Tommaso Martinelli

Tesina di
Signal Processing and Optimization for Big Data
Prof. Paolo Banelli

Esplorazione delle varianti di Lasso: Gradient Descent, ADMM e ADMM distribuito a confronto

Perugia, Anno Accademico 2023/2024
Università degli Studi di Perugia
Corso di laurea magistrale in Ingegneria Informatica e Robotica
Curriculum Data Science
Dipartimento di Ingegneria

A.D. 1308
unipg
DIPARTIMENTO
DI INGEGNERIA

0. Indice

1. Introduzione

Lo studio in questione si propone di esplorare e confrontare diverse implementazioni dell'algoritmo LASSO, in contesti differenti.

1.1 Struttura del Documento

Il presente documento è organizzato nei seguenti capitoli:

- Nel **Capitolo 2**, verrà condotta un'analisi teorica della regressione LASSO e delle sue implementazioni, includendo una descrizione delle varianti come Gradient Descent, ADMM e ADMM distribuito.
- Il **Capitolo 3** si concentrerà sull'implementazione delle diverse versioni di LASSO, illustrando le tecnologie utilizzate e i dataset considerati.
- Nel **Capitolo 4**, verranno presentati gli esperimenti e i risultati ottenuti, con un focus sul confronto delle performance tra le implementazioni in diversi contesti.
- Infine, nel **Capitolo 5**, saranno fornite le conclusioni dello studio e possibili sviluppi futuri.

1.2 Contesto generale

La regressione LASSO (Least Absolute Shrinkage and Selection Operator) rappresenta una metodologia fondamentale nell'ambito della modellazione statistica e del machine

learning. Essa consente la gestione dell'overfitting e la selezione delle variabili, fornendo un approccio efficace per ottenere modelli robusti. Con l'avvento di algoritmi sempre più complessi, è emersa la sfida dell'overfitting, che si verifica quando un modello si adatta eccessivamente ai dati di addestramento, catturando rumore anziché relazioni reali. La regressione LASSO si presenta come una soluzione a questo problema, introducendo una penalità sulla somma degli apporti delle variabili nel modello. La sua peculiarità è dunque quella di applicare una regolarizzazione di tipo L1, promuovendo la sparsità del modello. Ciò significa che la LASSO favorisce la presenza di coefficienti nulli, consentendo di selezionare in modo automatico le variabili più rilevanti per la predizione.

1.3 Obiettivo dello studio

L'obiettivo principale è implementare e testare diverse versioni dell'algoritmo LASSO, analizzandone le prestazioni in scenari di regressione differenti. Le varianti prese in considerazione includono l'utilizzo del *Gradient Descent* (GD), l'*Alternating Direction Method of Multipliers* (ADMM), e una versione distribuita dell'*ADMM*, splittando i campioni. Questo approccio multifase consentirà di confrontare le performance di ciascuna implementazione, valutando i loro vantaggi e svantaggi in differenti contesti, tramite l'utilizzo di vari dataset.

2. Analisi teorica

In questo capitolo, viene svolta un'analisi approfondita del problema dal punto di vista teorico e matematico, allo scopo di comprendere gli algoritmi che saranno successivamente implementati. Si fornisce una panoramica dettagliata della regressione LASSO, introducendo il concetto di regolarizzazione L1 e esaminando le soluzioni

teoriche per il problema di ottimizzazione associato. Inoltre, vengono esplorate le varianti dell'algoritmo, fornendo le basi concettuali e matematiche necessarie per una comprensione completa delle implementazioni pratiche che saranno esaminate nei capitoli successivi.

2.1 Regressione LASSO

La LASSO è una tecnica di regressione che introduce una regolarizzazione L1 al problema di ottimizzazione. Ciò consiste nell'aggiunta di un termine di penalità proporzionale alla somma dei valori assoluti dei coefficienti del modello, favorendo la presenza di coefficienti nulli, semplificando il modello e prevenendo l'overfitting. Il problema di ottimizzazione associato alla regressione LASSO coinvolge la minimizzazione di una funzione di costo, che comprende sia il termine di regressione che la penalità L1.

$$\min_{\underline{x}} \frac{1}{2} \left\| \underline{y} - A\underline{x} \right\|_2^2 + \lambda \left\| \underline{x} \right\|_1$$

L'equazione sopra rappresenta il problema di ottimizzazione associato alla regressione LASSO. Il termine $\underline{x}$ rappresenta il vettore dei coefficienti del modello, $\underline{y}$ è il vettore delle risposte osservate, A è la matrice delle features e λ è il parametro di regolarizzazione.

2.1.1 Gradient Descent

Il Gradient Descent è un algoritmo di ottimizzazione ampiamente utilizzato nella minimizzazione di funzioni di costo. Nell'ambito della regressione LASSO, il GD viene applicato per risolvere il problema di ottimizzazione precedentemente mostrato e ha l'obiettivo di aggiornare i pesi del modello, nella direzione opposta del gradiente della funzione costo.

$$\underline{x}^{(k+1)} = \underline{x}^{(k)} - \gamma \nabla \left(\frac{1}{2} \left\| \underline{y} - A\underline{x}^{(k)} \right\|_2^2 + \lambda \left\| \underline{x}^{(k)} \right\|_1 \right)$$

La discesa del gradiente dipende dal valore dello step-size γ. Il termine regolarizzante risulta essere non derivabile, in quanto ha punti angolosi, quindi è necessario introdurre il concetto di sub-gradiente.

$$[sub\nabla f(x_0)]_i = \begin{cases} \frac{\partial f(x_0)}{\partial x_i}, & \text{se differenziabile in } x_0 \\ [\frac{\partial f(x_0^-)}{\partial x_i}, \frac{\partial f(x_0^+)}{\partial x_i}], & \text{se non differenziabile in } x_0 \end{cases}$$

Che applicato al problema in questione risulta:

$$sub\nabla \underline{\Phi} = \begin{cases} \gamma\lambda + (u_j - x_j) & \text{se } u_j > 0 \\ -\gamma\lambda + (u_j - x_j) & \text{se } u_j < 0 \\ [-\gamma\lambda + (u_j - x_j), \gamma\lambda + (u_j - x_j)] & \text{se } u_j = 0 \end{cases}$$

$$\Rightarrow \begin{cases} u_j = x_j - \gamma\lambda & \text{se } x_j > \gamma\lambda \\ u_j = x_j + \gamma\lambda & \text{se } x_j < -\gamma\lambda \\ u_j = 0 & \text{se } -\gamma\lambda < x_j < \gamma\lambda \end{cases}$$

Questo prende il nome di $Soft - Thresholding$ e permette di calcolare il gradiente della funzione costo in esame. La convergenza viene raggiunta quando la variazione è inferiore ad una quantità che indica la tolleranza.

