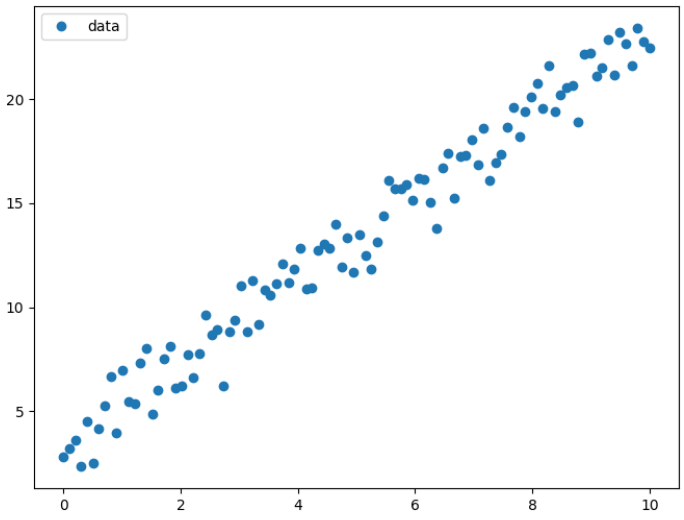
COURSE NAME: Hillel - Machine Learning. Lesson 2. (02.03.2023 19:15)
LECTURER: Yevhen Krasnokutsky
STUDENT: Maryna Lysenko

HW 2 REPORT “Linear Regression”

Introduction

Ця лабораторна робота присвячена задачі лінійної регресії. Для експериментів було згенеровано випадкові дані, що відповідають лінійній залежності між змінною X та змінною Y з деяким додатковим шумом. Конкретніше, змінна X є рівномірно розподіленою на відрізку від 0 до 10 з 100 рівномірно розподіленими значеннями. Змінна Y визначається формулою $y = 2 * x + 3.5$ з додаванням гаусового шуму. Шум було згенеровано за допомогою випадкової величини з нормальним розподілом з середнім значенням рівним 0 та стандартним відхиленням рівним 1.



Отже, експериментальні дані відповідають лінійній залежності з невеликим додатковим шумом, що робить їх придатними для застосування методів лінійної регресії.

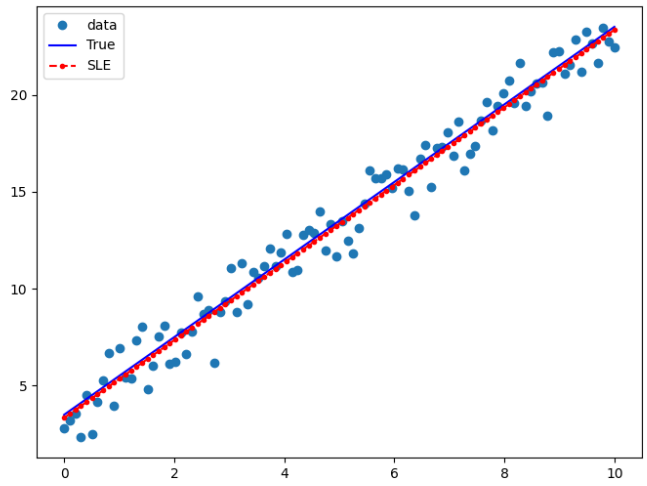
Метод 1.SLE

Потім проводиться розв'язання задачі лінійної регресії за допомогою методу SLE. Для цього обчислюється матриця X, що містить значення x та константу 1, і обернена до неї матриця Xinv. Потім розв'язується рівняння Xa=y, де a - вектор параметрів моделі, а m та b - параметри лінійної регресії.

Знайдені параметри:

