCheckpoint: Questo callback salva vari "checkpoint" durante l'addestramento. Permette di mantenere un backup del progresso dell'addestramento.

Callbacks

Per ottimizzare la fase di addestramento vengono utilizzati diversi callback:

- Early stopping: Questo callback interrompe l'addestramento se, dopo un numero di epoche successive impostato dall'utente, la funzione di perdita non è diminuita. Permette di fermare l'addestramento prima di incorrere in overfitting. Dopo l'interruzione, ripristina i pesi con le migliori prestazioni.
- ModelCheckpoint: Questo callback salva vari "checkpoint" durante l'addestramento. Permette di mantenere un backup del progresso dell'addestramento.
- ReduceLROnPlateau: Questo callback riduce il *learning rate* del modello se l'addestramento raggiunge un plateau (significa se per un certo numero di epoche la funzione di perdita smette di diminuire). Riduce la quantità con cui i pesi del modello vengono cambiati durante l'addestramento, permettendo un ulteriore addestramento dopo aver raggiunto i plateaux.

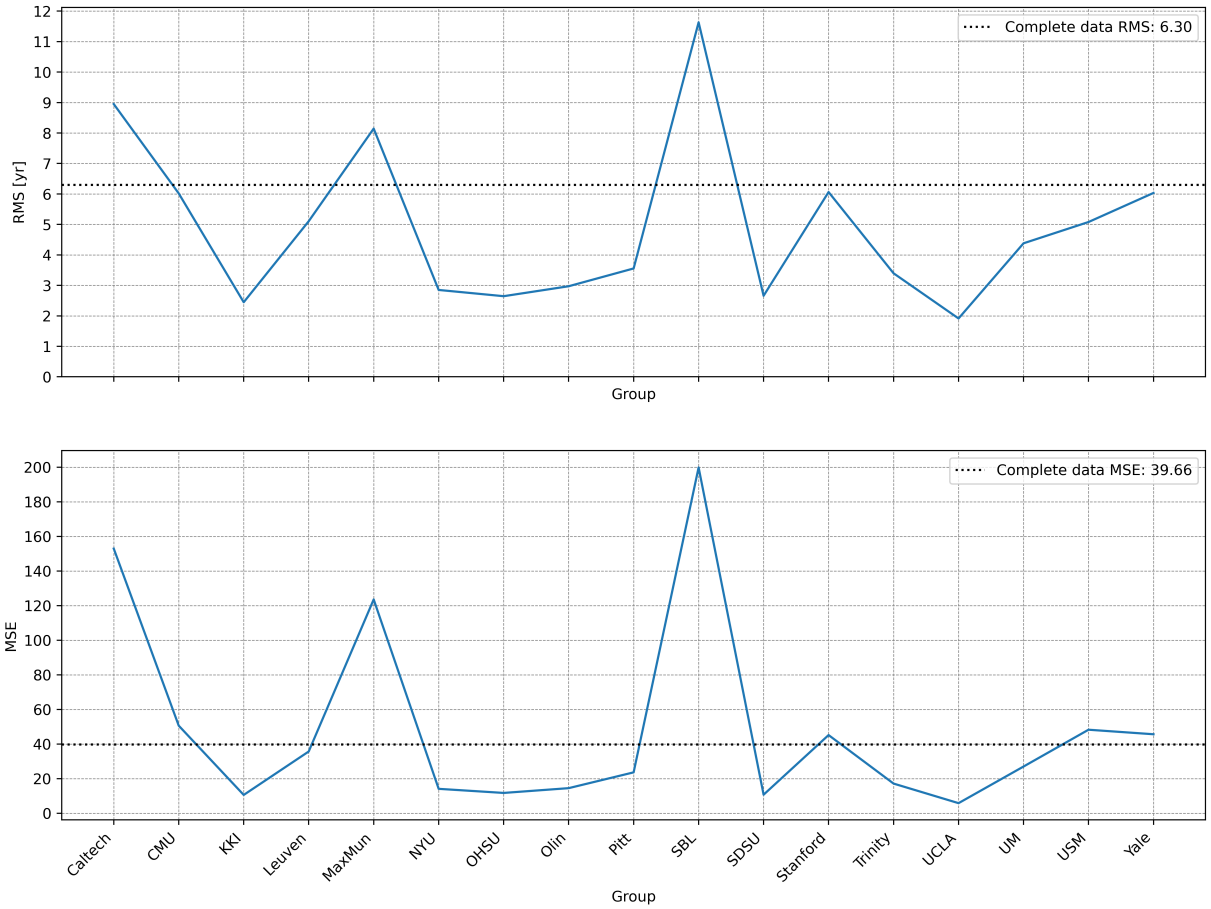
Callbacks

Per ottimizzare la fase di addestramento vengono utilizzati diversi callback:

- Early stopping: Questo callback interrompe l'addestramento se, dopo un numero di epoche successive impostato dall'utente, la funzione di perdita non è diminuita. Permette di fermare l'addestramento prima di incorrere in overfitting. Dopo l'interruzione, ripristina i pesi con le migliori prestazioni.
- ModelCheckpoint: Questo callback salva vari "checkpoint" durante l'addestramento. Permette di mantenere un backup del progresso dell'addestramento.
- ReduceLROnPlateau: Questo callback riduce il *learning rate* del modello se l'addestramento raggiunge un plateau (significa se per un certo numero di epoche la funzione di perdita smette di diminuire). Riduce la quantità con cui i pesi del modello vengono cambiati durante l'addestramento, permettendo un ulteriore addestramento dopo aver raggiunto i plateaux.

