Progetto 3 - Matera Manuel Roberto

Introduzione

Il presente progetto pone come obiettivo principale la costruzione di modelli di regressione per fare delle previsioni sui dati climatici del dataset di Delhi.

Nello specifico sono stati utilizzati 4 modelli di regressione, ciascuno dei quali, utilizza diversi metodi tra cui le equazioni normali, la fattorizzazione QR in modalità economica sia con pivoting che senza, la regressione alle componenti principali, e, come aggiunta nel progetto, l'intelligenza swarm.

L'intero progetto, scritto in Matlab, ha anche il suo corrispettivo in Python, in maniera tale da poter confrontare le varie implementazioni, che a loro volta utilizzano librerie diverse e sono state implementate in linguaggi differenti ma utili per lo stesso scopo.

L'obiettivo di questo documento è quello di poter descrivere la metodologia generale con cui ci si è approciati alla soluzione del problema principale, poter anche dare un'interpretazione dei risultati ottenuti nel progetto in Matlab e poterli confrontare con quelli ottenuti dalle implementazioni in Python, ispirate da quelle mostrate durante il corso

NOTE: All'interno del file **MATERA_MANUEL_ROBERTO_758386_PROGETTO3**.zip sono presenti i file con le implementazioni in Matlab, tra cui il **main.m** che è stato utilizzato per eseguire le routines implementate. Inoltre sono presenti due file .ipynb che sono stati utilizzati per il confronto fra le implementazioni in Python e in Matlab a seconda della feature target considerata.

Metodologia generale

Descrizione dei dati

I dati di training e di test utilizzati provengono dai file CSV "DailyDelhiClimateTrain.csv" e "DailyDelhiClimateTest.csv" rispettivamente. La dimensioni delle tabelle dei dati di training e di test sono rispettivamente: **1462 x 5**, **114 x 5**.

In questi dati rileviamo le feature **date**, **meantemp**, **humidity**, **wind_speed** e **meanpressure**. Inoltre, avendo notato che le misurazioni sono quotidiane, e ciò lo si evince dalla feature **date**, quest'ultima è stata esclusa nel training e test del modello. Pertanto fra le feature **target** consideriamo tutte le precedenti tranne la feature date.

Di seguito, mostriamo la metodologia per il caricamento dei dati di addestramento e di test:

---

main.m

```
% Caricamento dei dati di training
trainData = readtable('data/DailyDelhiClimateTrain.csv');

% Caricamento dei dati di test
testData = readtable('data/DailyDelhiClimateTest.csv');
```

Progetto3_MateraManuelRoberto_*.ipynb

```
import pandas as pd
trainData = pd.read_csv('/content/drive/MyDrive/DailyDelhiClimateTrain.csv')
testData = pd.read_csv('/content/drive/MyDrive/DailyDelhiClimateTest.csv')
```

Costruzione del modello di regressione

Per poter spiegare i passaggi, attraverso i quali viene costruito uno specifico modello di regressione, cerchiamo di formalizzare il problema.

Consideriamo un dataset contenente n osservazioni. Per ciascuna osservazione i, disponiamo di un valore della variabile indipendente x_i e un corrispondente valore della variabile dipendente y_i. Cerchiamo di modellare la relazione tra x e y attraverso un polinomio di grado d . Quindi possiamo esprimere y nel seguente modo:

$$y_i = b_0 + b_1 x_i + b_2 x_i^2 + \cdots + b_d x_i^d + \epsilon_i, \quad i = 1, \ldots, n$$

dove:

- $b_0, b_1, \ldots, b_d$ sono i coefficienti del polinomio da stimare.
- ϵ_i è l'errore che commettiamo nella misurazione.
- se $d = 1$ il modello di regressione è lineare, altrimenti polinomiale.

Per modellare questa relazione, definiamo la matrice di **Vandermonde** X e il vettore dei coefficienti b come segue:

$$X = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^d \\ 1 & x_1 & x_1^2 & \cdots & x_1^d \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^d \end{bmatrix}$$

$$b = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_d \end{bmatrix}$$

Pertanto il vettore $\hat{y}$, che rappresenta le previsioni del modello per le n osservazioni, sarà dato da:

$$\hat{y} = Xb$$

Il nostro obiettivo è trovare i coefficienti $b_0, b_1, \ldots, b_d$ che minimizzano l'errore ϵ, attraverso la soluzione del problema ai minimi quadrati.

Partendo dai dati di training, per trovare b sono stati utilizzati come metodi:

(Di seguito mostriamo: Metodo ==> nome per la selezione del metodo nel costruttore di **Modello**)

- **Fattorizzazione Thin QR ==> thin_qr**
  - **lss_thin_qr.m**
- **Fattorizzazione Thin QR con Pivoting ==> thin_qr_pivoting**
  - **lss_qr_pivoting.m**
- **Equazioni normali ==> normali**
  - **lss_normali.m**

- **Regressione alle componenti principali**, con i criteri per il troncamento dell'**SVD**:

  I metodi sono implementati all'interno della classe **PCR**

  - **Scree-Plot Cattel** ==> **svd_scree_plot_cattel**
  - **Criterio di Guttman Keiser** ==> **svd_guttman_keiser**
  - **Criterio dell'energia** ==> **svd_energia**
  - **Criterio dell'entropia** ==> **svd_entropia**

- **Intelligenza swarm**, in particolare l'algoritmo **PSO** è stato implementato sia senza velocità pesate (==> **swarm**), sia con velocità pesate secondo i seguenti metodi:

  I metodi sono implementati all'interno della classe **Swarm**

  - **Aggiustamento randomico** ==> **swarm_rand**
  - **Decremento lineare** ==> **swarm_d_lineare**
  - **Decremento non lineare** ==> **swarm_d_non_lineare**

Una volta determinato b con uno dei metodi sopra elencati, termina la fase di training e si passa a quella di testing, in cui, calcoliamo il vettore di predizioni sui dati di testing e, sfruttando la relazione precedente $\hat{y} = Xb$, ci basterà moltiplicare b, per la matrice X di test e poter ottenere le nostre predizioni.

In seguito a questa fase valutiamo la qualità delle nostre predizioni attraverso l'**RMSE**, che è stata implementata nel seguente modo:

Util.m

```
% Metodo statico della classe Util
function errore = rmse(valore_esatto, predizione)
    errore = sqrt(mean((valore_esatto - predizione).^2));
end
```

Progetto3_MateraManuelRoberto_*.ipynb

```
from sklearn.metrics import mean_squared_error
def RMSE(valore_esatto, predizione):
   return np.sqrt(mean_squared_error(valore_esatto, predizione))
```

Tutto il procedimento sopra descritto è stato implementato in Matlab, analogamente anche in Python, sfruttando una classe **Modello**, che potesse permettere di poter generalizzare la costruzione del modello a seconda delle esigenze dell'utente destinato ad utilizzare queste routines.

Richiamando il costruttore di classe, l'utente sarà, appunto, in grado di poter specificare una fra le feature target, precedentemente descritte, indicare se normalizzare i dati, specificare il metodo per il calcolo dei coefficienti, e il grado del polinomio che esprime la relazione precedentemente discussa. (Il massimo grado da poter indicare è 4)

Modello.m

```
function obj = Modello(trainData, testData, metodo,
target, isNormalizzato, d)
    obj.metodo = metodo;
    obj.trainData = trainData;
    obj.testData = testData;
    %Setto la variabile target a seconda di quella specificata
    switch target
        case 'meantemp'
            [obj.yTrain, obj.yTest] = obj.setVariabileTarget(2);
        case 'humidity'
            [obj.yTrain, obj.yTest] = obj.setVariabileTarget(3);
        case 'wind_speed'
            [obj.yTrain, obj.yTest] = obj.setVariabileTarget(4);
        case 'mean_pressure'
            [obj.yTrain, obj.yTest] = obj.setVariabileTarget(5);
        otherwise
            disp('Target non valida');
    end
    obj.isNormalizzato = isNormalizzato;
    % d indica il grado del polinomio per il modello
    obj.d = d;
    % Entrambe le dimensioni servono per poter definire le feature
    % di input
    dim1 = size(obj.yTrain, 1);
    dim2 = size(obj.yTest, 1);
    % Inizializzo tutte le feature necessarie per la costruzione
    % del modello
    [obj.XTrain, obj.yTrain, obj.XTest, obj.yTest]
                                        = obj.getFeature(dim1, dim2);
    % Ricavo i coefficienti, a seconda del metodo specificato, per
    % poter ottenere le predizioni
    [coeff, obj.condizionamento, obj.tempo] = obj.getCoefficienti();
    %disp(coeff);
    % Calcolo la predizione
    obj.yPred = obj.getPredizione(coeff);
    % Calcolo l'RMSE
    obj.rmse = Util.rmse(obj.yTest, obj.yPred);
end
```

Risultati

Di seguito mostriamo i risultati prodotti per le feature target **meantemp** e **mean_pressure**, su ciascun modello, metodo, con e senza normalizzazione.

Come criterio di normalizzazione si è scelta la normalizzazione minMax, dato che sperimentalmente ha prodotto una RMSE più bassa rispetto alla normalizzazione standard.

Questo metodo di normalizzazione è implementato all'interno della classe **Util** come metodo statico:

Util.m

```
function val = normalizza_minMax(X, min, Max)
    val = (X - min) ./ (Max - min);
end
```

Scelta dei parametri

La scelta dei parametri dipende dal metodo utilizzato:

- **Scree-Plot Cattel**: il criterio è soggettivo, il k selezionato viene scelto dall'utente finale. Nel nostro caso di studio li abbiamo considerati tutti.
- **Guttman-Keiser**: si è voluto generalizzare il metodo ad un certo valore 'soglia', nel nostro caso la soglia scelta è quella per definzione, cioè 1.
- **Energia**: si è scelto di conservare il 90% dell'energia di Σ.
- **Entropia**: si è scelta come percentuale di entropia il 95%.
- **PSO**: volendo adottare una strategia di **global best**, si è deciso di dare un valore maggiore per la componente c_2 nell'algoritmo. I parametri scelti sono presenti all'interno del metodo privato **getParametriPSO** all'interno della classe **Modello**.

  Inoltre la funzione obiettivo scelta è il **Mean Squared Error**, che è una funzione convessa.

Tempo medio di esecuzione

Per poter confrontare l'efficienza dei metodi utilizzati per ciascun modello in Matlab e in Python si è deciso di sfruttare l'attributo **tempo** di ciascun modello costruito in base ad uno specifico metodo e calcolarne la media su 100 iterazioni, questo viene realizzato in Matlab nel metodo statico **calcola_tempo_medio_esecuzione** nella classe **Util**.

Richiamando il metodo statico della classe **Util stampa_tempo_medio_esecuzione**, è possibile ottenere il tempo medio di esecuzione di ciascun metodo in base a quelli che sono i parametri del modello desiderato e in base ad un massimo di iterazioni. Si possono mediare tutti i metodi tranne quello dello Scree-Plot di Cattel, che richiede la visualizzazione del grafico.

Target: 'meantemp'

Dati non normalizzati

Modello di regressione lineare

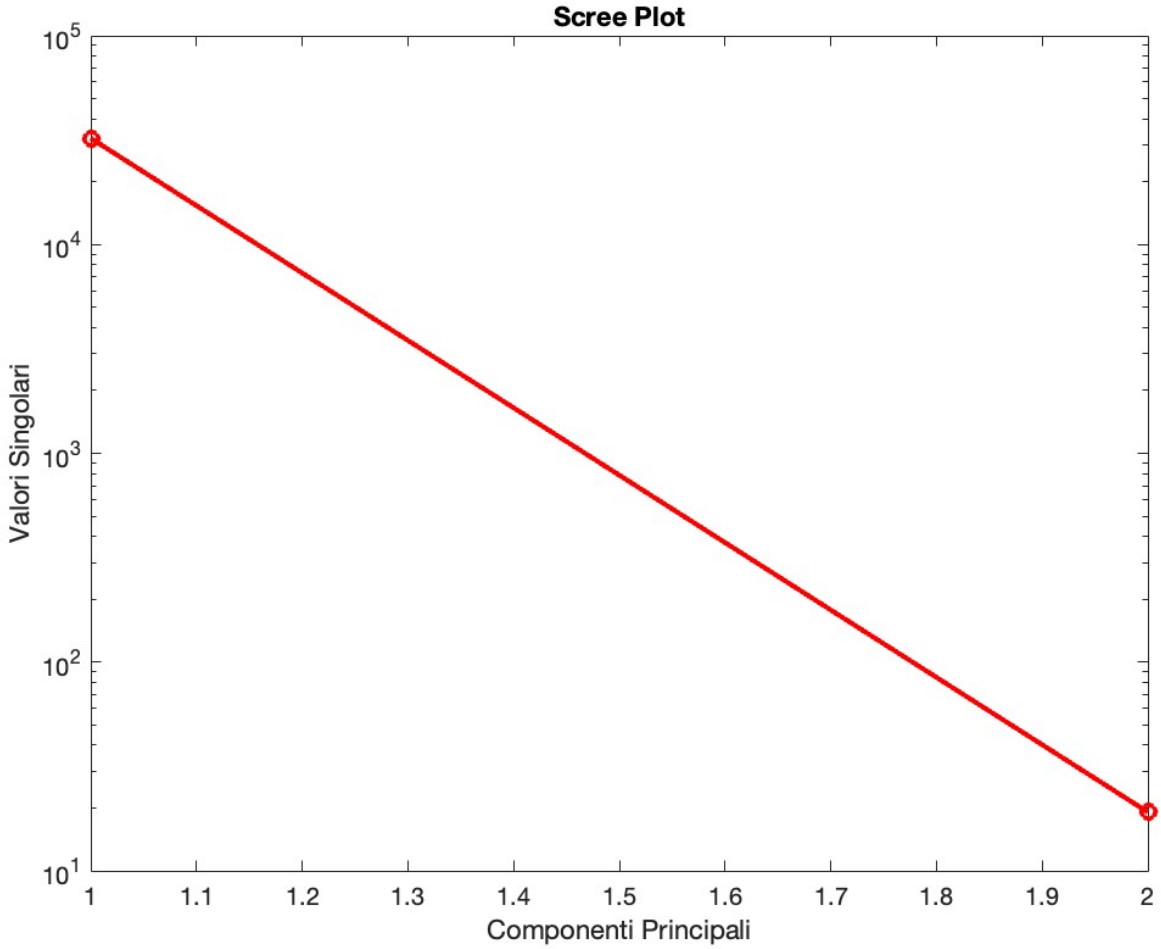
MATLAB

Rango della matrice di Training: 2

Notiamo che il condizionamento con le equazioni Normali è significativamente più alto rispetto agli altri metodi, tuttavia, in termini di tempo di esecuzione, la risoluzione con equazioni Normali risulta più efficiente.

I metodi di PSO risultano più lenti rispetto agli altri.