Parameters: 2.00057348731401
3.3425098845082033

$SLE = mx + b$ — отримана модель

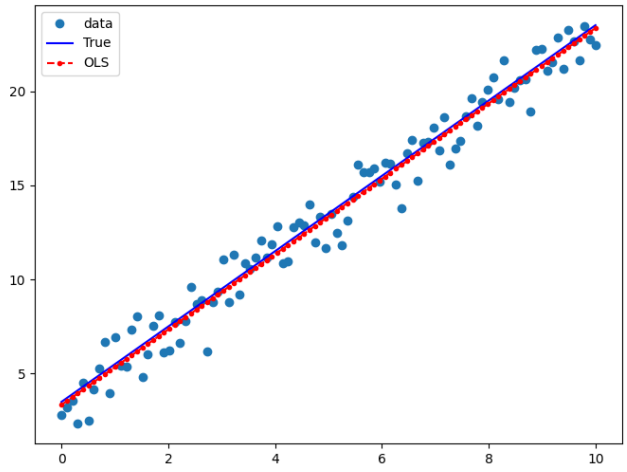


Метод 2. Least Squares Method

Модель OLS Regression (реалізація `statsmodels`),

Results: 2.00057349; 3.34250988

R2: 0.9707819205904828

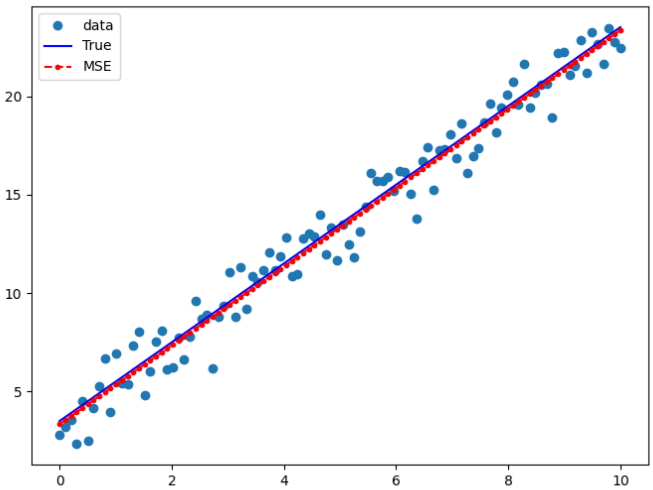


Метод 3. MSE

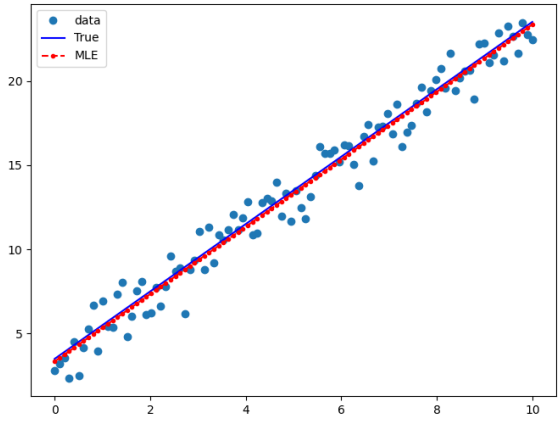
Мінімізуємо MSE

Parameters: 2.00058069; 3.3425008

Зійшлась за 55 ітерацій



Метод 4. Maximum Likelihood Estimation (MLE) regression using the Nelder-Mead optimization algorithm



Мінімізуємо функцію максимальної правдоподібності

Parameters:

[2.00057341 3.34251743 1.01195181]
(m,b,sigma)

Зійшлась за 97 ітерацій

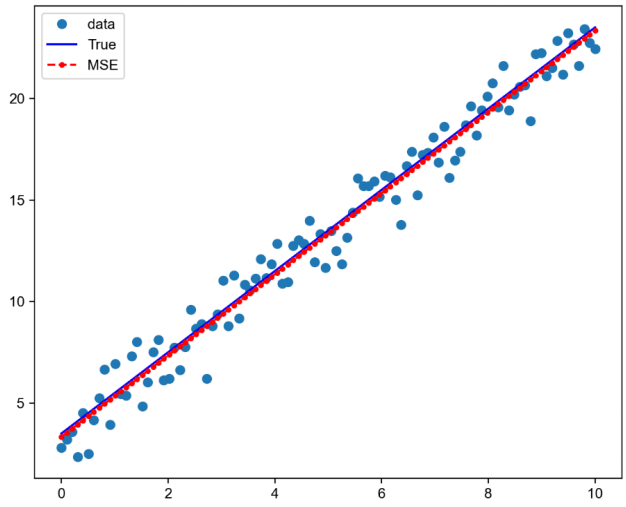
Виконання домашньої роботи:

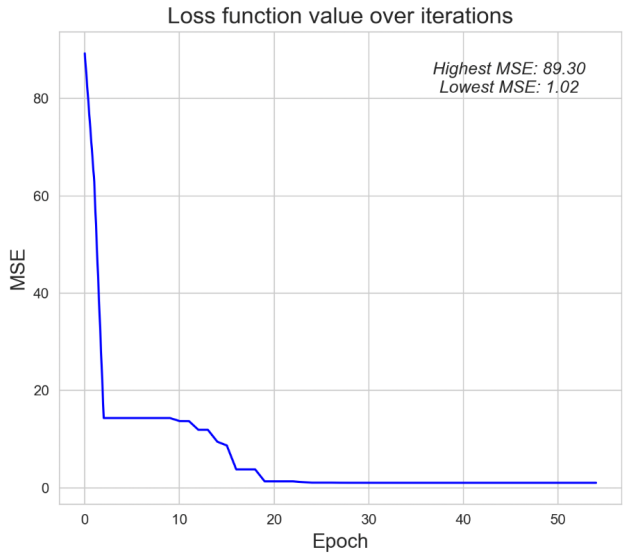
Нагадаємо базові параметри і результати в демо тесту з методу №3 (MSE)

Експеримент 0:

- Архітектура: y_hat = m * x + b
- method="Nelder-Mead"
- loss = MSE = (np.square(y - y_hat)).mean()
- Зійшлась за 55 ітерацій
- Highest MSE: 89.29567546107769
- Lowest MSE: 1.0241037044214218
- Parameters: 2.00058069; 3.3425008
- Parameters real: 2; 3.5

До методу 3 (MSE) додаємо також побудову графіку loss function для кожної ітерації, з 20 ітерації модель майже зійшлась.





А також згенерована анімація процессу навчання (`fit_animation.gif`)

Частина 1. Експерименти з Loss function

- Parameters real: 2; 3.5
- Архітектура: y_hat = m * x + b
- method="Nelder-Mead"

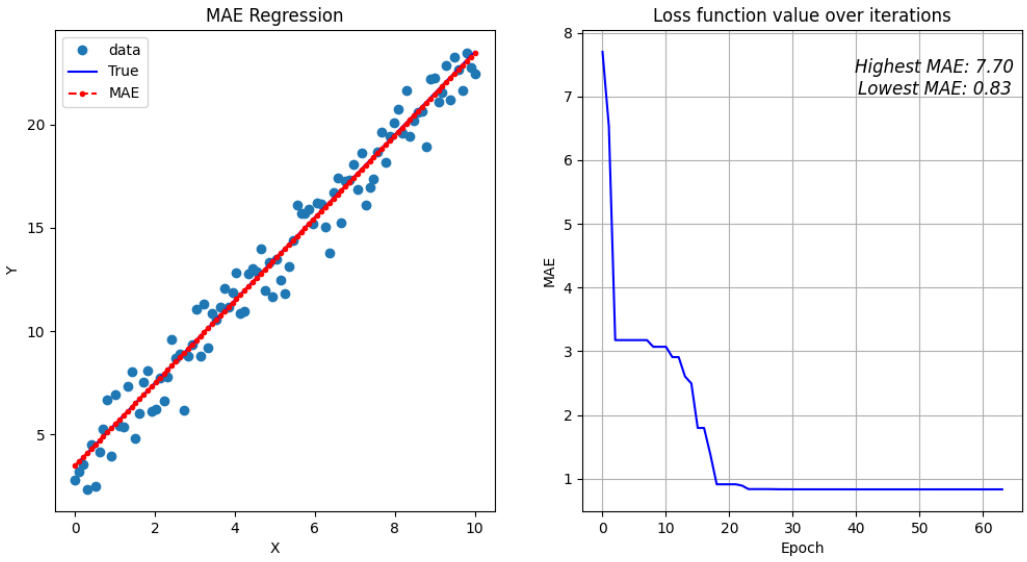
MSE

- loss = MSE = (np.square(y - y_hat)).mean()
- Зійшлась за 55 ітерацій
- Highest MSE: 89.29567546107769
- Lowest MSE: 1.0241037044214218
- Parameters: 2.00058069; 3.3425008

(графіки вище)

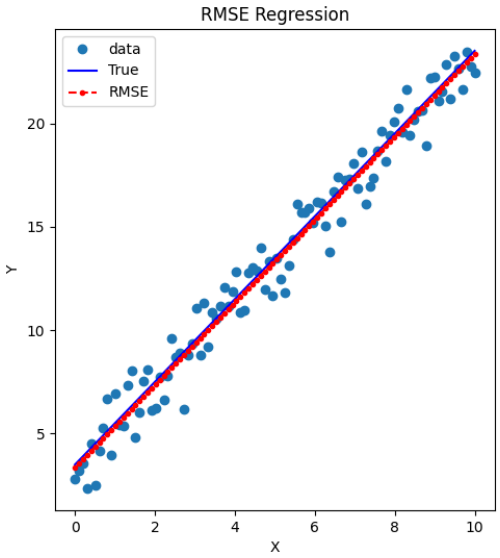
MAE:

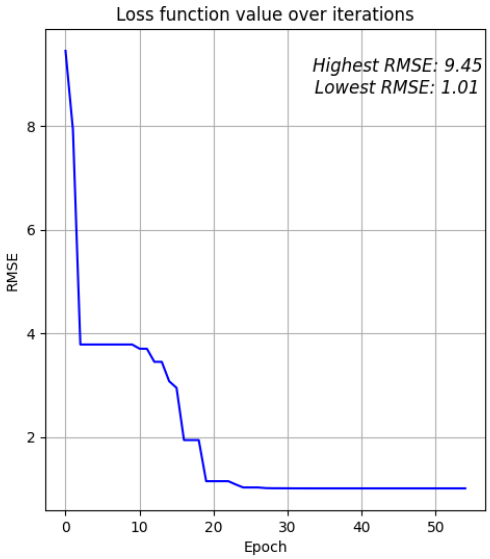
- Parameters (MAE): [1.99623445 3.50454274]
- Зійшлась за 64 ітерацій
- Highest loss: 7.7001695759455515
- Lowest loss: 0.8305578476764989



RMSE:

- Parameters (RMSE): [2.00058069 3.3425008]
- Зійшлась за 55 ітерацій
- Highest loss: 9.449638906385667
- Lowest loss: 1.0119800909214676





Conclusions 1

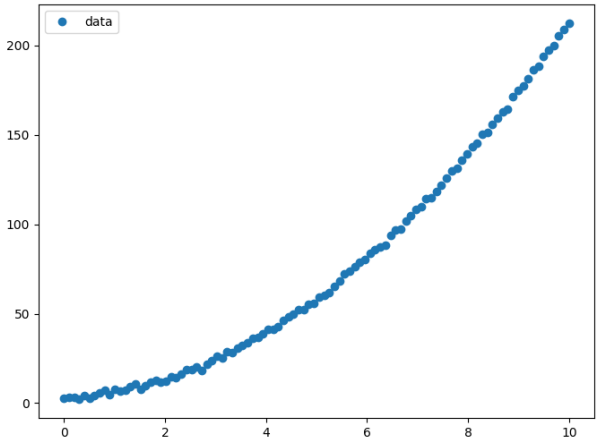
Загалом можна зробити висновок, що усі три функції втрат змогли сходитися до подібного набору параметрів, Отже, якщо ми зосереджуємось на точності передбачення, то MSE. RMSE є кращим вибором (бистріше зійшлись, маленька похибка), але якщо нам потрібно більше ваги приділити великим відхиленням, то MAE може бути кращим варіантом.

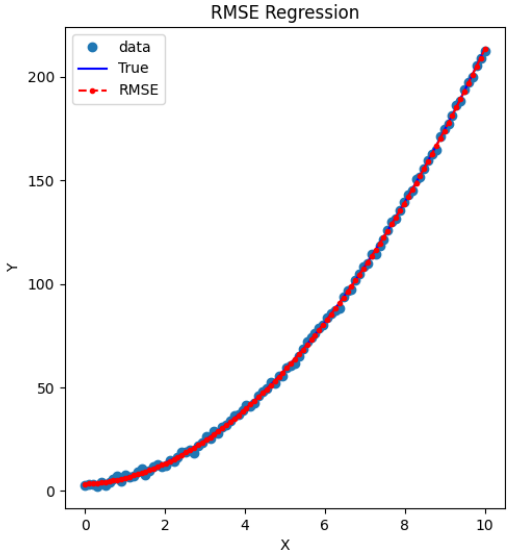
Частина 2. Експерименти з нелінійними даними