2.2 Alternating Direction Method of Multipliers (ADMM)

L'Alternating Direction Method of Multipliers (ADMM) è un approccio di ottimizzazione che si basa sulla decomposizione del problema in sottoproblemi più gestibili. Appartiene alla famiglia degli algoritmi di ottimizzazione primal-dual. Per applicarlo alla Lasso è per prima cosa necessario riformulare il problema, andando ad introdurre la slack variable $\underline{z}$ per separare l'ottimizzazione della funzione originale da quella della regolarizzazione. La formulazione generale è dunque:

$$\underset{\underline{x}}{argmin}\ \frac{1}{2}\left\|A\underline{x} - \underline{y}\right\|_2^2 + \lambda\left\|\underline{z}\right\|_1, \quad s.t.\ \ \underline{x} - \underline{z} = 0$$

ADMM suddivide il problema in 3 sotto-problemi, le cui soluzioni verranno ricercate in maniera alternata fino a convergenza. Viene mostrata la formulazione standard scalata:

$$\underline{x}^{(k+1)} = \underset{\underline{x}}{argmin}\{\underbrace{f(\underline{x})}_{\text{Convex}} + \underbrace{\frac{\rho}{2}\left\|A\underline{x} + B\underline{z}^{(k)} - \underline{c} + \underline{u}^{(k)}\right\|_2^2}_{\text{Strongly Convex}}\}$$

$$\underline{z}^{(k+1)} = \underset{\underline{z}}{argmin}\{\underbrace{g(\underline{z})}_{\text{Convex}} + \underbrace{\frac{\rho}{2}\left\|A\underline{x}^{(k+1)} + B\underline{z} - \underline{c} + \underline{u}^{(k)}\right\|_2^2}_{\text{Strongly Convex}}\}$$

$$\underline{u}^{(k+1)} = \underline{u}^{(k)} + \underbrace{(A\underline{x}^{(k+1)} + B\underline{z}^{(k+1)} - \underline{c})}_{\underline{r}^{(k+1)}} = \underline{u}^{(0)} + \sum_{i=1}^{k+1}\underline{r}^{(i)}$$

Che nel caso della LASSO diventa:

$$\underline{x}^{(k+1)} = \underset{\underline{x}}{argmin}\{\frac{1}{2}\left\|A\underline{x} - \underline{y}\right\|_2^2 + \frac{\rho}{2}\left\|\underline{x} - \underline{z}^{(k)} + \underline{u}^{(k)}\right\|_2^2\}$$

$$\underline{z}^{(k+1)} = \underset{\underline{z}}{argmin}\{\lambda\left\|\underline{z}\right\|_1 + \frac{\rho}{2}\left\|\underline{x}^{(k+1)} - \underline{z} + \underline{u}^{(k)}\right\|_2^2\}$$

$$\underline{u}^{(k+1)} = \underline{u}^{(0)} + \sum_{i=1}^{k+1}\left\|\underline{x}^{(i)} - \underline{z}^{(i)}\right\|_2$$

E conduce ad una soluzione in forma chiusa:

$$\underline{x}^{(k+1)} = (A^TA + \rho I)^{-1}(A^T\underline{y} + \rho(\underline{z}^{(k)} - \underline{u}^{(k)}))$$

$$\underline{z}^{(k+1)} = S_{\frac{\lambda}{\rho}}(\underline{x}^{(k+1)} + \underline{u}^{(k)})$$

$$\underline{u}^{(k+1)} = \underline{u}^{(k)} + \underline{x}^{(k+1)} - \underline{z}^{(k+1)}$$

I parametri ρ e λ rivestono un ruolo critico e la loro scelta richiede attente valutazioni per bilanciare la velocità di convergenza e la robustezza. L'utilizzo dell'ADMM porta svariati vantaggi, come la suddivisione in sotto-problemi di più facile gestione, buona stabilità e convergenza a valori ottimali in tempi generalmente più brevi ed inoltre apre la strada a possibili implementazioni distribuite.

2.3 ADMM Distribuito

La versione distribuita di ADMM estende l'ADMM classico per affrontare problemi di dimensioni più grandi distribuendo il lavoro tra diversi agenti. L'ottimizzazione al consenso è implementata suddividendo gli esempi tra gli agenti, ognuno con copie delle stesse variabili di ottimizzazione. Il problema è quindi diviso tra i dati e risolto in modo collaborativo. Questo tipo di approccio è possibile quando la funzione obiettivo è additiva e si può quindi scomporre l'ottimizzazione. La formulazione standard di una ottimizzazione al consenso è:

$$\underset{\underline{z},\underline{x}_1,...,\underline{x}_N}{min} \sum_{i=1}^{N} f_i(\underline{x}) + g(\underline{z}) \quad s.t. \;\; \underline{x}_i - \underline{z} = \underline{0}, \;\; i = 1, ..., N$$

Andando a formulare al consenso il problema trattato si ottiene:

$$\underset{\underline{x}}{min} \;\; \sum_{i=1}^{N} \left\| A_i \underline{x} - \underline{y}_i \right\|_2^2 + \lambda \left\| \underline{z} \right\|_1 \quad s.t. \;\; \underline{x}_i - \underline{z} = \underline{0}, \;\; i = 1, ..., N$$

Quindi è possibile distribuire l'aggiornamento delle $\underline{x}$ tra i vari agenti, che calcoleranno la soluzione sulla loro porzione di dati. Il problema viene risolto tramite l'utilizzo dell'ADMM scalato.

$$\underline{x}_i^{(k+1)} = \underset{\underline{x}}{argmin}\{\left\| A_i \underline{x}_i - \underline{y}_i \right\|_2^2 + \frac{\rho}{2} \left\| \underline{x}_i - \underline{z}^{(k)} + \underline{u}_i^{(k)} \right\|_2^2\}$$

$$\underline{z}^{(k+1)} = Prox_{\frac{\lambda}{N\rho}\|\cdot\|}\{\hat{\underline{x}}^{(k+1)} + \hat{\underline{u}}^{(k)}\} = S_{\frac{\lambda}{N\rho}}(\hat{\underline{x}}^{(k+1)} + \hat{\underline{u}}^{(k)})$$

$$\underline{u}_i^{(k+1)} = \underline{u}_i^{(k)} + \underline{x}_i^{k+1} - \underline{z}^{k+1}$$

Per il calcolo della $\underline{z}$ è necessario un *Fusion Center*, in cui viene aggiornata la variabile dopo che tutti gli agenti hanno comunicato i loro aggiornamenti locali. L'algoritmo iterativo può essere riscritto in forma chiusa:

$$\underline{x}_i^{(k+1)} = (A_i^T A_i + \frac{\rho}{2} I)^{-1} (A_i^T \underline{y}_i + \frac{\rho}{2} (\underline{z}^{(k)} - \underline{u}_i^{(k)}))$$

$$\underline{z}^{(k+1)} = S_{\frac{\lambda}{N\rho}}(\hat{\underline{x}}^{(k+1)} + \hat{\underline{u}}^{(k)})$$

$$\underline{u}^{(k+1)} = \underline{u}^{(k)} + \underline{x}^{k+1} - \underline{z}^{k+1}$$

Questa tecnica si presta bene a dataset di grandi dimensioni e inoltre consente di gestire situazioni in cui i dati sono distribuiti.