	RMSE	TEMPO	CONDIZIONAMENTO	TEMPO MEDIO DI ESECUZIONE
Normali	8.3843	0.0002	2855785.0123	0.000022
Thin QR	8.3843	0.0004	1689.9068	0.000120
Thin QR Pivoting	8.3843	0.0007	1689.9068	0.000083
Scree-Plot Cattel	8.3843	0.0013	1689.9068	/
Guttman Keiser	8.3843	0.0010	1689.9068	0.000018
Energia	19.6832	0.0016	1	0.000084
Entropia	19.6832	0.0023	1	0.000157
Swarm	16.8023	0.0464	/	0.030649
Swarm Random	16.2815	0.0598	/	0.031236
Swarm Decremento Lineare	16.9699	0.0415	/	0.030767
Swarm Decremento Non Lineare	18.6259	0.0410	/	0.048254



Ulteriori informazioni

- **k** scelto in **Scree-Plot di Cattel**, **Guttman Keiser**: 2
- **k** scelto dal criterio dell'**energia** e dell'**entropia**: 1
- Vettore di permutazione qr pivoting: $[2, 1]$

PYTHON

Non ci sono significativi cambiamenti se non in termini di tempo e nella variazione della RMSE per i metodi di PSO.

Tutti i metodi utilizzati in Python risultano più lenti rispetto a quelli in Matlab, tranne che per i metodi di Entropia ed Energia che si sono rivelati leggermente più efficienti, rispetto alle implementazioni in Matlab.

	RMSE	TEMPO	CONDIZIONAMENTO	TEMPO MEDIO DI ESECUZIONE
Normali	8.3843	0.0002	2855785.0123	0.000149
Thin QR	8.3843	0.0003	1689.9068	0.000185
Thin QR Pivoting	8.3843	0.0003	1689.9068	0.000189
Scree-Plot Cattel	8.3843	0.0021	1689.9068	/
Guttman Keiser	8.3843	0.0003	1689.9068	0.000064
Energia	19.6832	0.0001	1	0.000066
Entropia	19.6832	0.0003	1	0.000152
Swarm	16.5261	0.9781	/	0.883972
Swarm Random	14.5293	1.0187	/	0.821853
Swarm Decremento Lineare	19.6107	0.9661	/	0.857342
Swarm Decremento Non Lineare	19.1814	0.9020	/	0.802227

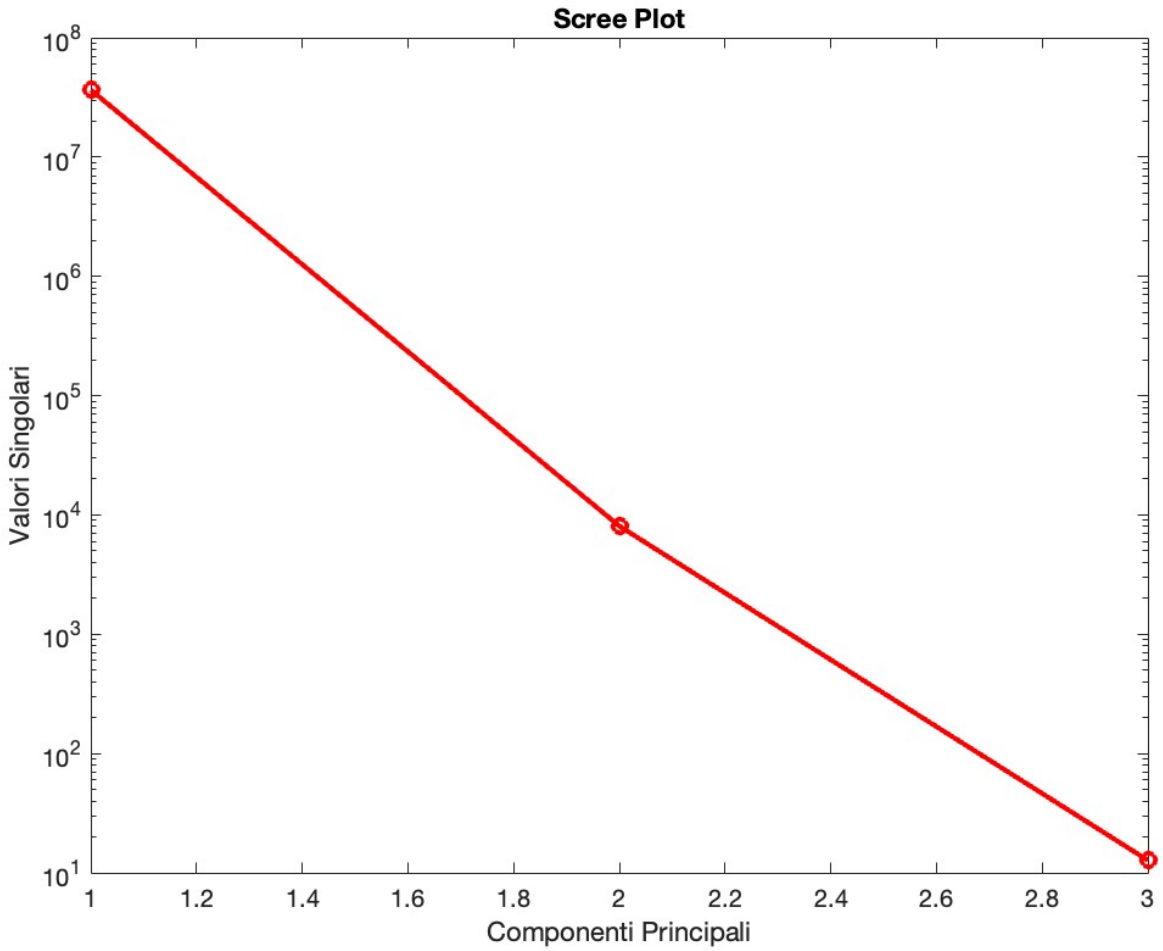
Modello di regressione polinomiale, d = 2

MATLAB

Rango della matrice di training: 3

I metodi con le equazioni Normali, la Thin QR con e senza pivoting, e il criterio di Guttman Keiser hanno mostrato in questo caso una RMSE più bassa rispetto al modello con $d = 1$. In termini di tempo di esecuzione non ci sono cambiamenti significativi rispetto al modello lineare, tuttavia risulta evidente che ad un aumento della dimensione dell'input, ossia la matrice di training, corrisponde un conseguente aumento del condizionamento. In questo caso il condizionamento delle Normali è maggiore di ordini di grandezza rispetto agli altri metodi. Ci saremmo aspettati che con un condizionamento più alto, si sarebbe ottenuta una RMSE altrettanto più alta, ma così non è stato. Osserveremo i risultati dei successivi modellli per trarre delle conclusioni.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	8.2575	0.0002	8260316083806.6846	0.000022
Thin QR	8.2575	0.0011	2874076.5598	0.000171
Thin QR Pivoting	8.2575	0.0010	2874076.5598	0.000081
Scree-Plot Cattel	9.2824	0.0008	4531.3776	/
Guttman Keiser	8.2575	0.0014	2874076.5598	0.000016
Energia	26.0449	0.0018	1	0.000012
Entropia	26.0449	0.0053	1	0.000224
Swarm	1404.1630	0.0506	/	0.033391
Swarm Random	9.1504	0.0552	/	0.032146
Swarm Decremento Lineare	144.1117	0.0451	/	0.028757
Swarm Decremento Non Lineare	44.4654	0.0707	/	0.029142



Ulteriori informazioni

- **k** scelto per lo **Scree-Plot di Cattel**: 2
- **k** scelto da **Guttman Keiser**: 3
- **k** scelto dal criterio dell'**energia** e dell'**entropia**: 1
- Vettore di permutazione qr pivoting: $[3, 2, 1]$

PYTHON

In termini di RMSE non cambia nulla, tuttavia qui risulta ancora più evidente la leggera inefficienza dei metodi in Python rispetto a quelli in Matlab. Per i metodi di PSO in Python notiamo che oltre alla forte variabilità dell'RMSE rispetto alle implementazioni in Matlab, il tempo medio di esecuzione è significativamente maggiore rispetto ai metodi in Matlab.

Un'altra osservazione è che cambia leggermente il condizionamento nelle equazioni Normali, molto probabilmente ciò è dovuto ad errori di arrotondamento.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	8.2575	0.0006	8260316067790.5234	0.000283
Thin QR	8.2575	0.0006	2874076.5598	0.000237
Thin QR Pivoting	8.2575	0.0003	2874076.5598	0.000236
Scree-Plot Cattel	9.2824	0.0014	4531.3776	/
Guttman Keiser	8.2575	0.0002	2874076.5598	0.000129
Energia	26.0449	0.0002	1	0.000054
Entropia	26.0449	0.0004	1	0.000162
Swarm	983.4858	1.1903	/	0.924309
Swarm Random	9.1310	1.0243	/	0.922556
Swarm Decremento Lineare	9.0518	0.9175	/	0.908149
Swarm Decremento Non Lineare	399.9416	0.9291	/	0.829374

Modello di regressione polinomiale, d = 3

MATLAB

Rango della matrice di Training: 4

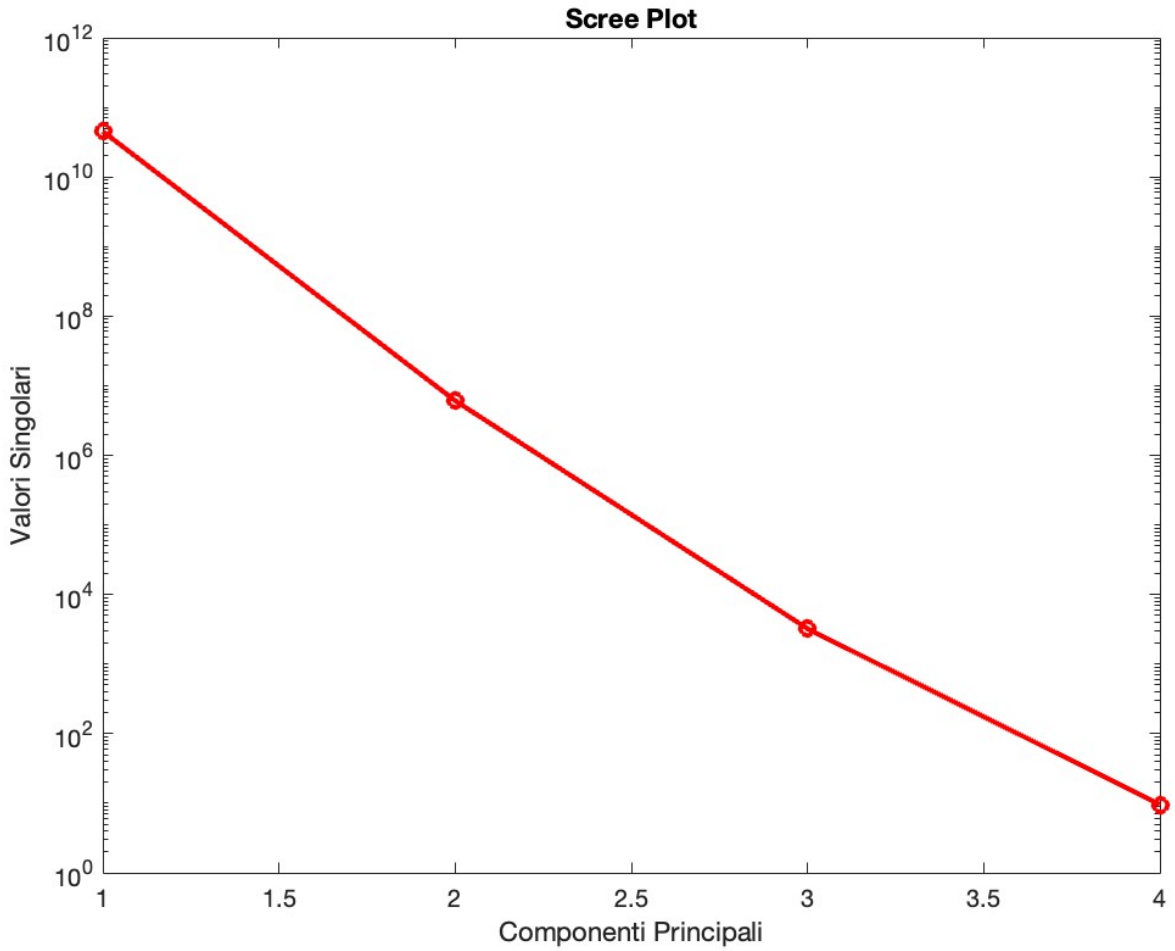
In questo caso otteniamo un **warning**, molto probabilmente legato alla matrice AA^T, nelle equazioni Normali, dal momento che risulta molto evidente quanto sia davvero molto mal condizionato il problema in quel caso, dato che la matrice è appunto **quasi singolare**:

```
Warning: Matrix is close to singular or badly scaled. Results may
be inaccurate. RCOND =  4.420321e-20.
```

Infatti RCOND, ossia il reciproco del numero di condizionamento è molto prossimo allo 0, il che è un segnale che indica la forte instabilità numerica della matrice.

In termini di RMSE notiamo un peggioramento generale di ciascun metodo rispetto al modello con $d = 2$. Lo stesso vale anche per il tempo medio di esecuzione di ciascun metodo, tranne che per quelli in cui si applicato un criterio di troncamento dell'SVD per la regressione alle componenti principali.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E
Normali	8.8917	0.0040	22490009842337112064	0.000801
Thin QR	8.8917	0.0005	4742348076.0351	0.000383
Thin QR Pivoting	8.8917	0.0007	4742348080.3964	0.000107
Scree-Plot Cattel	16.6172	0.0033	14002282.9919	/
Guttman Keiser	8.8917	0.0023	4742348076.0351	0.000018
Energia	30.5398	0.0028	1	0.000011
Entropia	30.5398	0.0051	1	0.000150
Swarm	126685.0048	0.0528	/	0.029014
Swarm Random	45135.8488	0.0586	/	0.029143
Swarm Decremento Lineare	678155.6231	0.0667	/	0.028235
Swarm Decremento Non Lineare	162132.7604	0.0462	/	0.051208



Ulteriori informazioni

- **k** scelto per lo **Scree-Plot di Cattel**: 3
    - Scegliendo k=2 abbiamo che:
        - RMSE = 16.4181
        - Tempo trascorso: 0.0003
        - Condizionamento: 7416.2212
- **k** scelto da **Guttman Keiser**: 4
- **k** scelto dal criterio dell'**energia** e dell'**entropia**: 1
- Vettore di permutazione qr pivoting: $[4, 3, 2, 1]$

PYTHON

L'**errror handler** di Python ci ha confermato dove si verifica il problema discusso in precedenza, ossia nel calcolo con le equazioni normali:

```
/var/folders/81/54fv74bj2d5_6s7bxrxxsz280000gn/T/ipykernel_79897/1209688551.py:
4:
LinAlgWarning: Ill-conditioned matrix (rcond=4.42032e-20):
result may not be accurate.
soluzioni = las.solve(ATA, (A.T @ b))
```

In termini di tempo sono risultati più efficienti rispetto al corrispettivo in Matlab i metodi con le equazioni Normali e la Thin QR.

Notiamo inoltre che, per quanto non sia variata l'RMSE per ciascun metodo in Python rispetto ai metodi in Matlab, tuttavia il condizionamento risulta leggermente diverso per quanto riguarda: Normali, Thin QR, Thin QR con Pivoting, Scree-Plot Cattel, Guttman-Keiser.