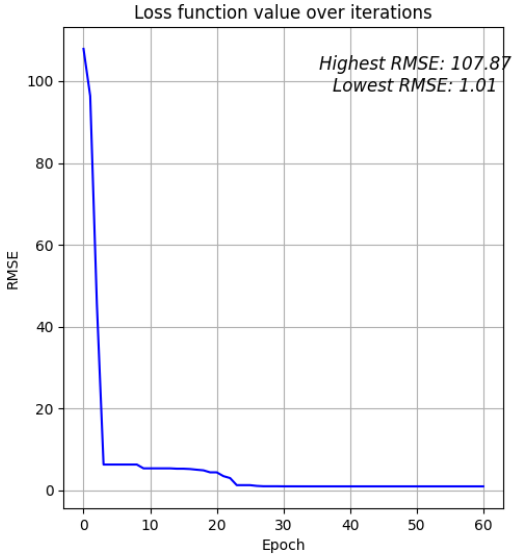
Experiment with non-linear data

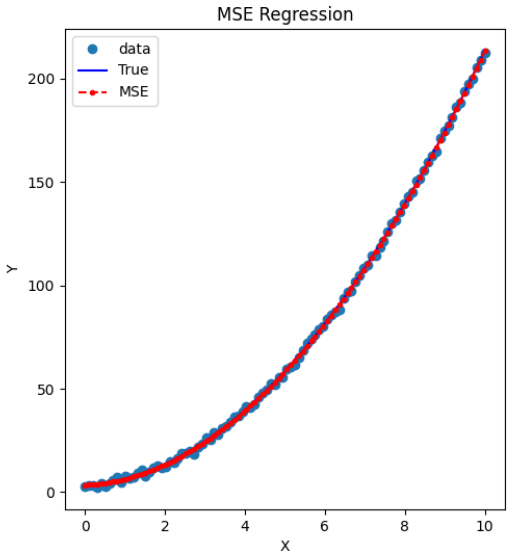
y = 2 * x**2 + x + 3.5 + noise

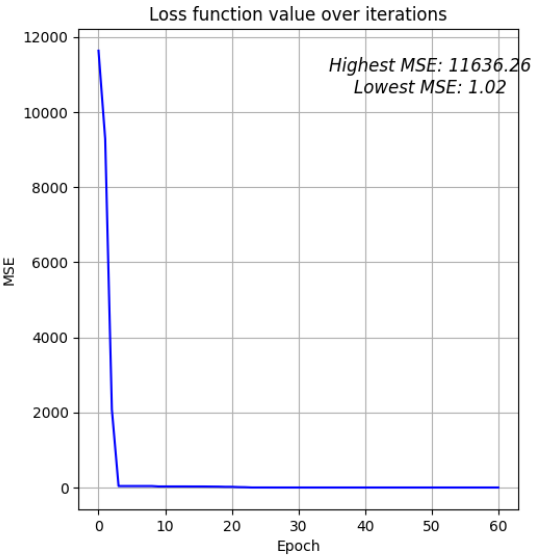
y_hat = m * x**2 + x + b

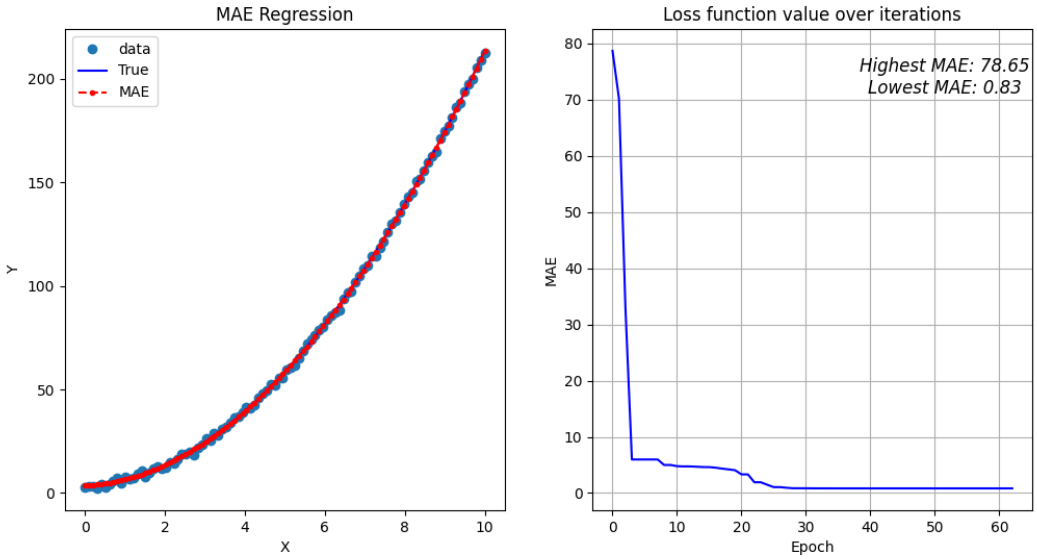



RMSE Regression
data
True
RMSE
X
Y


Loss function value over iterations
Highest RMSE: 107.87
Lowest RMSE: 1.01
RMSE
Epoch