3. Implementazione

Le tecniche precedentemente descritte sono state implementate per eseguire poi vari test in contesti differenti.

3.1 Tecnologie utilizzate

L'implementazione della LASSO con 3 diverse tecniche di ottimizzazione è stata realizzata utilizzando il linguaggio di programmazione *Python*, che grazie alle sue librerie *numpy* e *pandas* ha reso possibile eseguire le operazioni descritte nella sezione precedente. Lo studio è stato effettuato utilizzando un *Jupyter Notebook* che ha permesso sia di implementare le tecniche di ottimizzazione, sia di testarle su vari dataset.

3.2 Implementazione delle Tre Versioni

Le 3 versioni LASSO sono state implementate tramite la creazione di una classe *LassoReg*, che ha al suo interno una funzione fit, a cui va specificata la tecnica di ottimizzazione scelta e i parametri da utilizzare. Sono state sviluppate le tecniche descritte in precedenza, ma la struttura dell'implementazione lascia spazio all'inserimento di nuove strategie di ottimizzazione.

3.3 Dataset Utilizzati

Il modello sviluppato è stato testato su tre dataset differenti che verranno, in questa sezione, descritti brevemente.

3.3.1 Primo dataset sintetico

Il primo dataset è stato generato per rappresentare un problema di regressione. Esso è composto da 300 campioni, caratterizzati da 3 variabili numeriche, con valori compresi tra 0 e 1, e una variabile categorica binaria. L'etichetta ha una dipendenza lineare con alcune delle features ed inoltre gli è stato aggiunto del rumore, per rendere più realistici i dati. Sono mostrate le distribuzioni delle features e dell'etichetta nelle immagini sottostanti. Questi primi dati sono volutamente semplici, per essere utilizzati come primo test per tutti i modelli.

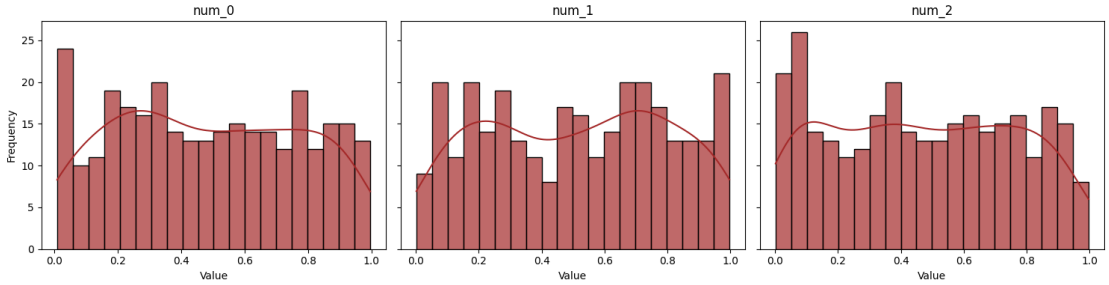


Figura 3.1: Features del primo dataset

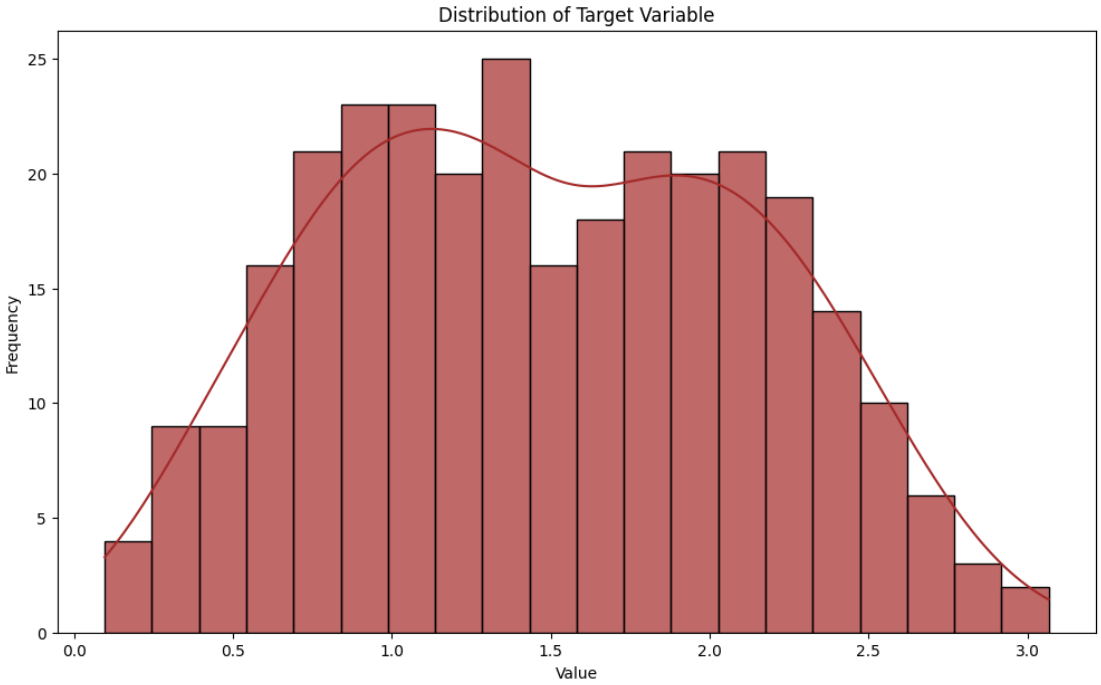


Figura 3.2: Label del primo dataset

3.3.2 Secondo dataset sintetico

Anche il secondo insieme di dati è stato generato in fase di test. Il dataset è composto da 2000 campioni con 8 features numeriche e 2 categoriche. Le feature numeriche sono generate con distribuzioni diverse, tra cui normale, uniforme, esponenziale, poisson, gamma, logistica, chi-quadro e potenza. Le feature categoriche assumono valori da un insieme di scelte discrete di 4 valori. Le etichette sono generate utilizzando pesi veri per le feature numeriche e categoriche, oltre a un termine costante. Alcuni termini sono influenzati da funzioni non lineari delle feature numeriche, come il seno, l'esponenziale, e il coseno moltiplicato per una feature categorica. Inoltre, è aggiunto un termine di errore casuale per simulare la variabilità nei dati. Questo dataset è stato progettato per rappresentare una situazione più complessa con diverse distribuzioni di feature e relazioni non lineari tra feature e etichette.