Anche in questo caso i metodi di PSO sono decisamente più inefficienti rispetto alla implementazione in Matlab, oltre che a mostrare una forte variabilità nell'RMSE.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	8.8917	0.0016	22490142494780456960	0.000261
Thin QR	8.8917	0.0005	4742348081.2452	0.000299
Thin QR Pivoting	8.8917	0.0003	4742348080.3965	0.000296
Scree-Plot Cattel	16.6172	0.0010	14002282.9854	/
Guttman Keiser	8.8917	0.0002	4742348081.2453	0.000151
Energia	30.5398	0.0001	1	0.000055
Entropia	30.5398	0.0004	1	0.000263
Swarm	1669267.3522	1.0081	/	0.950188
Swarm Random	470303.7841	0.9431	/	0.937658
Swarm Decremento Lineare	230945.0942	0.9472	/	0.945349
Swarm Decremento Non Lineare	337008.5361	0.8698	/	0.861545

- troncando con $k = 2$:
  - RMSE: 16.4181
  - Tempo: 0.0011
  - Condizionamento: 7416.2212

Modello di regressione polinomiale, d = 4

MATLAB

Rango della matrice di Training: 4, quindi la matrice non ha rango massimo e alcuni valori singolari sono nulli.

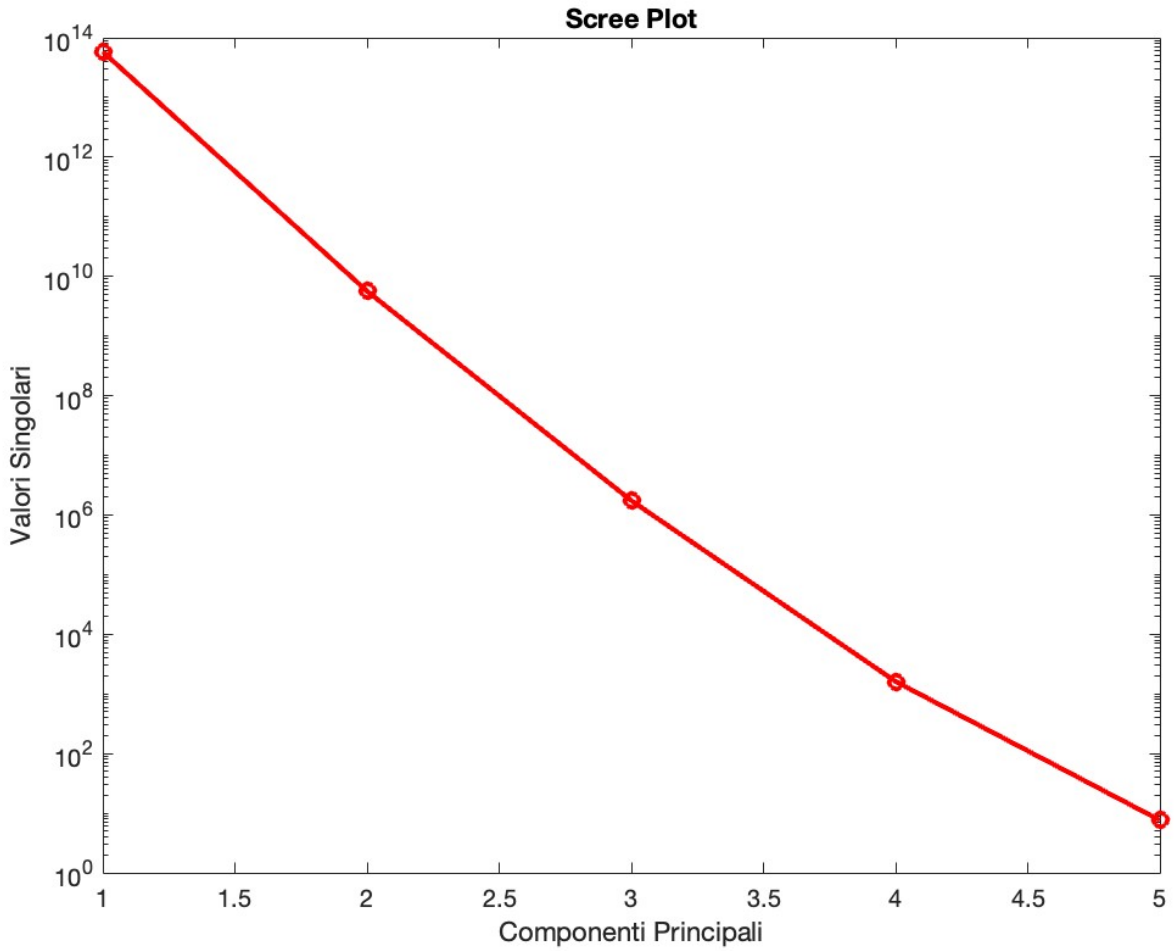
Come nel caso con $d = 3$, anche qui il condizionamento della matrice AA^T nelle equazioni normali, è significativamente più alto, e quindi un forte segnale della sua instabilità numerica:

```
Warning: Matrix is close to singular or badly scaled. Results may
be inaccurate. RCOND =  1.690155e-26.
```

Anche in questo caso l'RMSE per ciascun metodo utilizzato risulta più alto rispetto al modello precedente, lo stesso vale per il tempo di esecuzione di ciascun metodo; tuttavia in quest'ultimo caso notiamo che il tempo medio di esecuzione nei metodi di Guttman Keiser, Energia ed Entropia risulta leggermente più basso rispetto a quello che si ha nel modello precedente.

Anche qui l'RMSE dei metodi di PSO risulta decisamente più alta rispetto al modello precedente.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E
Normali	10.5328	0.0030	77700313479421129543122944	0.0010
Thin QR	10.5328	0.0005	7657887098487.1289	0.0004
Thin QR Pivoting	10.5328	0.0007	7657829856730.1572	0.0001
Scree-Plot Cattel	17.2643	0.0003	36119589295.4145	/
Guttman Keiser	10.5329	0.0008	7657887098487.1279	0.0000
Energia	34.2716	0.0008	1	0.0000
Entropia	34.2716	0.0100	1	0.0001
Swarm	1180455496.3883	0.0857	/	0.0289
Swarm Random	1267933.9255	0.0715	/	0.0305
Swarm Decremento Lineare	510074734.5880	0.0577	/	0.0292
Swarm Decremento Non Lineare	168421289.6292	0.0418	/	0.0300



Il grafico tuttavia non sembra essere coerente con ciò che è stato detto in precedenza, molto probabilmente trattandosi di una matrice molto mal condizionata, ci saranno stati sicuramente diversi errori di arrotondamento.

Ulteriori informazioni

- **k** scelto per lo **Scree-Plot di Cattel**: 4
  - Scegliendo $k = 2$ abbiamo che:
    - RMSE = 24.7055
    - Tempo trascorso: 0.0022
    - Condizionamento: 10318.4287
  - Scegliendo $k = 3$ abbiamo che
    - RMSE: 22.6599
    - Tempo trascorso: 0.0021

    - Condizionamento: 33479063.4719

- **k** scelto da **Guttman Keiser**: 5
- **k** scelto dal criterio dell'**energia** e dell'**entropia**: 1
- Vettore di permutazione qr pivoting: $[5, 4, 3, 2, 1]$

PYTHON

Anche qui l'error handler di Python ci ha dato la stessa conferma discussa in precedenza.

Notiamo che i metodi con le Normali e Thin QR risultano più efficienti rispetto alle implementazioni in Matlab, tuttavia ciò non è valido anche per gli altri metodi che sono più lenti rispetto ai corrispondenti.

Vale anche qui la forte variabilità dell'RMSE per i metodi di PSO, oltre che a un forte peggioramento in termini di tempo di esecuzione.

Cambia leggermente anche il condizionamento per i metodi Normali, Thin QR, Thin QR con pivoting, Scree-Plot Cattel, Guttman Keiser.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E
Normali	10.5328	0.0006	52792673932931379167756288	0.0002
Thin QR	10.5328	0.0004	7657841198171.7930	0.0002
Thin QR Pivoting	10.5328	0.0003	7657829856730.1201	0.0003
Scree-Plot Cattel	17.2643	0.0014	36119753800.8594	/
Guttman Keiser	10.5329	0.0003	7657841198125.8379	0.0001
Energia	34.2716	0.0002	1	0.0000
Entropia	34.2716	0.0004	1	0.0003
Swarm	3644459525.5722	0.9859	/	1.0331
Swarm Random	6799920.1851	1.0004	/	1.0266
Swarm Decremento Lineare	60657973.8335	1.0454	/	1.0358
Swarm Decremento Non Lineare	257672515.4910	0.9261	/	0.9383

troncando con:

- $k = 2$:
  - RMSE: 24.7055
  - Tempo: 0.0010
  - Condizionamento: 10318.4287
- $k = 3$:
  - RMSE: 22.6599
  - Tempo: 0.0010

- Condizionamento: 33479063.4723

Dati normalizzati

Modello di regressione lineare

MATLAB

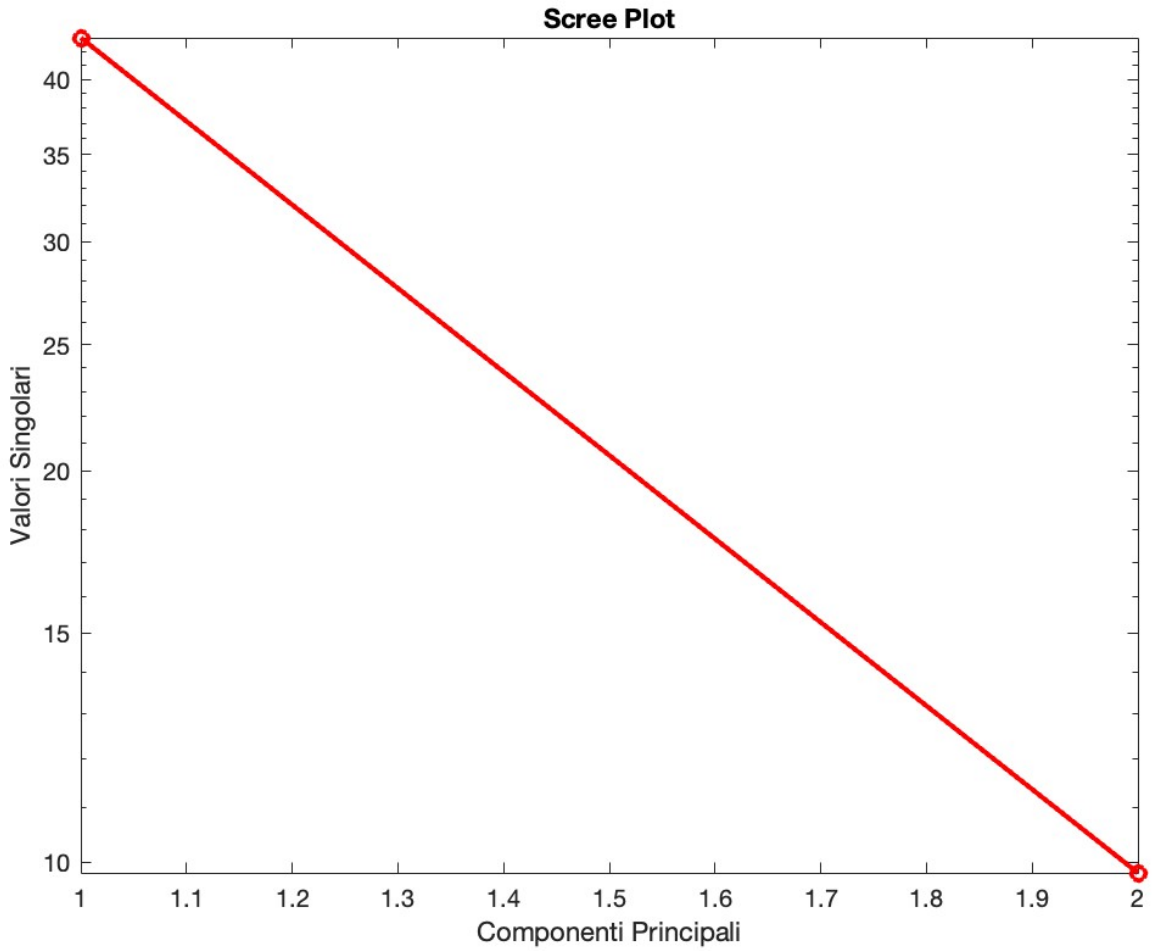
Rango della matrice di Training: 2

Risulta evidente come la normalizzazione dei dati abbia prodotto risultati in termini di ordine di grandezza inferiori per quanto riguarda la RMSE e il condizionamento rispetto al modello lineare con dati non normalizzati.

È notevole il fatto che per quanto riguarda i metodi di PSO, sebbene più lenti rispetto agli altri metodi, non mostrano una forte variabilità in termini di RMSE, quindi sembrano essere più stabili.

In termini di tempo medio di esecuzione notiamo un leggero miglioramento per ciascun metodo.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E
Normali	0.2563	0.0050	19.2559	0.000018
Thin QR	0.2563	0.0021	4.3882	0.000151
Thin QR Pivoting	0.2563	0.0061	4.3882	0.000047
Scree-Plot Cattel	0.2563	0.0058	4.3882	/
Guttman Keiser	0.2563	0.0043	4.3882	0.000015
Energia	0.3091	0.0063	1	0.000011
Entropia	0.3091	0.0140	1	0.000128
Swarm	0.2481	0.0548	/	0.035021
Swarm Random	0.2563	0.0375	/	0.025276
Swarm Decremento Lineare	0.2563	0.0296	/	0.021493
Swarm Decremento Non Lineare	0.2563	0.0358	/	0.029895



Ulteriori informazioni

- **k** scelto in **Scree-Plot di Cattel**, **Guttman Keiser**: 2
- **k** scelto dal criterio dell'**energia** e dell'**entropia**: 1
- Vettore di permutazione qr pivoting: **ordinato**

PYTHON

Rispetto al corrispettivo i Matlab, ciascun metodo in Python risulta più lento, tuttavia in termini di RMSE e condizionamento non notiamo cambiamenti significativi se non per quanto riguarda la RMSE per la swarm con velocità non pesate e per il metodo con decremento non lineare.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E
Normali	0.2563	0.0012	19.2559	0.000189
Thin QR	0.2563	0.0019	4.3882	0.000224
Thin QR Pivoting	0.2563	0.0002	4.3882	0.000422
Scree-Plot Cattel	0.2563	0.0012	4.3882	/
Guttman Keiser	0.2563	0.0002	4.3882	0.000225
Energia	0.3091	0.0001	1	0.000081
Entropia	0.3091	0.0004	1	0.000278
Swarm	0.2418	0.9814	/	0.978239
Swarm Random	0.2563	0.8621	/	0.747191
Swarm Decremento Lineare	0.2563	0.7566	/	0.729972
Swarm Decremento Non Lineare	0.2567	0.8171	/	0.831281