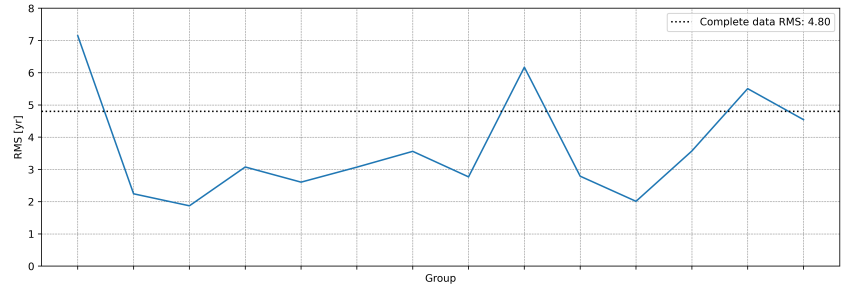
Confronto dei risultati

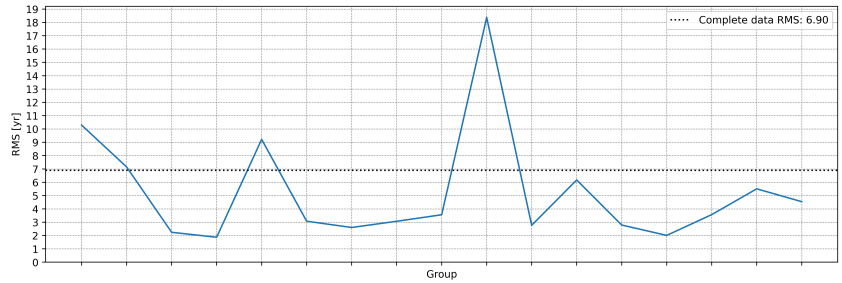
Le prestazioni del modello sui diversi gruppi sono confrontate nella figura sottostante: l'errore di previsione per i dati dei gruppi "Caltech", "MaxMun" e "SBL" è nettamente superiore alla media.

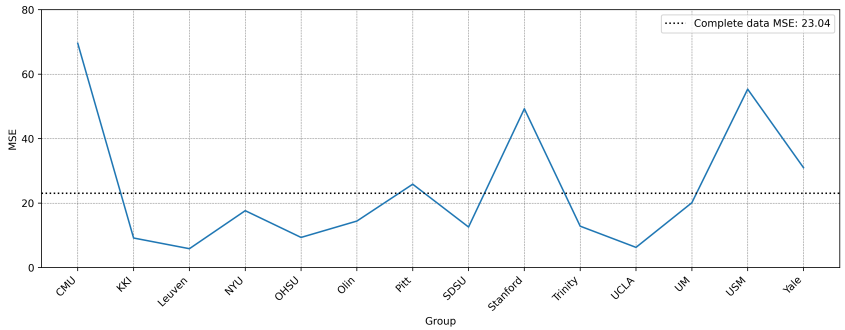


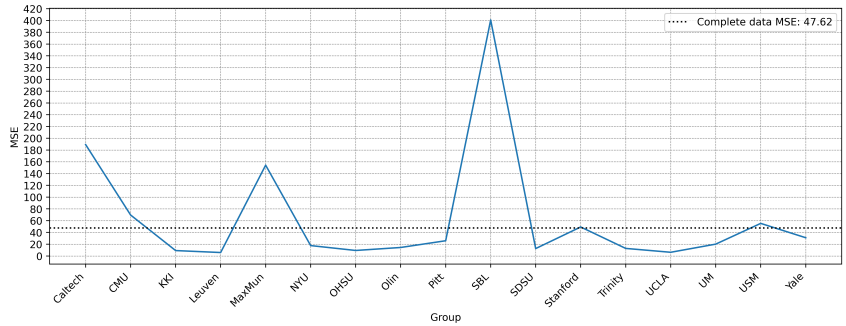
Confronto dei risultati 2

Ripetere l'addestramento su un set di dati ridotto, rimuovendo i tre gruppi di dati outlier, non migliora significativamente le prestazioni del modello: il modello addestrato si comporta in modo simile sui gruppi di dati rimanenti ma in modo significativamente peggiore per i tre outlier.









Grazie per l'attenzione!

Git e il repository

Il progetto utilizza un repository pubblico su GitHub.
Ha un ramo principale e tre rami attivi: develop, feature/docs e feature/presentation.

Documentazione

La documentazione del progetto è stata generata tramite Sphinx, utilizzando la sua funzionalità *autodoc* per estrarre le docstring dal codice sorgente.
Il repository pubblico su GitHub consente di distribuire la documentazione del progetto attraverso ReadTheDocs e le Pages di GitHub.