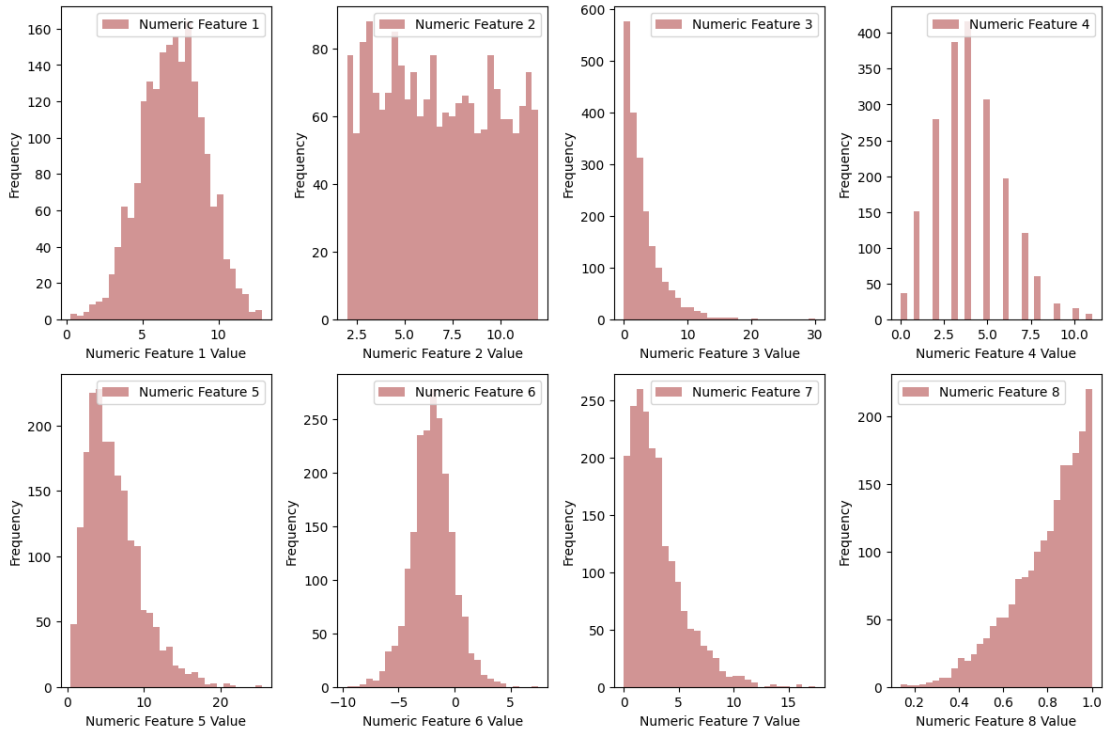


Figura 3.3: Features del secondo dataset

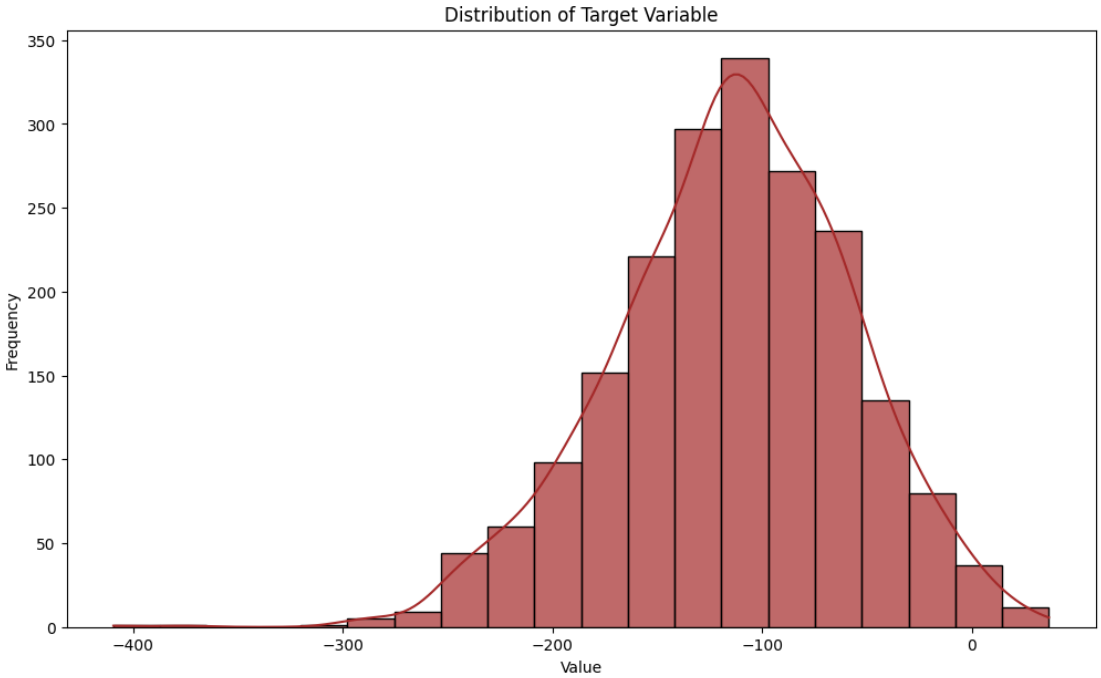


Figura 3.4: Label del secondo dataset

3.3.3 Dataset reale: "Concrete Compressive Strength"

Come terzo dataset ne è stato scelto uno reale, che può essere consultato al link: Concrete Compressive Strength Dataset. Questo insieme di dati è stato originariamente raccolto per studiare la resistenza a compressione del calcestruzzo e comprende diverse proprietà del calcestruzzo, come età, cemento, sabbia, pietra frantumata, acqua e altri ingredienti. Il dataset contiene un totale di 1030 istanze, ognuna con 8 attributi diversi. La variabile target, in questo caso, è la resistenza a compressione del calcestruzzo. Questo dataset reale è stato scelto per fornire una valutazione delle prestazioni del modello in un contesto pratico, in cui le relazioni tra le variabili possono essere più complesse rispetto ai dataset sintetici. Per questo dataset sono state richieste delle semplici operazioni di preprocessing, svolte in un Notebook separato, per rendere i dati utilizzabili per questa analisi.