Modello di regressione polinomiale, d = 2

MATLAB

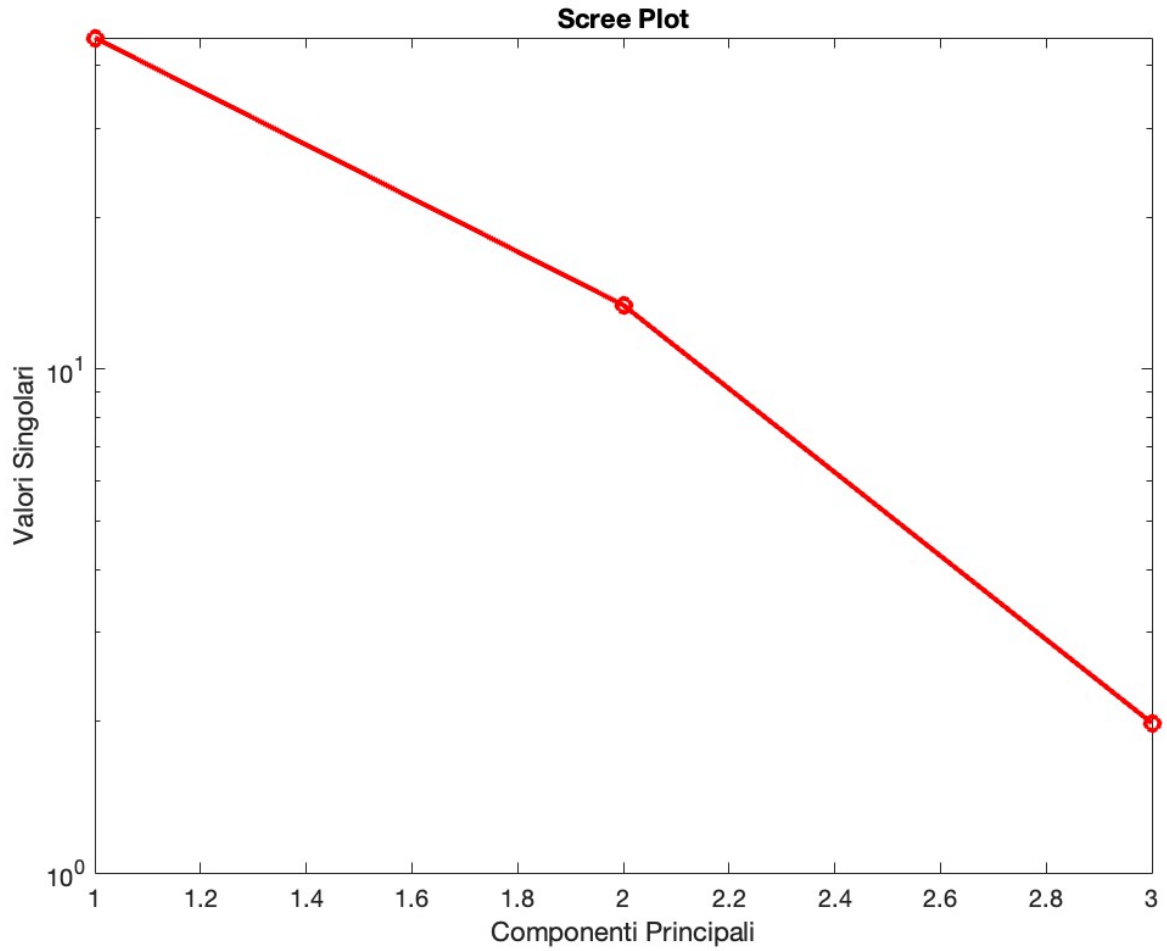
Rango della matrice di training: 3

Per quanto riguarda la RMSE per ciascun metodo, questa risulta leggermente migliore rispetto al modello lineare discusso in precedenza; ma per i metodi di Energia, Entropia e Swarm con velocità non pesate, quest'ultima affermazione è falsa.

Inoltre c'è un conseguente aumento del condizionamento delle matrici considerate, ma non sembra essere nulla di allarmante se considerati i risultati discussi in precedenza.

In termini di tempo medio di esecuzione non ci sono significativi peggioramenti

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	0.2524	0.0009	524.1339	0.000022
Thin QR	0.2524	0.0007	22.8940	0.000158
Thin QR Pivoting	0.2524	0.0005	22.8940	0.000068
Scree-Plot Cattel	0.2560	0.0012	3.3917	/
Guttman Keiser	0.2524	0.0009	22.8940	0.000019
Energia	0.3882	0.0013	1	0.000010
Entropia	0.3882	0.0035	1	0.000209
Swarm	0.2570	0.0436	/	0.045120
Swarm Random	0.2524	0.0472	/	0.046131
Swarm Decremento Lineare	0.2524	0.0375	/	0.037038
Swarm Decremento Non Lineare	0.2500	0.0420	/	0.039201



Ulteriori informazioni

- **k** scelto per lo **Scree-Plot di Cattel**: 2
- **k** scelto da **Guttman Keiser**: 3
- **k** scelto dal criterio dell'**energia** e dell'**entropia**: 1
- Vettore di permutazione qr pivoting: [1, 3, 2]

PYTHON

Per quanto la RMSE gli unici cambiamenti, rispetto a Matlab, poco evidenti sono nei metodi di Swarm con velocità non pesate, Swarm con decremento lineare e Swarm con decremento non lineare.

Inoltre, rispetto a Matlab è possibile notare un leggero peggioramento per quanto riguarda il tempo medio di esecuzione di ciascun metodo, tranne che per l'Entropia.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	0.2524	0.0006	524.1339	0.000166
Thin QR	0.2524	0.0008	22.8940	0.000316
Thin QR Pivoting	0.2524	0.0003	22.8940	0.000469
Scree-Plot Cattel	0.2560	0.0014	3.3917	/
Guttman Keiser	0.2524	0.0003	22.8940	0.000162
Energia	0.3882	0.0002	1	0.000047
Entropia	0.3882	0.0004	1	0.000207
Swarm	0.2644	0.9963	/	0.996812
Swarm Random	0.2524	0.9230	/	0.912404
Swarm Decremento Lineare	0.2539	0.8162	/	0.876031
Swarm Decremento Non Lineare	0.2637	0.8143	/	0.892226

Modello di regressione polinomiale, d = 3

MATLAB

Rango della matrice di Training: 4

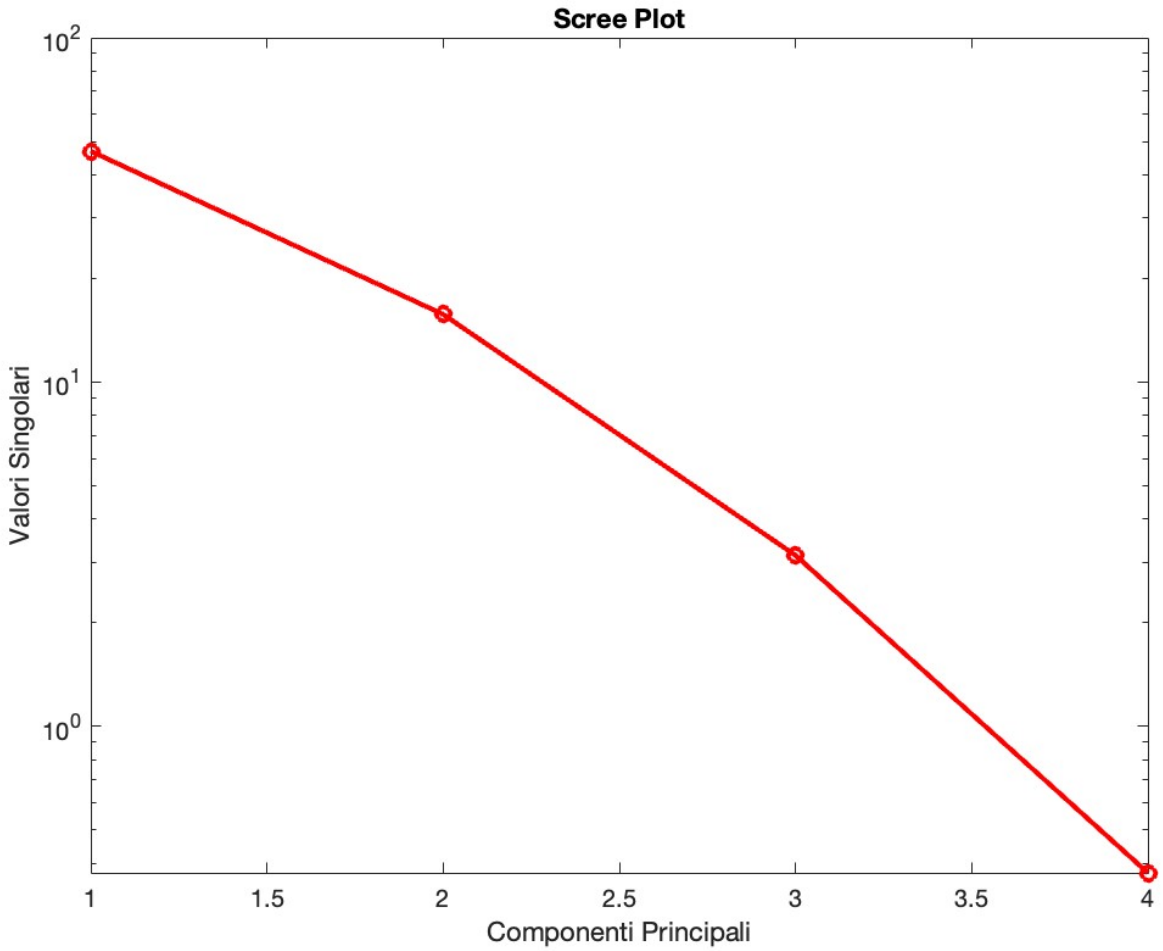
Innanzitutto, dato il condizionamento di AA^T nelle normali, notiamo subito che il problema discusso con i risultati di questo stesso modello, ma con i dati non normalizzati, è stato appunto risolto: non abbiamo warning e la matrice non è mal condizionata e quasi-singolare.

Rispetto al modello precedente notiamo un leggero peggioramento della RMSE, oltre che ad un aumento del condizionamento, per i motivi già discussi.

Notiamo che in questo caso la Thin QR con Pivoting risulta essere mediamente più efficiente rispetto alla variante senza pivoting.

Non notiamo significativi peggioramenti in termini di tempo medio di esecuzione, però sono comunque evidenti, tranne che per le equazioni Normali che in questo caso sono state leggermente più efficienti rispetto al modello precedente.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E
Normali	0.2718	0.0005	15536.1484	0.000017
Thin QR	0.2718	0.0014	124.6441	0.000220
Thin QR Pivoting	0.2718	0.0008	124.6441	0.000094
Scree-Plot Cattel	0.2538	0.0005	14.9176	/
Guttman Keiser	0.2538	0.0012	14.9176	0.000023
Energia	0.2525	0.0046	2.9771	0.000039
Entropia	0.4596	0.0012	1	0.000123
Swarm	0.2205	0.0508	/	0.030194
Swarm Random	0.2605	0.0503	/	0.033172
Swarm Decremento Lineare	0.2528	0.0351	/	0.028118
Swarm Decremento Non Lineare	0.2923	0.0407	/	0.031052



Ulteriori informazioni

- **k** scelto per lo **Scree-Plot di Cattel**: 3
- **k** scelto da **Guttman Keiser**: 3
  - infatti il quarto valore singolare è strettamente minore di 1
- **k** scelto dal criterio dell'**energia**: 2
- **k** scelto dal criterio dell'**entropia**: 1
- con $k = 4$:
  - RMSE: 0.2718
  - Tempo: 0.0007
  - Condizionamento: 124.6441
- Vettore di permutazione qr pivoting: $[1, 3, 2, 4]$

PYTHON

Rispetto a Matlab abbiamo gli stessi risultati per quanto riguarda l'RMSE di ciascun metodo, tranne che per i metodi di PSO.

Anche in questo caso si evidenza la leggera inefficienza dei metodi in Python rispetto a quelli in Matlab: i tempi di esecuzione sono maggiori.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	0.2718	0.0037	15536.1484	0.000147
Thin QR	0.2718	0.0006	124.6441	0.000303
Thin QR Pivoting	0.2718	0.0004	124.6441	0.000343
Scree-Plot Cattel	0.2538	0.0010	14.9176	/
Guttman Keiser	0.2538	0.0003	14.9176	0.000241
Energia	0.2525	0.0002	2.9771	0.000091
Entropia	0.4596	0.0004	1	0.000313
Swarm	0.2090	0.9533	/	1.039017
Swarm Random	0.2614	0.9281	/	1.018678
Swarm Decremento Lineare	0.2594	0.8645	/	0.952300
Swarm Decremento Non Lineare	0.2447	0.8954	/	0.915784

troncando con $k = 4$:

- RMSE: 0.2718
- Tempo: 0.0010
- Condizionamento: 124.6441

Modello di regressione polinomiale, d = 4

MATLAB

Rango della matrice di Training: 5

Anche in questo caso vale lo stesso discorso fatto per il modello precedente, sebbene il condizionamento nelle Normali sia alto, questo comunque non ci crea problemi.

Inoltre questo modello rispetto al corrispettivo con i dati non normalizzati ha rango massimo: quindi siamo certi che tutti i valori singolari siano non nulli.

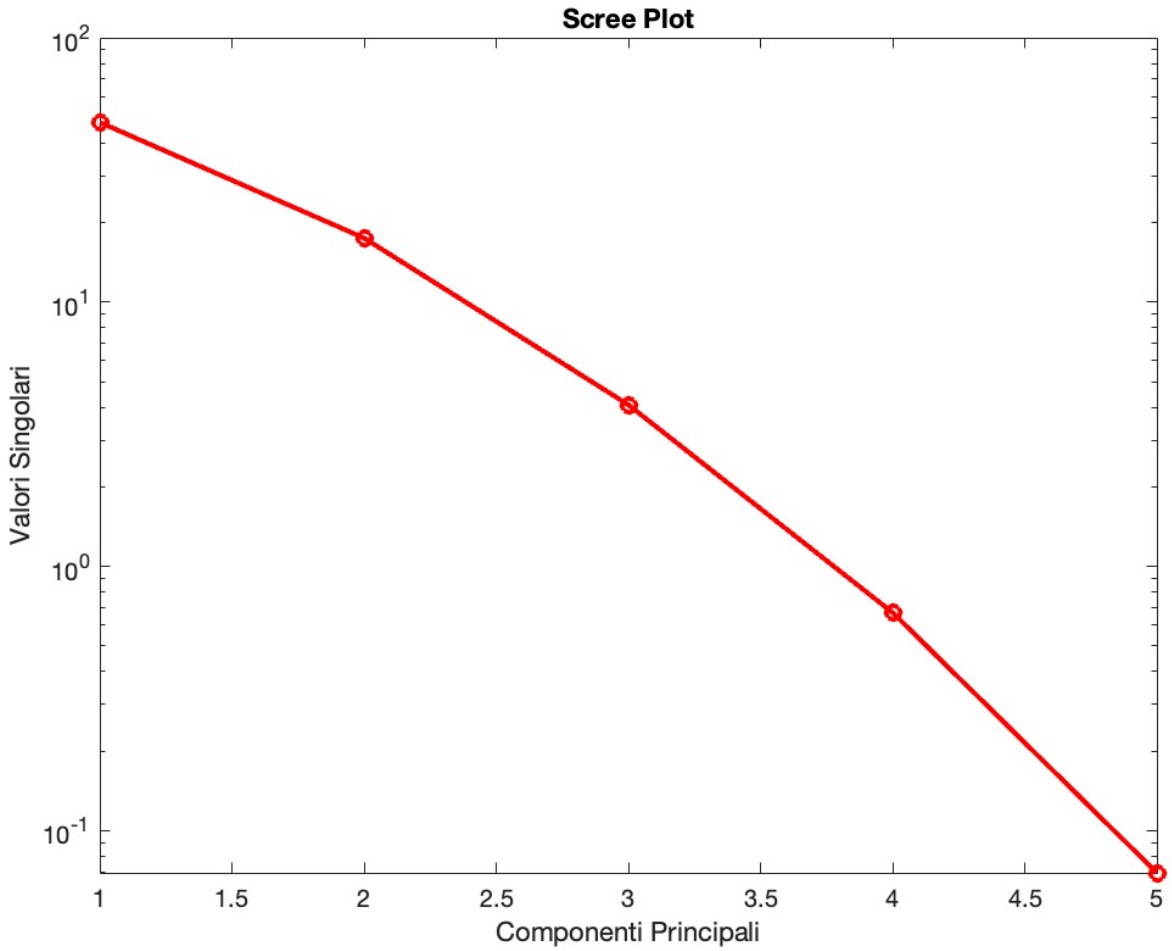
In termini di RMSE, notiamo che per il criterio dell'Energia, dell'Entropia, e nei metodi PSO di Swarm Random e Swarm con decremento non lineare, abbiamo un miglioramento rispetto al modello precedente.

Il tempo di esecuzione medio di ciascun metodo non presenta significativi peggioramenti.

Anche qui Thin QR con Pivoting risulta più efficiente della sua variante senza Pivoting.

I criteri di Guttman-Keiser ed Energia oltre a mostrare una RMSE più bassa rispetto agli altri metodi, risultano anche più efficienti.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	0.3220	0.0003	477661.1552	0.000035
Thin QR	0.3220	0.0004	691.1303	0.000202
Thin QR Pivoting	0.3220	0.0008	691.1303	0.000116
Scree-Plot Cattel	0.2531	0.0003	71.5992	/
Guttman Keiser	0.2523	0.0007	11.7158	0.000023
Energia	0.2471	0.0017	2.7403	0.000021
Entropia	0.2471	0.0094	2.7403	0.000274
Swarm	0.2872	0.0460	/	0.034328
Swarm Random	0.2534	0.0616	/	0.032405
Swarm Decremento Lineare	0.2564	0.0425	/	0.027382
Swarm Decremento Non Lineare	0.2324	0.0461	/	0.033310



Ulteriori informazioni

- **k** scelto per lo **Scree-Plot di Cattel**: 4
- **k** scelto da **Guttman Keiser**: 3
  - infatti gli ultimi due valori singolari sono strettamente minori di 1
- **k** scelto dal criterio dell'**energia** e dal criterio dell'**entropia**: 2
- con $k = 1$:
  - RMSE: 0.5251
  - Tempo: 0.0041
  - Condizionamento: 1
- con $k = 5$:
  - RMSE: 0.3220

    - Tempo: 0.0002
    - Condizionamento: 691.1303
- Vettore di permutazione qr pivoting: $[1, 3, 5, 2, 4]$

PYTHON

In questo caso cambia la RMSE per ciascun metodo di PSO: notiamo un peggioramento per quello con velocità non pesate e un miglioramento per le altre varianti.

Anche qui il tempo medio di esecuzione di ciascun metodo è più lento rispetto alle implementazioni in Matlab.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	0.3220	0.0005	477661.1552	0.000208
Thin QR	0.3220	0.0004	691.1303	0.000491
Thin QR Pivoting	0.3220	0.0003	691.1303	0.000313
Scree-Plot Cattel	0.2531	0.0010	71.5992	/
Guttman Keiser	0.2523	0.0013	11.7158	0.000298
Energia	0.2471	0.0002	2.7403	0.000094
Entropia	0.2471	0.0005	2.7403	0.000363
Swarm	0.3889	0.9793	/	1.000583
Swarm Random	0.2434	0.9568	/	1.010023
Swarm Decremento Lineare	0.2482	0.8769	/	0.931642
Swarm Decremento Non Lineare	0.2575	0.8610	/	0.894315

con $k = 1$:

- RMSE: 0.5251
- Tempo: 0.0020
- Condizionamento: 1

con $k = 5$:

- RMSE: 0.3220
- Tempo: 0.0021
- Condizionamento: 691.1303

Target: 'mean_pressure'

Dati non normalizzati

Modello di regressione lineare

MATLAB

Rango della matrice di Training: 2

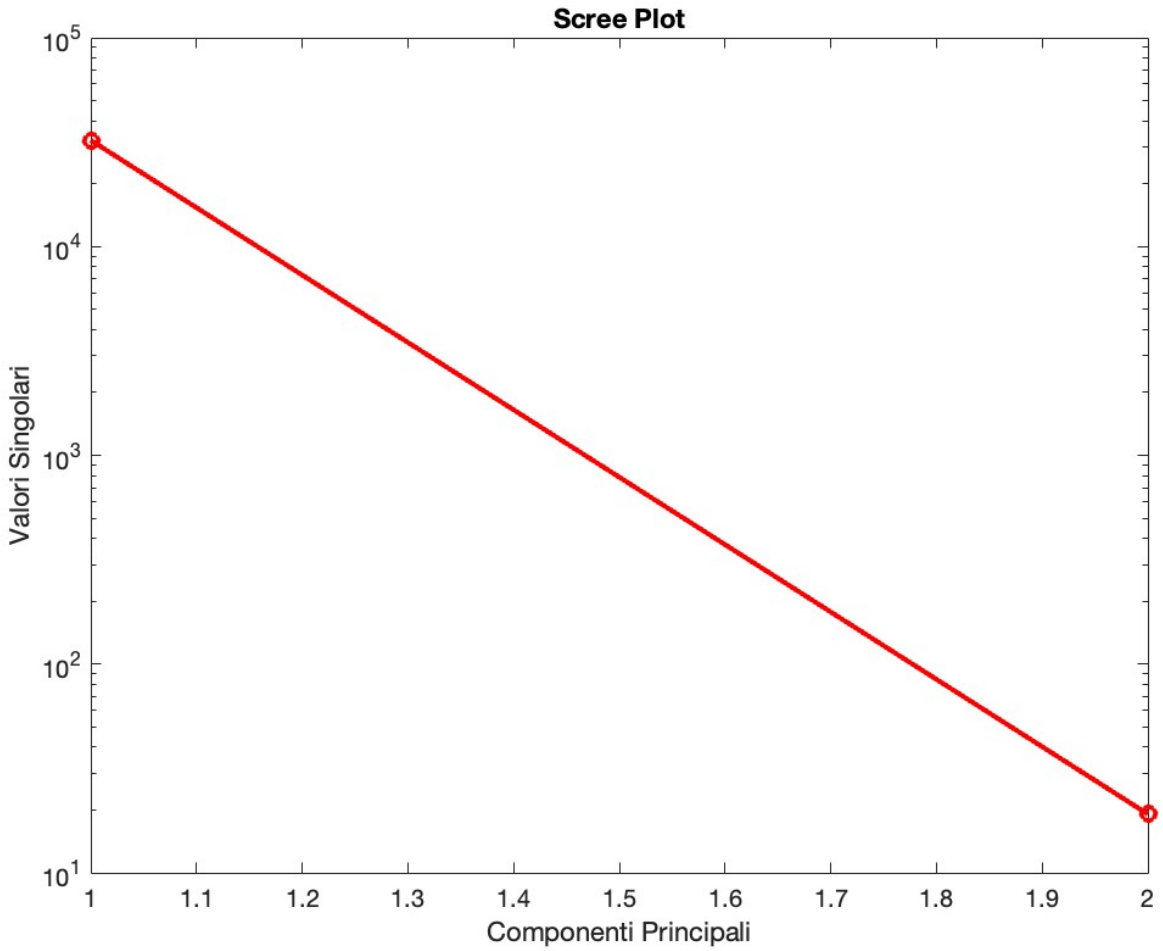
Possiamo trarre le stesse osservazioni fatte già per il modello lineare con dati non normalizzati per la feature 'meantemp'.

Notiamo inoltre che il tempo di esecuzione del metodo con Thin QR con Pivoting risulta più efficiente rispetto alla sua variante senza Pivoting.

Inoltre risultano più efficienti Guttman Keiser ed Energia, tuttavia quest'ultima produce una RMSE più alta.

I metodi di PSO sono sempre i più lenti e mostrano una RMSE molto variabile.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	89.8283	0.0003	2855785.0123	0.000015
Thin QR	89.8283	0.0003	1689.9068	0.000098
Thin QR Pivoting	89.8283	0.0006	1689.9068	0.000057
Scree-Plot Cattel	89.8283	0.0003	1689.9068	/
Guttman Keiser	89.8283	0.0004	1689.9068	0.000016
Energia	581.2799	0.0005	1	0.000014
Entropia	581.2799	0.0019	1	0.000106
Swarm	800.2978	0.0452	/	0.030069
Swarm Random	570.5642	0.0450	/	0.030513
Swarm Decremento Lineare	583.4478	0.0491	/	0.028744
Swarm Decremento Non Lineare	393.8607	0.0402	/	0.027915



Ulteriori informazioni

- **k** scelto in **Scree-Plot di Cattel**, **Guttman Keiser**: 2
- **k** scelto dal criterio dell'**energia** e dell'**entropia**: 1
- Vettore di permutazione qr pivoting: $[2, 1]$

PYTHON

In questo caso rispetto al corrispettivo in Matlab abbiamo che l'RMSE per ciascun metodo di PSO risulta diverso: abbiamo un miglioramento per quanto riguarda la variante con velocità non pesate e un peggioramento per quella con decremento non lineare.

Il tempo medio di esecuzione di ciascun metodo in Python è più lento rispetto al corrispettivo in Matlab.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	89.8283	0.0022	2855785.0123	0.000222
Thin QR	89.8283	0.0008	1689.9068	0.000275
Thin QR Pivoting	89.8283	0.0006	1689.9068	0.000293
Scree-Plot Cattel	89.8283	0.0013	1689.9068	/
Guttman Keiser	89.8283	0.0003	1689.9068	0.000133
Energia	581.2799	0.0002	1	0.000085
Entropia	581.2799	0.0024	1	0.000168
Swarm	586.5322	1.0149	/	0.939499
Swarm Random	575.7519	0.7220	/	0.733244
Swarm Decremento Lineare	575.7520	0.7342	/	0.667518
Swarm Decremento Non Lineare	581.1821	0.8484	/	0.799860

Modello di regressione polinomiale, d = 2

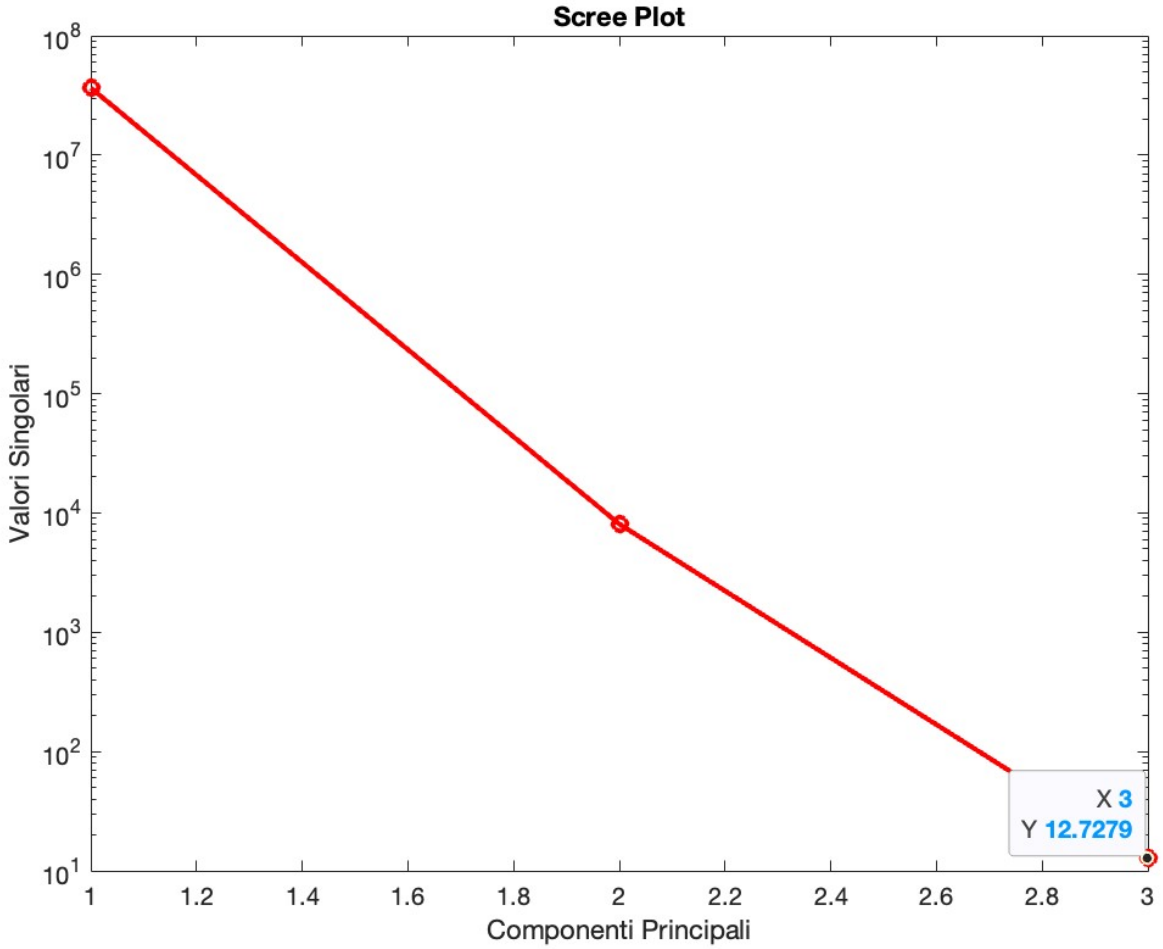
MATLAB

Rango della matrice di training: 3

Notiamo una RMSE rispetto al modello precedente per quanto riguarda i metodi: Normali, Thin QR, Thin QR con Pivoting, Guttman Keiser, Swarm Random, Swarm con Decremento Lineare e con Decremento Non Lineare, mentre per gli altri metodi abbiamo un peggioramento non indifferente.

In termini di tempo di esecuzione non abbiamo significativi peggioramenti, ma sono comunque presenti, tranne che per il criterio di Guttman Keiser che risulta essere quello più efficiente rispetto ai metodi Normali, Thin QR e Thin QR con Pivoting che in genere sono quelli che mostrano l'RMSE più bassa e un tempo di esecuzione eccezionale. Il metodo più efficiente di tutti è quello dell'Energia, tuttavia ha una RMSE molto alta.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	89.4861	0.0001	8260316083806.6846	0.000020
Thin QR	89.4861	0.0002	2874076.5598	0.000119
Thin QR Pivoting	89.4861	0.0004	2874076.5598	0.000083
Scree-Plot Cattel	452.7535	0.0003	4531.3776	/
Guttman Keiser	89.4861	0.0002	2874076.5598	0.000058
Energia	826.9908	0.0010	1	0.000013
Entropia	826.9908	0.0053	1	0.000184
Swarm	1917.5023	0.0371	/	0.029206
Swarm Random	452.3461	0.0358	/	0.028736
Swarm Decremento Lineare	358.2796	0.0482	/	0.031785
Swarm Decremento Non Lineare	544.2109	0.0395	/	0.031061



Ulteriori informazioni

- **k** scelto per lo **Scree-Plot di Cattel**: 2
- **k** scelto da **Guttman Keiser**: 3
- **k** scelto dal criterio dell'**energia** e dell'**entropia**: 1
- Vettore di permutazione qr pivoting: $[3, 2, 1]$

PYTHON

Notiamo un leggero cambiamento per quanto riguarda il condizionamento nelle equazioni Normali, ciò può essere dovuto a degli errori di arrotondamento.