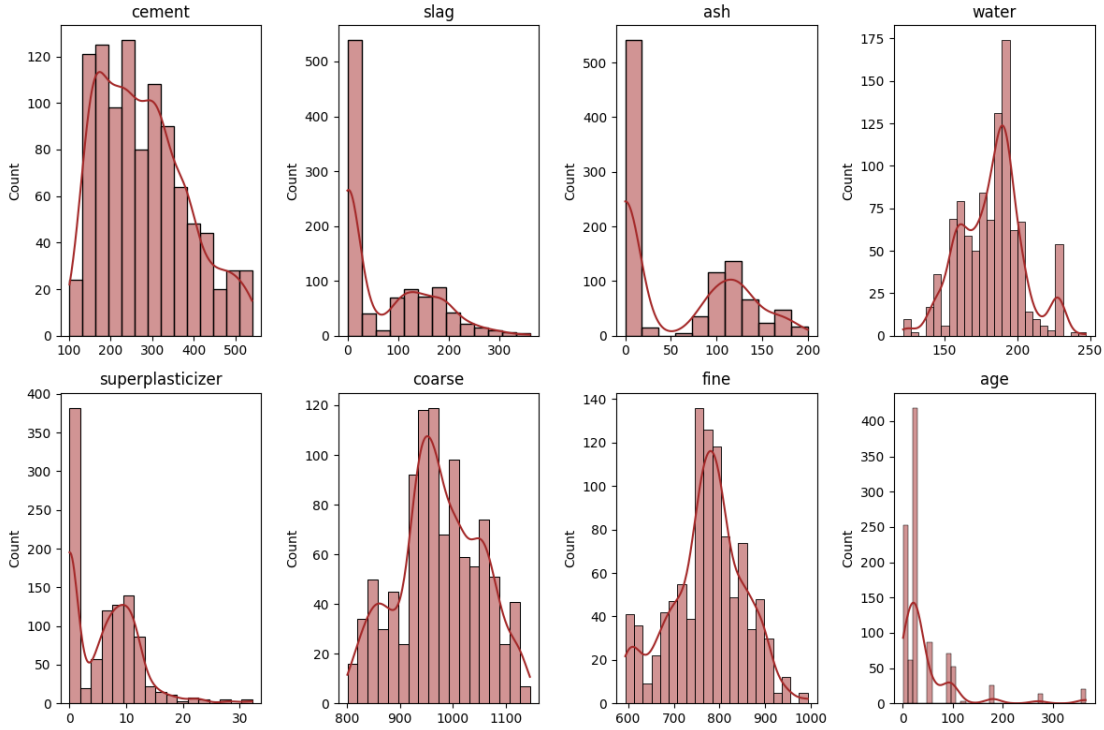


Figura 3.5: Features del terzo dataset

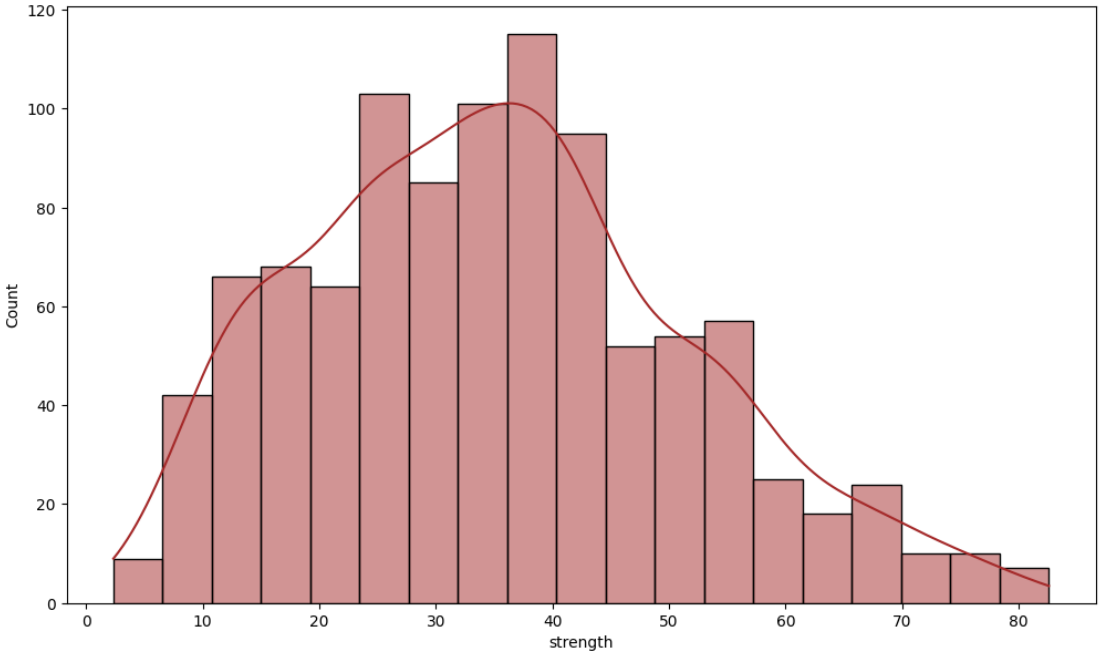


Figura 3.6: Label del terzo dataset

4. Esperimenti e Risultati

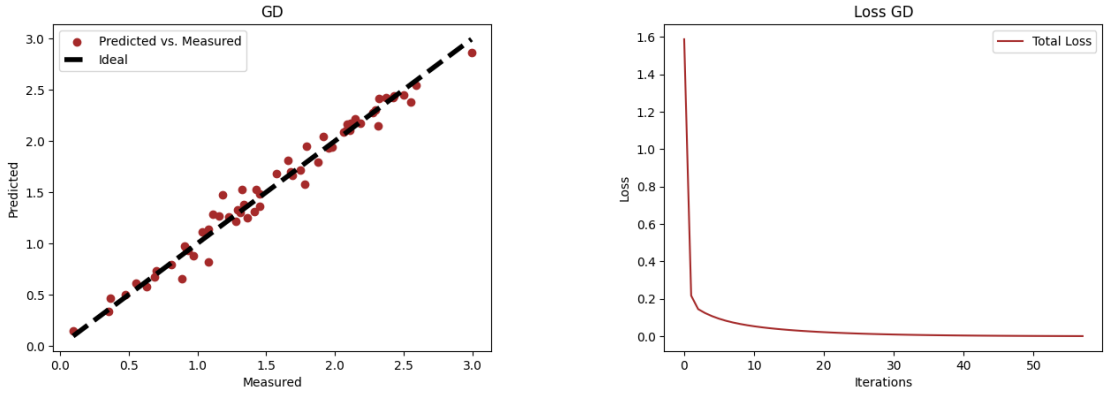
Su tutti i dataset sono state addestrate e testate le 3 versioni di LASSO implementate. Sono stati effettuati dei confronti tra i modelli e una valutazione su come la scelta dei parametri influisce sulle prestazioni del modello. Per le valutazioni è stata usata la metrica R^2, il tempo di esecuzione e il numero di iterazioni. Inoltre sono mostrati dei grafici riguardanti le predizioni e la discesa della loss o dei residui.