L'RMSE è la stessa per tutti i metodi tranne che per quelli di PSO, che mostrano dei miglioramenti rispetto alle varianti in Matlab, tranne che per la Swarm Random.

I tempi di esecuzione di ciascun metodo risultano tutti più lenti rispetto ai corrispettivi in Matlab.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	89.4861	0.0007	8260316067790.5234	0.000208
Thin QR	89.4861	0.0004	2874076.5598	0.000223
Thin QR Pivoting	89.4861	0.0004	2874076.5598	0.000420
Scree-Plot Cattel	452.7535	0.0009	4531.3776	/
Guttman Keiser	89.4861	0.0004	2874076.5598	0.000136
Energia	826.9908	0.0002	1	0.000213
Entropia	826.9908	0.0015	1	0.000244
Swarm	1264.7550	1.0164	/	0.991547
Swarm Random	674.9617	1.0148	/	0.986112
Swarm Decremento Lineare	365.4132	1.0175	/	0.984560
Swarm Decremento Non Lineare	505.9911	0.8661	/	0.897360

Modello di regressione polinomiale, d = 3

MATLAB

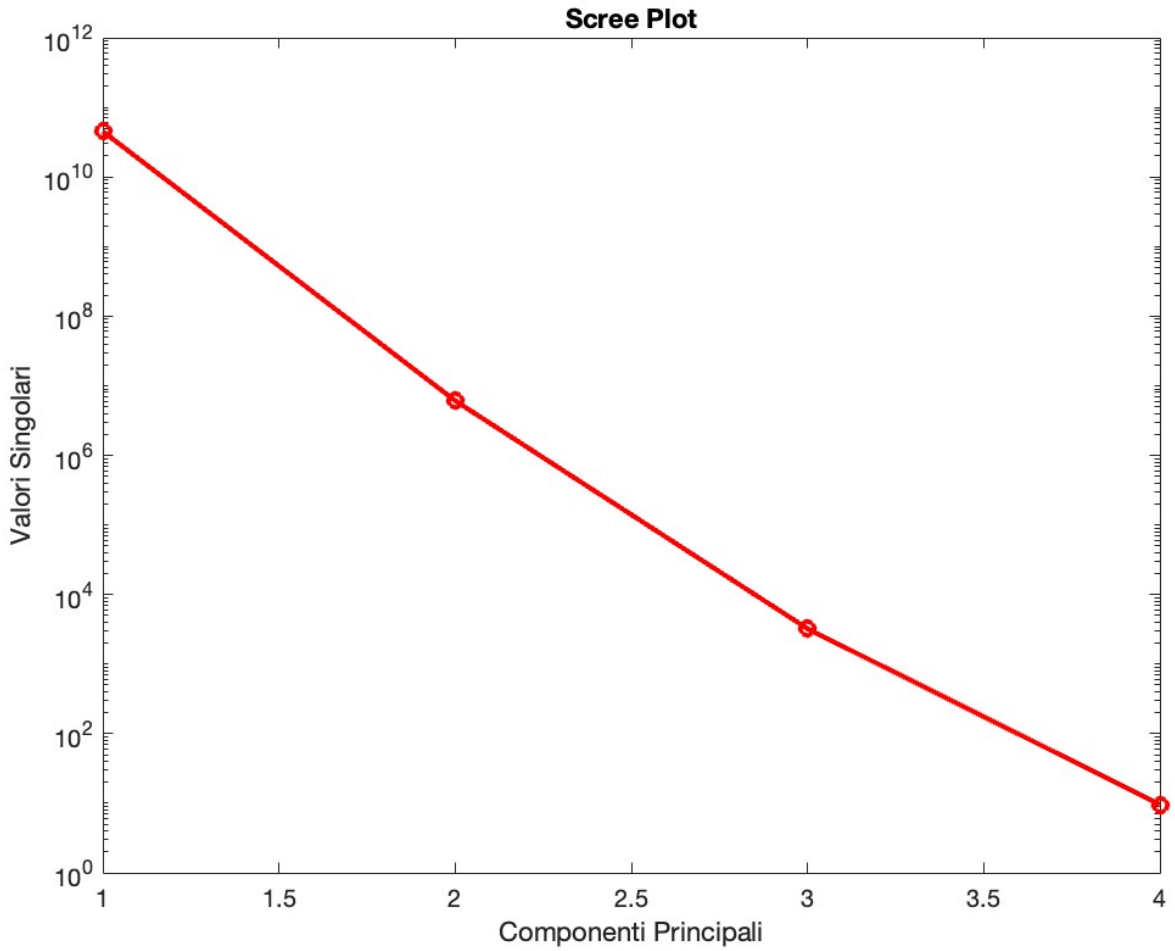
Rango della matrice di Training: 4

Valgono le stesse considerazioni fatte per il corrispettivo modello con target 'meantemp'.

Anche qui Thin QR con Pivoting è più efficiente della variante senza Pivoting.

Valgono le stesse considerazioni di prima per il criterio dell'Energia e di Guttman Keiser.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	90.7305	0.0025	22490009842337112064	0.003412
Thin QR	90.7305	0.0007	4742348076.0351	0.000255
Thin QR Pivoting	90.7305	0.0005	4742348080.3964	0.000115
Scree-Plot Cattel	433.6790	0.0004	14002282.9919	/
Guttman Keiser	90.7305	0.0001	4742348076.0351	0.000021
Energia	997.6470	0.0005	1	0.000017
Entropia	997.6470	0.0021	1	0.000172
Swarm	197417.5656	0.0560	/	0.029564
Swarm Random	144055.2330	0.0648	/	0.031051
Swarm Decremento Lineare	163826.8487	0.0497	/	0.041876
Swarm Decremento Non Lineare	600841.5540	0.0466	/	0.031137



Ulteriori informazioni

- **k** scelto per lo **Scree-Plot di Cattel**: 3
  - Scegliendo $k = 2$ abbiamo che:
    - RMSE = 790.3949
    - Tempo trascorso: 0.0001
    - Condizionamento: 7416.2212
- **k** scelto da **Guttman Keiser**: 4
- **k** scelto dal criterio dell'**energia** e dell'**entropia**: 1
- Vettore di permutazione qr pivoting: $[4, 3, 2, 1]$

PYTHON

Notiamo un cambiamento per quanto riguarda il condizionamento nei metodi con Normali, Thin qr con e senza pivoting, Scree plot e Guttman Keiser.

Tuttavia rispetto a Matlab qui la Thin QR con Pivoting risulta meno efficiente rispetto alla variante senza.

Notiamo anche qui la forte vatriabilità dell'RMSE per i metodi di PSO.

In questo caso le Normali risultano più efficienti rispetto a Matlab, mentre per il resto il tempo medio di esecuzione è peggiore.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	90.7305	0.0032	22490142494780456960	0.000558
Thin QR	90.7305	0.0004	4742348081.2452	0.000311
Thin QR Pivoting	90.7305	0.0008	4742348080.3965	0.000364
Scree-Plot Cattel	433.6790	0.0034	14002282.9854	/
Guttman Keiser	90.7305	0.0009	4742348081.2453	0.000117
Energia	997.6470	0.0002	1	0.000051
Entropia	997.6470	0.0013	1	0.000205
Swarm	496789.6687	1.0221	/	1.010819
Swarm Random	336680.5987	1.0501	/	1.011798
Swarm Decremento Lineare	254741.4193	0.9974	/	1.015924
Swarm Decremento Non Lineare	285891.7639	0.9351	/	0.934143

con $k = 2$:

- RMSE: 790.3949
- Tempo: 0.0058
- Condizionamento: 7416.2212

Modello di regressione polinomiale, d = 4

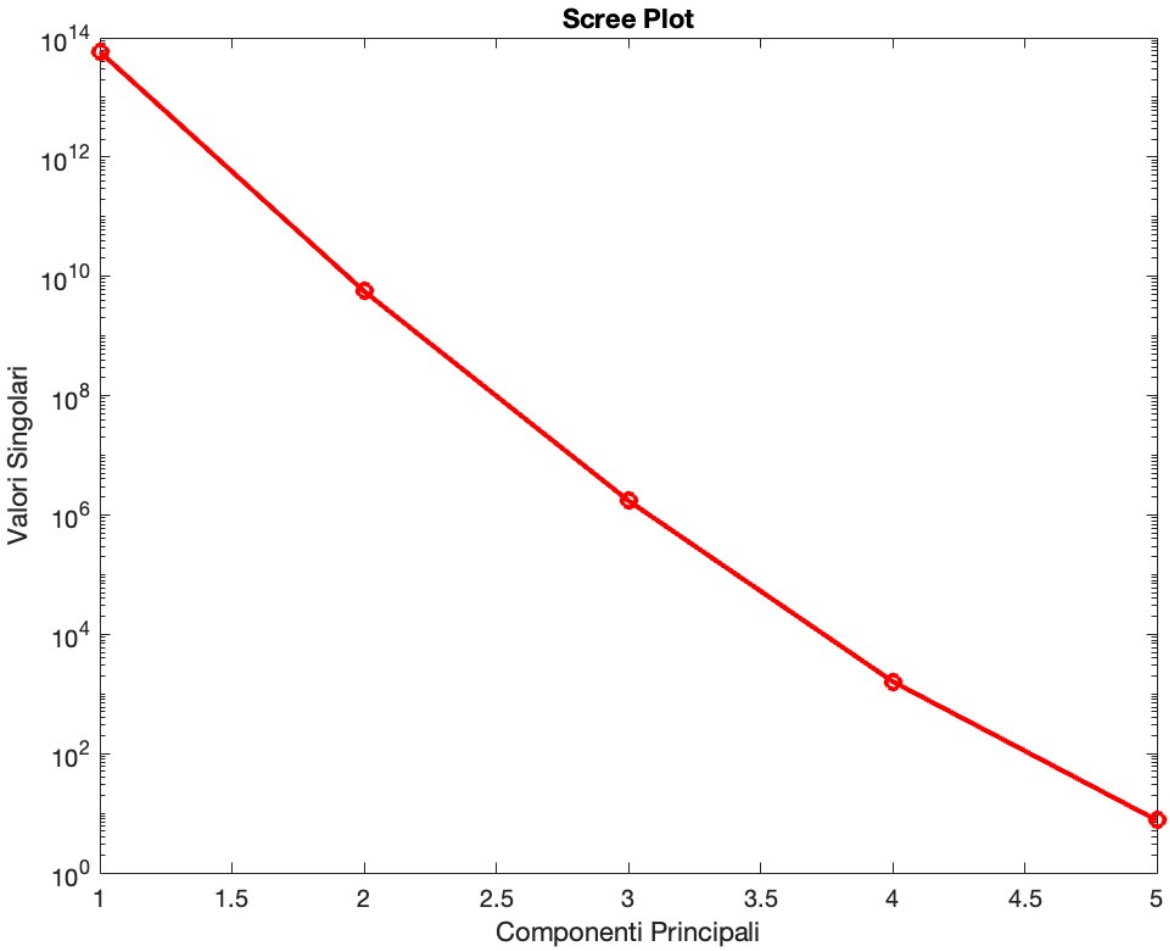
MATLAB

Rango della matrice di Training: 4, quindi la matrice non ha rango massimo e alcuni valori singolari sono nulli.

Anche qui valgono le stesse considerazioni fatte per lo stesso modello ma con target 'meantemp'.

Peggiora l'RMSE per ciascun metodo rispetto al modello precedente, tuttavia in termini di tempo medio di esecuzione abbiamo che le Normali vengono computate in maniera più efficiente, mentre per il resto c'è un peggioramento generale.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	97.8711	0.0015	77700313479421129543122944	0.00125
Thin QR	97.8711	0.0007	7657887098487.1289	0.00035
Thin QR Pivoting	97.8711	0.0005	7657829856730.1572	0.00013
Scree-Plot Cattel	522.6819	0.0002	36119589295.4145	/
Guttman Keiser	97.8720	0.0001	7657887098487.1279	0.00002
Energia	1141.7073	0.0002	1	0.00002
Entropia	1141.7073	0.0022	1	0.00034
Swarm	248519857.8514	0.0474	/	0.03006
Swarm Random	24153943.1808	0.0607	/	0.03451
Swarm Decremento Lineare	169444477.2716	0.0743	/	0.02991
Swarm Decremento Non Lineare	632667.6384	0.0419	/	0.03283



Il grafico tuttavia non sembra essere coerente con ciò che è stato detto in precedenza, molto probabilmente trattandosi di una matrice molto mal condizionata, ci saranno stati sicuramente diversi errori di arrotondamento.

Ulteriori informazioni

- **k** scelto per lo **Scree-Plot di Cattel**: 4
  - Scegliendo $k = 2$ abbiamo che:
    - RMSE = 1102.3002
    - Tempo trascorso: 0.0001
    - Condizionamento: 10318.4287
  - Scegliendo $k = 3$ abbiamo che
    - RMSE: 831.4798
    - Tempo trascorso: 0.0002

        - Condizionamento: 33479063.4719

- **k** scelto da **Guttman Keiser**: 5
- **k** scelto dal criterio dell'**energia** e dell'**entropia**: 1
- Vettore di permutazione qr pivoting: $[5, 4, 3, 2, 1]$

PYTHON

Notiamo che il criterio di Guttman Keiser sceglie $k = 5$ e per lo Scree Plot $k = 4$, ma l'RMSE in Python differisce da quella in Matlab di poco, molto probabilmente questo comportamento può essere legato agli errori di arrotondamento

Notiamo anche che il condizionamento in Normali, Thin qr con e senza pivoting, Scree Plot di Cattel e Guttman Keiser, differisce rispetto a quello che abbiamo in Matlab.

Evidente anche la variabilità dell'RMSE per i metodi di PSO e per quanto riguarda il tempo di esecuzione di ciascun metodo c'è comunque un peggioramento generale tranne che per i metodi con Normali e Thin QR.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E
Normali	97.8711	0.0009	52792673932931379167756288	0.0002
Thin QR	97.8711	0.0004	7657841198171.7930	0.0003
Thin QR Pivoting	97.8711	0.0005	7657829856730.1201	0.0002
Scree-Plot Cattel	522.6804	0.0012	36119753800.8594	/
Guttman Keiser	97.8731	0.0003	7657841198125.8379	0.0002
Energia	1141.7073	0.0002	1	0.0000
Entropia	1141.7073	0.0005	1	0.0002
Swarm	3059350972.2675	1.0096	/	1.0298
Swarm Random	33232399.0214	1.1107	/	1.0471
Swarm Decremento Lineare	655121986.5397	1.0408	/	1.0252
Swarm Decremento Non Lineare	352272068.2617	0.9471	/	0.9463

con $k = 2$: (invariato rispetto a matlab)

- RMSE: 1102.3002
- Tempo: 0.0021
- Condizionamento: 10318.4287

con $k = 3$: (cambia solo il condizionamento)

- RMSE: 831.4798
- Tempo: 0.0023
- Condizionamento: 33479063.4723

Dati normalizzati

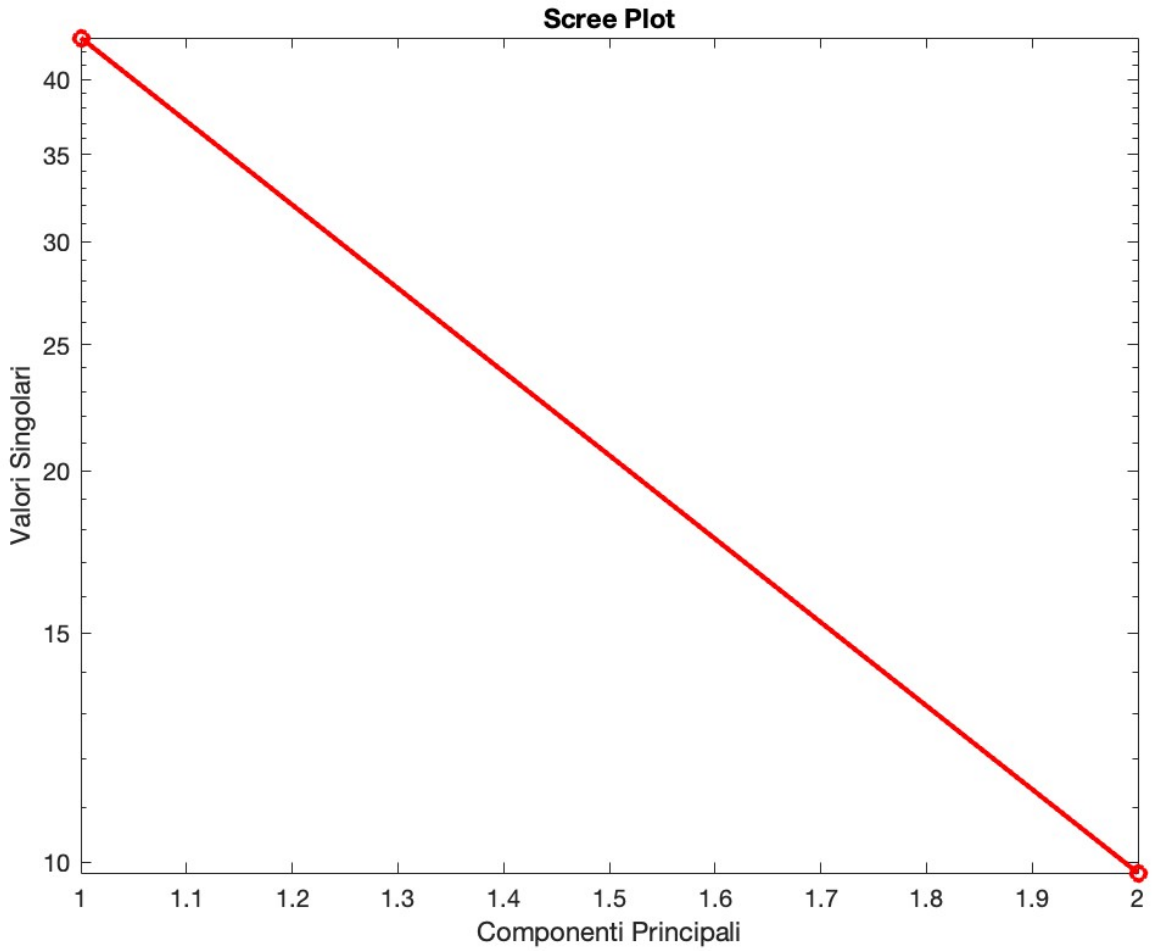
Modello di regressione lineare

MATLAB

Rango della matrice di Training: 2

Valgono le stesse osservazioni fatte per lo stesso modello con dati normalizzati sulla feature 'meantemp'.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	0.0117	0.0001	19.2559	0.000022
Thin QR	0.0117	0.0001	4.3882	0.000101
Thin QR Pivoting	0.0117	0.0004	4.3882	0.000048
Scree-Plot Cattel	0.0117	0.0002	4.3882	/
Guttman Keiser	0.0117	0.0002	4.3882	0.000020
Energia	0.0309	0.0003	1	0.000016
Entropia	0.0309	0.0013	1	0.000136
Swarm	0.0133	0.0447	/	0.029502
Swarm Random	0.0117	0.0295	/	0.027693
Swarm Decremento Lineare	0.0117	0.0392	/	0.021797
Swarm Decremento Non Lineare	0.0117	0.0371	/	0.029001



Ulteriori informazioni

- **k** scelto in **Scree-Plot di Cattel**, **Guttman Keiser**: 2
- **k** scelto dal criterio dell'**energia** e dell'**entropia**: 1
- Vettore di permutazione qr pivoting: **ordinato**

PYTHON

Rispetto a Matlab, l'unico risultato in termini di RMSE che cambia è quello della Swarm con velocità non pesate, notiamo appunto un leggero peggioramento. Mentre lli altri metodi di PSO sembrano stabili.

In termini di tempo medio di esecuzione notiamo un generale peggioramento rispetto a Matlab, oltre al fatto che la Thin QR con Pivoting risulta più lenta rispetto alla variante senza, mentre in Matlab possiamo notare il contrario.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	0.0117	0.0006	19.2559	0.000230
Thin QR	0.0117	0.0004	4.3882	0.000278
Thin QR Pivoting	0.0117	0.0003	4.3882	0.000382
Scree-Plot Cattel	0.0117	0.0029	4.3882	/
Guttman Keiser	0.0117	0.0010	4.3882	0.000104
Energia	0.0309	0.0002	1	0.000044
Entropia	0.0309	0.0004	1	0.000257
Swarm	0.0137	0.9784	/	0.977900
Swarm Random	0.0117	0.7186	/	0.761471
Swarm Decremento Lineare	0.0117	0.7150	/	0.751784
Swarm Decremento Non Lineare	0.0117	0.8118	/	0.843186

Modello di regressione polinomiale, d = 2

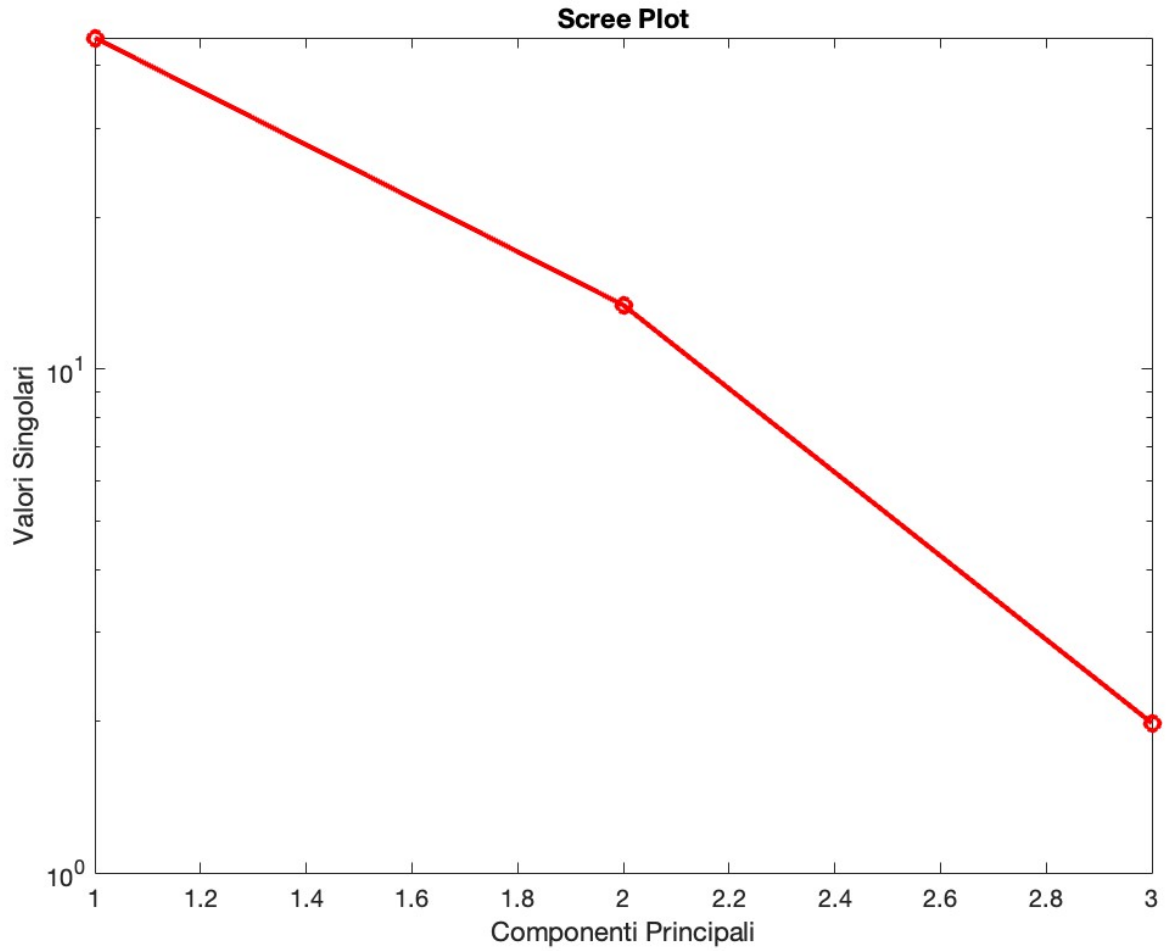
MATLAB

Rango della matrice di training: 3

Per quanto riguarda la RMSE per ciascun metodo, questa risulta leggermente migliore rispetto al modello lineare discusso in precedenza; ma per i metodi di Energia, Entropia, Swarm con velocità non pesate e Swarm con decremento non lineare, quest'ultima affermazione è falsa.

In generale non ci sono significativi peggioramenti in termini di tempo medio di esecuzione.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	0.0116	0.0002	524.1339	0.000021
Thin QR	0.0116	0.0002	22.8940	0.000147
Thin QR Pivoting	0.0116	0.0005	22.8940	0.000067
Scree-Plot Cattel	0.0116	0.0002	3.3917	/
Guttman Keiser	0.0116	0.0001	22.8940	0.000018
Energia	0.0510	0.0002	1	0.000012
Entropia	0.0510	0.0034	1	0.000143
Swarm	0.0233	0.0415	/	0.030951
Swarm Random	0.0116	0.0336	/	0.030286
Swarm Decremento Lineare	0.0116	0.0461	/	0.025802
Swarm Decremento Non Lineare	0.0252	0.0568	/	0.032503



Ulteriori informazioni

- **k** scelto per lo **Scree-Plot di Cattel**: 2
- **k** scelto da **Guttman Keiser**: 3
- **k** scelto dal criterio dell'**energia** e dell'**entropia**: 1
- Vettore di permutazione qr pivoting: $[1, 3, 2]$

PYTHON

In questo caso sembra venire meno la stabilità della swarm con velocità non pesate, swarm con decremento lineare e swarm con decremento non lineare: sono evidenti dei leggeri peggioramenti.

Il tempo medio di esecuzione di ciascun metodo in Python è più lento del corrispettivo in Matlab.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	0.0116	0.0030	524.1339	0.000154
Thin QR	0.0116	0.0005	22.8940	0.000220
Thin QR Pivoting	0.0116	0.0005	22.8940	0.000271
Scree-Plot Cattel	0.0116	0.0008	3.3917	/
Guttman Keiser	0.0116	0.0002	22.8940	0.000106
Energia	0.0510	0.0001	1	0.000049
Entropia	0.0510	0.0008	1	0.000138
Swarm	0.0294	0.9828	/	0.992238
Swarm Random	0.0116	0.8998	/	0.960243
Swarm Decremento Lineare	0.0117	0.9090	/	0.853199
Swarm Decremento Non Lineare	0.0370	0.8589	/	0.858411

Modello di regressione polinomiale, d = 3

MATLAB

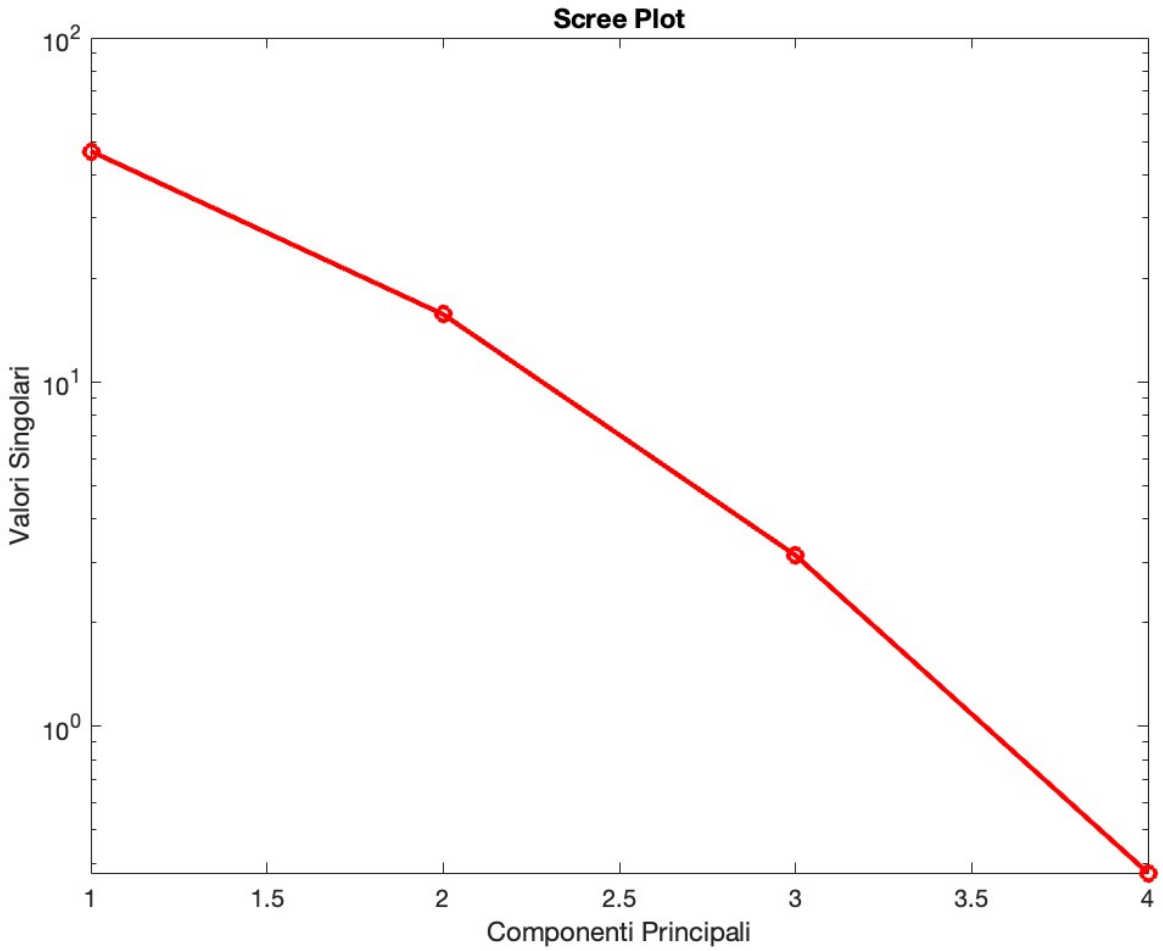
Rango della matrice di Training: 4

In generale c'è un leggero peggioramento per quanto riguarda l'RMSE tranne per quanto riguarda lo Scree-Plot di Cattel e il criterio di Guttman-Keiser dove il k selezionato è pari a 3, quindi mostrano una RMSE più bassa rispetto agli altri metodi, oltre che a un ottimo tempo medio di esecuzione almeno per quanto riguarda il criterio di Guttman Keiser.