4.1 Dataset 1

Per ogni algoritmo sono stati valutati diversi valori per i parametri, al fine di trovare la configurazione più performante e comprendere l'impatto delle scelte sul dataset trattato. Al termine verrà effettuato un confronto tra le tre versioni, mostrandone tempi, iterazioni e risultati.

4.1.1 Gradient Descent

La versione della discesa al gradiente è stata testata utilizzando uno *step size* di 0.5 ed una $l1$ *penalty* di 10. Valori minori per lo *step size* portano a soluzioni peggiori e tempi più lunghi, mentre al variare di $l1$ *penalty* vi sono differenze minime sia in termini di metriche che di tempi di esecuzione.



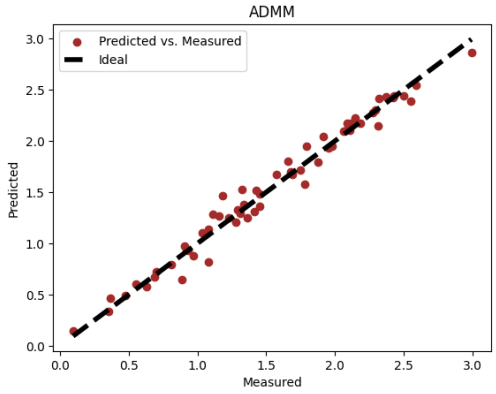
(a) Predizioni GD

(b) Loss GD

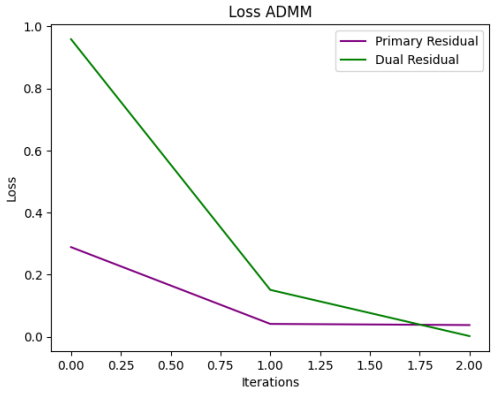
Figura 4.1: Gradient Descent sul Dataset 1

4.1.2 ADMM

Il metodo ADMM è stato utilizzato sul primo dataset con parametro *step size* 0.5 e $l1$ *penalty* 0.1, in questo caso aumentando la penalità si nota un rallentamento della convergenza, sia in termini di tempi, che di numero di iterazioni necessarie.



(a) Predizioni ADMM

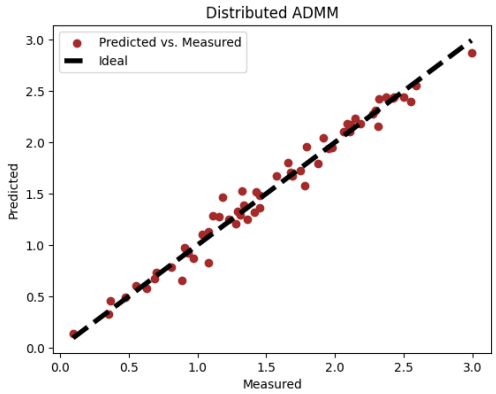


(b) Residui ADMM

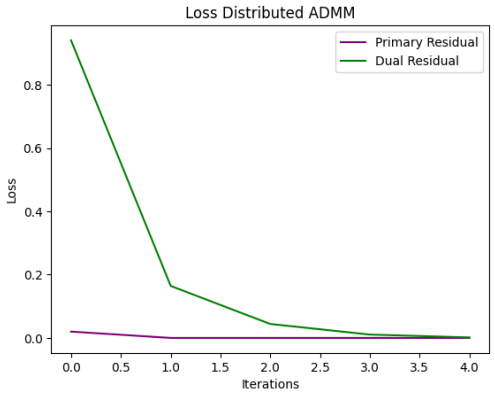
Figura 4.2: ADMM sul Dataset 1

4.1.3 ADMM distribuito

Per la versione distribuita sono stati effettuati test anche sull'impatto del numero di agenti. I parametri utilizzati sono uno *step size* di 0.5 e una $l1$ *penalty* di 0.1. Aumentando il numero di agenti aumentano i tempi per il raggiungimento della convergenza, ma i valori della metrica variano di poco. Perciò è stata testata una versione con 20 agenti.



(a) Predizioni ADMM distribuito



(b) Residui ADMM distribuito

Figura 4.3: ADMM distribuito sul Dataset 1

4.1.4 Confronto

Le versioni descritte sono confrontate nella seguente tabella

	R2	Tempo (s)	Iterazioni
GD	0.9763	0.0070	58
ADMM	0.9766	0.0012	3
ADMM Distribuito	0.9770	0.0113	5

Tabella 4.1: Confronto sul Dataset 1

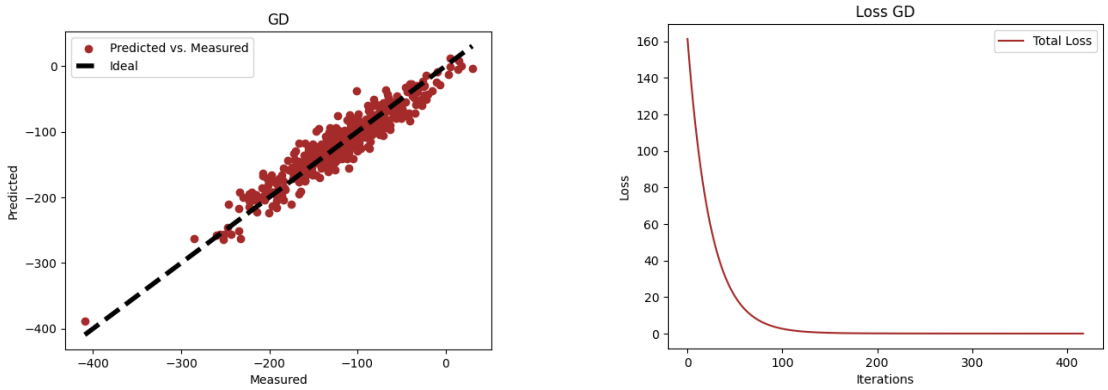
I risultati osservati mostrano differenze minime per quanto riguarda la metrica, ma si può notare che il tempo di esecuzione e il numero di iterazioni variano maggiormente.