L'altro metodo che offre ottimi tempi di esecuzione è quello dell'Energia, che dà comunque una RMSE piuttosto accettabile rispetto ad altri metodi come Entropia e quelli di PSO; ciò non vale per il metodo swarm con decremento non lineare che mostra una RMSE più bassa rispetto a metodi come Normali e Thin QR, ma dall'altra parte ha tempi di esecuzione più lunghi.

Buono il condizionamento per i criteri utilizzati per la regressione alle componenti principali.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	0.0118	0.0002	15536.1484	0.000021
Thin QR	0.0118	0.0003	124.6441	0.000251
Thin QR Pivoting	0.0118	0.0008	124.6441	0.000102
Scree-Plot Cattel	0.0116	0.0004	14.9176	/
Guttman Keiser	0.0116	0.0002	14.9176	0.000019
Energia	0.0123	0.0009	2.9771	0.000016
Entropia	0.0683	0.0018	1	0.000204
Swarm	0.0993	0.0441	/	0.029717
Swarm Random	0.2605	0.0405	/	0.031510
Swarm Decremento Lineare	0.0166	0.0375	/	0.025226
Swarm Decremento Non Lineare	0.0117	0.0363	/	0.029858



Ulteriori informazioni

- **k** scelto per lo **Scree-Plot di Cattel**: 3
- **k** scelto da **Guttman Keiser**: 3
    - infatti il quarto valore singolare è strettamente minore di 1
- **k** scelto dal criterio dell'**energia**: 2
- **k** scelto dal criterio dell'**entropia**: 1
- con $k = 4$:
    - RMSE: 0.0118
    - Tempo: 0.0005
    - Condizionamento: 124.6441
- Vettore di permutazione qr pivoting: $[1, 3, 2, 4]$

PYTHON

Notiamo che qui cambia l'RMSE per ciascun metodo di PSO, e i tempi di esecuzione di ciascun metodo in Python sono più lenti rispetto ai corrispettivi in Matlab.

	RMSE	TEMPO	CONDIZIONAMENTO	TEMPO MEDIO DI ESECUZIONE
Normali	0.0118	0.0006	15536.1484	0.000197
Thin QR	0.0118	0.0017	124.6441	0.000242
Thin QR Pivoting	0.0118	0.0008	124.6441	0.000259
Scree-Plot Cattel	0.0116	0.0006	14.9176	/
Guttman Keiser	0.0116	0.0002	14.9176	0.000141
Energia	0.0123	0.0002	2.9771	0.000088
Entropia	0.0683	0.0004	1	0.000183
Swarm	0.0901	1.0155	/	0.980390
Swarm Random	0.0223	1.0065	/	1.017576
Swarm Decremento Lineare	0.0124	0.9013	/	0.905422
Swarm Decremento Non Lineare	0.0384	0.9255	/	0.888239

con $k = 4$:

- RMSE: 0.0118
- Tempo: 0.0020
- Condizionamento: 124.6441

Modello di regressione polinomiale, d = 4

MATLAB

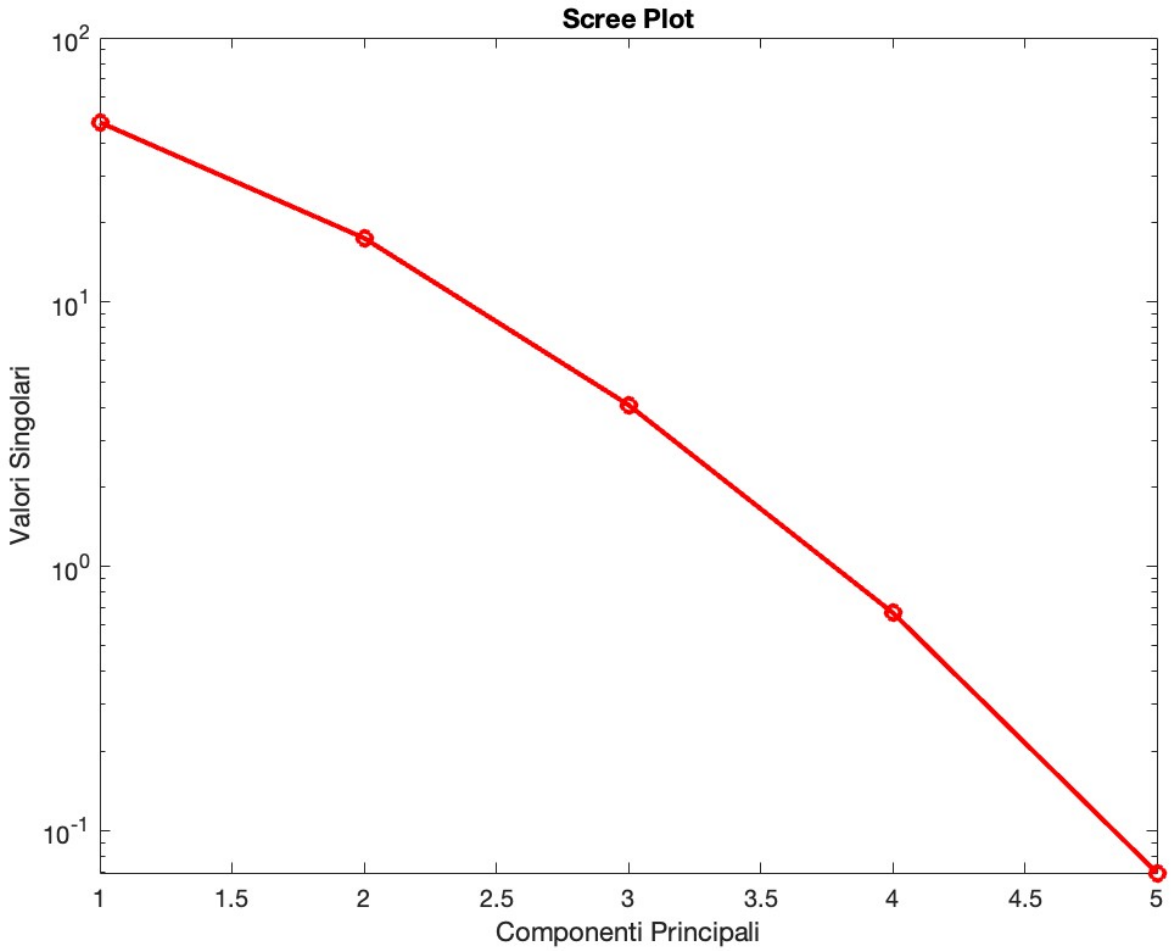
Rango della matrice di Training: 5

Rispetto al modello precedente peggiora l'RMSE e i tempi di esecuzione risultano leggermente più lunghi. Sorprendentemente i metodi di PSO non hanno prodotto una RMSE così tanto alta rispetto agli altri metodi, anzi nel caso della random abbiamo la migliore in questo contesto. Tuttavia bisognerebbe valutare un trade-off tra tempo medio di esecuzione ed RMSE, quindi anche gli altri metodi vanno abbastanza bene.

In termini di tempo medio di esecuzione c'è un leggero peggioramento rispetto al modello precedente.

Buono il condizionamento per i criteri utilizzati per la regressione alle componenti principali.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	0.0127	0.0002	477661.1552	0.000028
Thin QR	0.0127	0.0002	691.1303	0.000293
Thin QR Pivoting	0.0127	0.0006	691.1303	0.000116
Scree-Plot Cattel	0.0120	0.0002	71.5992	/
Guttman Keiser	0.0116	0.0001	11.7158	0.000022
Energia	0.0141	0.0003	2.7403	0.000015
Entropia	0.0141	0.0023	2.7403	0.000144
Swarm	0.1302	0.0411	/	0.030803
Swarm Random	0.0117	0.0396	/	0.033097
Swarm Decremento Lineare	0.0134	0.0359	/	0.027627
Swarm Decremento Non Lineare	0.0344	0.0391	/	0.032888



Ulteriori informazioni

- **k** scelto per lo **Scree-Plot di Cattel**: 4
- **k** scelto da **Guttman Keiser**: 3
  - infatti gli ultimi due valori singolari sono strettamente minori di 1
- **k** scelto dal criterio dell'**energia** e dal criterio dell'**entropia**: 2
- con $k = 1$:
  - RMSE: 0.0838
  - Tempo: 0.0008
  - Condizionamento: 1
- con $k = 5$:
  - RMSE: 0.0127

    - Tempo: 0.0001
    - Condizionamento: 691.1303
- Vettore di permutazione qr pivoting: $[1, 3, 5, 2, 4]$

PYTHON

In questo caso cambia l'RMSE di ciascun metodo di PSO, infatti notiamo un peggioramento rispetto a Matlab.

Il resto dei risultati rimane invariato, l'unica componente che cambia è il tempo medio di esecuzione per ciascun metodo che anche in questo caso risulta essere più lento rispetto al corrispettivo in Matlab.

	RMSE	TEMPO	CONDIZIONAMENTO	T.M.E.
Normali	0.0127	0.0005	477661.1552	0.000271
Thin QR	0.0127	0.0004	691.1303	0.000410
Thin QR Pivoting	0.0127	0.0003	691.1303	0.000305
Scree-Plot Cattel	0.0120	0.0012	71.5992	/
Guttman Keiser	0.0116	0.0003	11.7158	0.000096
Energia	0.0141	0.0002	2.7403	0.000090
Entropia	0.0141	0.0005	2.7403	0.000234
Swarm	0.0718	1.0366	/	1.035974
Swarm Random	0.0278	1.0476	/	1.020797
Swarm Decremento Lineare	0.0541	0.9610	/	1.146274
Swarm Decremento Non Lineare	0.0695	0.9303	/	1.153912

con $k = 1$:

- RMSE: 0.0838
- Tempo: 0.0026
- Condizionamento: 1

con $k = 5$:

- RMSE: 0.0127
- Tempo: 0.0007
- Condizionamento: 691.1303

Conclusioni

In questo caso di studio abbiamo considerato 4 modelli di regressione, per ciascuno dei quali sono stati valutati i metodi in base all'RMSE, al condizionamento e al tempo medio di esecuzione. Abbiamo cominciato con la nostra valutazione, andando a considerare prima una distribuzione di dati non normalizzata e abbiamo notato come il condizionamento fosse troppo elevato, quindi avevamo un problema molto mal condizionato, e di conseguenza lo erano anche i valori di RMSE per ciascun metodo di ciascun modello.

Aumentando sempre più la dimensione della matrice di input, abbiamo notato come il condizionamento crescesse, soprattutto andando a considerare il caso delle equazioni Normali, e nei casi in cui il grado del polinomio $d = 3$ e $d = 4$ abbiamo visto come la matrice AA^T fosse quasi singolare e di conseguenza portava a dei problemi di instabilità numerica, infatti conseguentemente abbiamo visto come l'RMSE crescesse.

Nel momento in cui siamo andati a normalizzare la nostra distribuzione di dati, abbiamo subito notato come le matrici fossero ben condizionate e come l'RMSE fosse sceso.

Pertanto risulta evidente la relazione fra condizionamento ed RMSE, dal momento che quest'ultima misura mostra una forte sensibilità al cambiamento dei dati: siamo partiti da un problema mal condizionato nei dati di training, le soluzioni non si adattano bene ai dati di test, perciò questo può comportare l'aumento dell'RMSE, proprio perché le previsioni del modello saranno meno accurate.

Per confrontare ciascun modello ci siamo basati sui risultati che ciascun suo metodo ha prodotto, pertanto ne risulta che i modelli che hanno fornito una RMSE migliore rispetto agli altri sono quelli di regressione lineare e quello di regressione polinomiale con grado del polinomio pari a 2.

Ovviamente sulla scelta del modello bisognerebbe fare un trade-off fra lo spazio e il tempo che si risparmia scegliendo un modello di regressione lineare, rispetto all'accuratezza delle predizioni dell'altro modello.

In termini di scelta per il metodo in base all'accuratezza, i metodi che non hanno mai tradito sono stati quelli basati su equazioni Normali, fattorizzazione Thin QR con e senza pivoting. In generale abbiamo notato che in Matlab il metodo con pivoting avesse dei tempi di esecuzione migliori rispetto a quelli della variante senza, mentre con Python abbiamo l'esatto opposto.

Se dovessimo considerare anche il numero di condizionamento preferiremmo la fattorizzazione QR rispetto al metodo basato sulle equazioni Normali, per quanto poi le Normali hanno in genere un tempo di esecuzione minore.

Abbiamo notato anche che ci sono stati casi in cui i metodi usati per la regressione alle componenti principali non solo si sono rivelati fortemente efficienti rispetto ai metodi sopracitati, ma hanno prodotto anche una RMSE o uguale o minore, tuttavia ciò non è stato sempre valido. Anche in questo caso andrebbe fatto un trade-off tra quello che è il tempo di esecuzione che ti dà un metodo per la regressione alle componenti principali, e l'accuratezza delle predizioni ottenute da Normali, Thin QR con e senza Pivoting.

Per quanto riguarda gli algoritmi di PSO, questi sono stati fortemente instabili, relativamente ai modelli in cui non è stata applicata la normalizzazione dei dati; tuttavia nel momento in cui abbiamo normalizzato i dati, è sembrato che, almeno per i primi due modelli (d=1, d=2), questi algoritmi fossero più stabili e producessero delle previsioni molto più accurate, ma questa regola non è stata confermata per i successivi modelli, e, inoltre, in Python abbiamo ottenuto dei risultati differenti. Perciò potremmo pensare che per questo specifico dominio quei metodi non fossero così tanto adatti al nostro scopo, anche perché non solo hanno prodotto una RMSE molto variabile, ma avevano dei tempi di esecuzione, che paragonati agli altri modelli, erano in grado di far sorridere per la loro giustificata lentezza, che in Python è risultata molto più evidente.

Un'ultima osservazione è che generalmente si è dimostrato come le routines implementate in Python siano state sperimentalmente più lente rispetto a quelle in Matlab, e talvolta producevano comunque dei risultati in termini di RMSE, che erano praticamente gli stessi, sebbene cambiasse leggermente anche il condizionamento.

Chiaramente non vogliamo concludere dicendo che un linguaggio sia meglio di un altro, ma che ognuno sia pensato appunto per i suoi specifici scopi, e la dimostrazione è data da questa caso di studio, che non può essere generalizzato ma preso in considerazione come un esempio.