4.2 Dataset 2

L'approccio utilizzato per il primo dataset è stato replicato anche per il secondo, con l'intenzione di ottenere ulteriori indicazioni sui modelli implementati ed eventualmente valutare se i risultati sono concordi con quelli visti precedentemente

4.2.1 Gradient Descent

Il Gradient descent ha mostrato migliori risultati con valori maggiori di *step size* e con il parametro $l1\ penalty$ settato a 0.1. Il numero di iterazioni richieste è piuttosto alto, ciò è diretta conseguenza della dimensionalità del dataset.



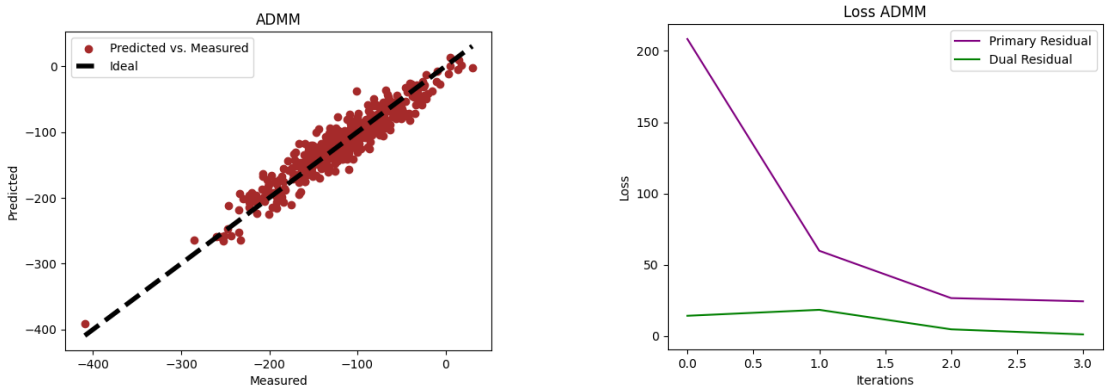
(a) Predizioni GD

(b) Loss GD

Figura 4.4: Gradient Descent sul Dataset 2

4.2.2 ADMM

Una versione di ADMM centralizzato è stata sviluppata con parametri di *step size*=0.1 e di $l1$ $penalty$=10. L'algoritmo impiega 4 iterazioni a convergere.



(a) Predizioni ADMM

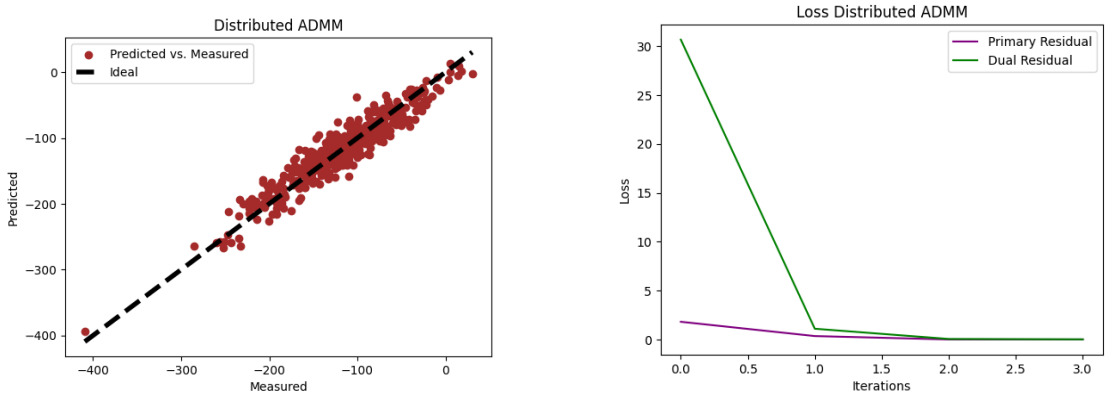
(b) Residui ADMM

Figura 4.5: ADMM sul Dataset 2

4.2.3 ADMM distribuito

Per la versione distribuita, come in precedenza, sono state valutate le prestazioni anche al variare del numero degli agenti. Fissando lo *step size* a 0.1 e $l1$ $penalty$ a 1, si registrano

prestazioni differenti in base al numero degli agenti. La metrica non varia notevolmente aumentando il numero di agenti, ma crescono solo tempo e numero di iterazioni. Il problema si presta bene ad essere distribuito.



(a) Predizioni ADMM distribuito

(b) Residui ADMM distribuito

Figura 4.6: ADMM distribuito sul Dataset 2

4.2.4 Confronto

Le versioni sono messe a confronto, riportando i risultati nella tabella sottostante.

	R2	Tempo (s)	Iterazioni
GD	0.8958	0.2186	418
ADMM	0.8960	0.0040	4
ADMM Distribuito	0.8961	0.0110	4

Tabella 4.2: Confronto sul Dataset 1

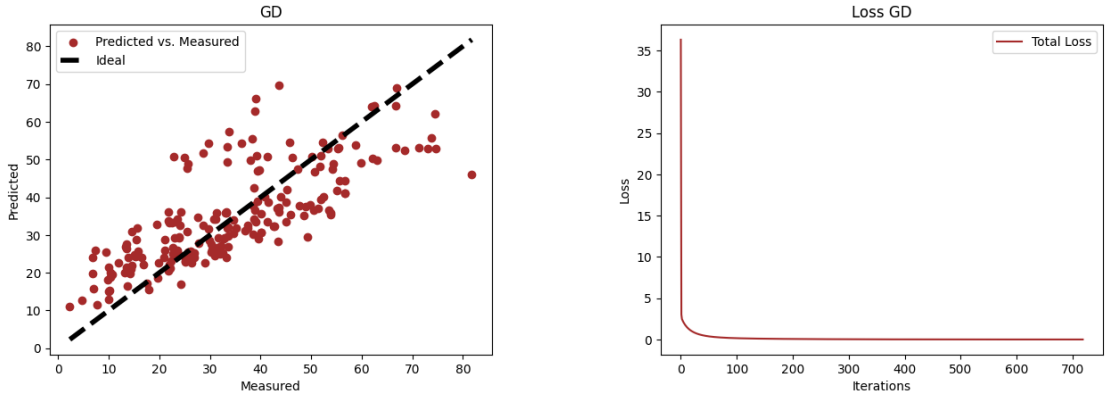
Si può notare che i valori della metrica siano molto simili per tutte le versioni, la differenza principale è data dai tempi.

4.3 Dataset 3

In seguito alle prove su dataset sintetici, si passa ad analizzare il modello in un contesto realistico. Anche per questo dataset è stata ripetuta la procedura utilizzata per i test precedenti.

4.3.1 Gradient Descent

Per questa versione si osservano buoni risultati con valori elevati di *step size*, che viene quindi posto a 0.5, inoltre si nota che al crescere di $l1$ *penalty* si ottengono miglioramenti sia per quanto riguarda la metrica, che i tempi.



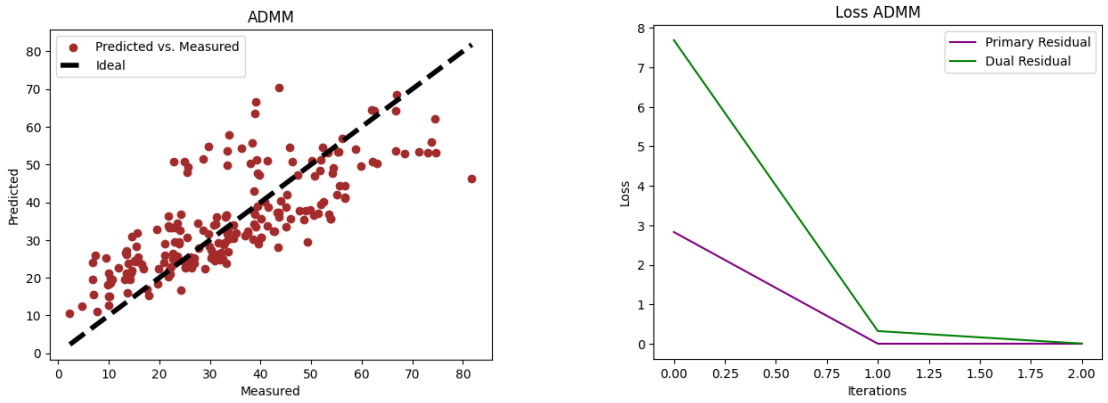
(a) Predizioni GD

(b) Loss GD

Figura 4.7: Gradient Descent sul Dataset 3

4.3.2 ADMM

L'ADMM è stata testata con un valore di *step size* di 0.1 e di $l1$ *penalty* di 0.1. In questo caso la scelta dei parametri influenza poco le prestazioni, sono stati dunque scelti dei parametri che garantiscono tempi brevi di addestramento.



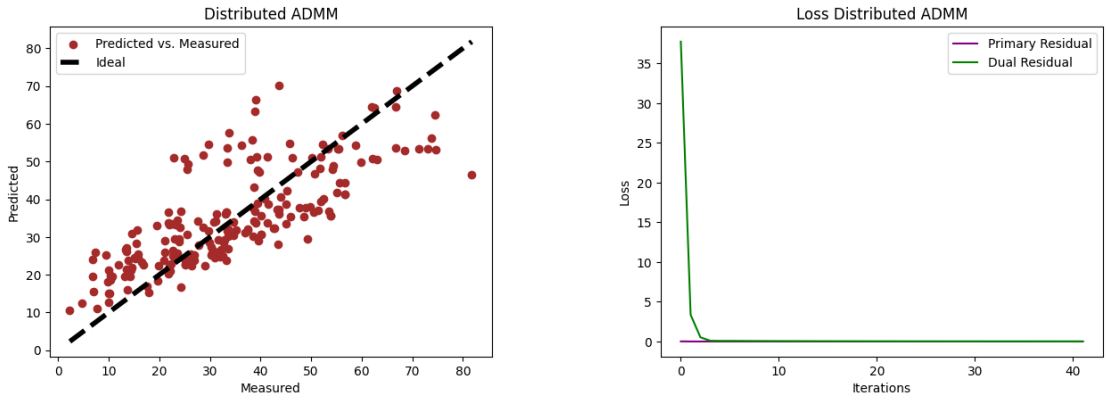
(a) Predizioni ADMM

(b) Residui ADMM

Figura 4.8: ADMM sul Dataset 3

4.3.3 ADMM distribuito

La versione distribuita dell'ADMM è stata testata con valori di 0.5 per lo *step size* e di 0.01 per $l1$ $penalty$. All'aumentare del numero degli agenti le prestazioni variano, sia in termini di R^2, che di numero di iterazioni e tempi. I risultati mostrano comunque che in caso di suddivisione dei campioni tra gli agenti si ottengono dei buoni risultati.



(a) Predizioni ADMM distribuito

(b) Residui ADMM distribuito

Figura 4.9: ADMM distribuito sul Dataset 3

4.3.4 Confronto

I risultati delle tre versioni sul dataset reale sono riportate nella tabella.

	R2	Tempo (s)	Iterazioni
GD	0.5786	0.0839	719
ADMM	0.5784	0.0026	3
ADMM Distribuito	0.5787	0.0560	42

Tabella 4.3: Confronto sul Dataset 1

Si può notare una minima differenza in termini di metrica, ma tempi inferiori per l'ADMM, che impiega anche un numero più basso di iterazioni.

4.4 Confronto Globale

Le versioni implementate sono state addestrate e testate su dati di varia natura, ciò che si è potuto osservare è che a livello di prestazioni si ottengono dei risultati molto simili. Il tempo impiegato dal Gradient Descent è generalmente maggiore rispetto agli altri ed anche il numero di iterazioni necessarie. La scelta dei parametri è importante, ma l'impatto dipende dal dataset utilizzato.

5. Conclusioni e Sviluppi Futuri

In conclusione, questa ricerca ha esplorato approcci teorici e implementativi alla Lasso Regression, analizzando le potenzialità di algoritmi alternativi come ADMM e ADMM distribuito. Durante l'implementazione, la versione distribuita è stata simulata mediante cicli for, non mostrando pienamente il potenziale di tale approccio come in ambienti distribuiti o con l'utilizzo di framework specifici. I risultati ottenuti, pur fornendo

indicazioni preziose, richiedono ulteriori approfondimenti e validazioni attraverso un numero più ampio di test su dataset diversificati.

- Sintesi dei Risultati: In generale, i risultati indicano una performance comparabile tra le diverse versioni in termini di predizioni, mentre si nota una maggiore differenza nei tempi.

- Considerazioni sull'Affidabilità: È fondamentale considerare che i risultati ottenuti sono indicativi e devono essere interpretati con attenzione. Limitazioni potenziali possono derivare dalla natura sintetica di alcuni dataset utilizzati, dalle configurazioni specifiche dei parametri e dall'ambiente di sviluppo. Verifiche supplementari e test su una gamma più ampia di dati contribuiranno a consolidare la robustezza delle conclusioni.

- Sviluppi Futuri e Ulteriori Ricerche: Per ampliare la comprensione e l'applicazione di tali modelli, futuri approfondimenti potrebbero includere l'esplorazione di varianti di ADMM, come quelle senza Fusion Center o basate sulla Sharing Optimization. Ulteriori ricerche potrebbero anche focalizzarsi su specifici scenari applicativi o confronti approfonditi con altri algoritmi di regressione.

In prospettiva, questa ricerca costituisce un punto di partenza solido per ulteriori indagini mirate a migliorare la comprensione e l'applicazione pratica della Lasso Regression e delle sue varianti.