MSE Regression
data
True
MSE
X
Y


Loss function value over iterations
Highest MSE: 11636.26
Lowest MSE: 1.02
MSE
Epoch



При зміні даних на нелінійні і зміні архітектури також досить вдало приближені дані всіма 3 методами. Найвдаліше MAE

Parameters (RMSE): [1.99932673 3.36795436] Iterations: 61

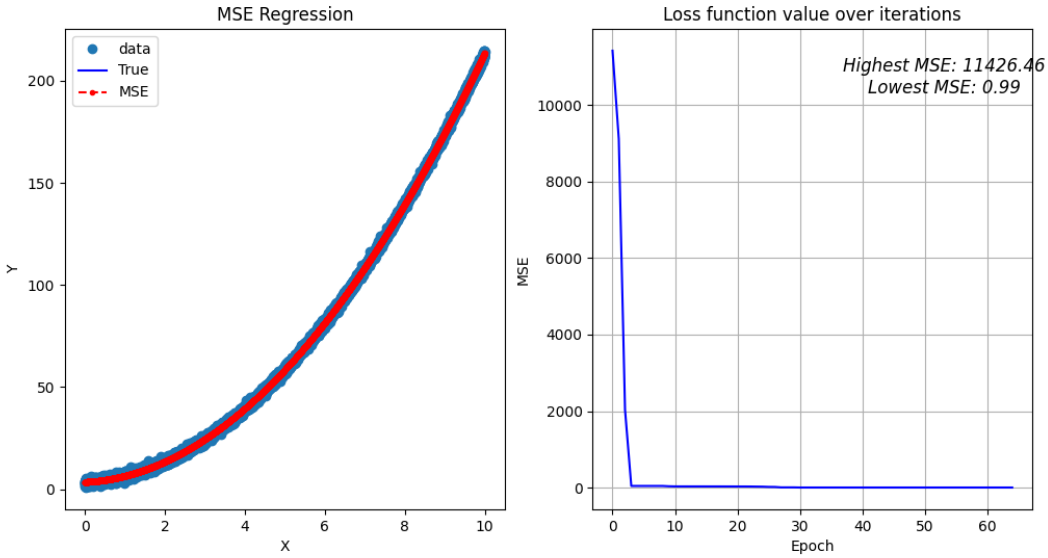
Parameters (MAE): [1.99874703 3.50691797] Iterations: 63

Parameters (MSE): [1.99932673 3.36795436] Iterations: 61

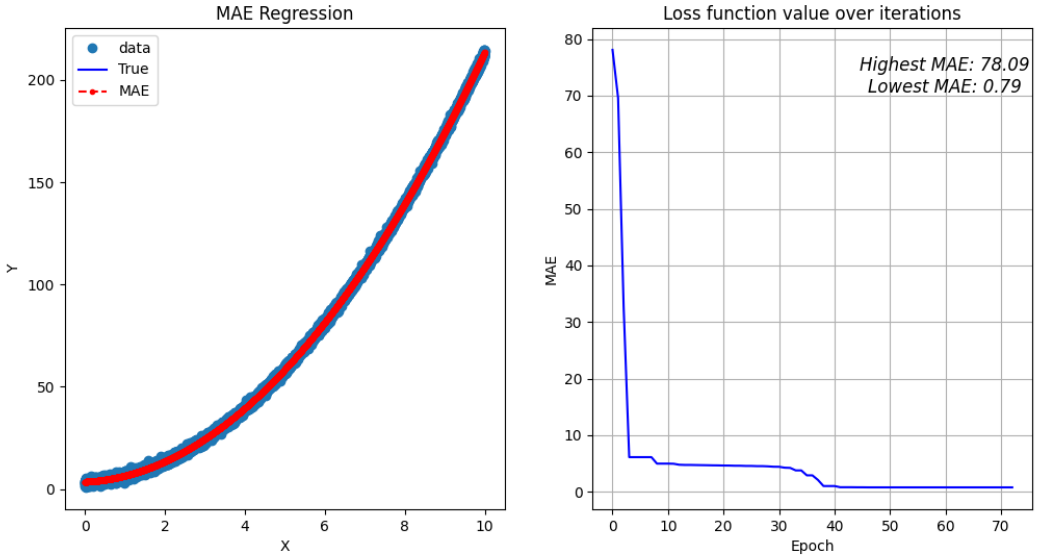
Частина 3. Experiment with number of samples, sigma, and optimization algorithms

Експеримент 1. Для даних з попереднього тесту змінюю семпл на 10000 спостережень

Parameters (MSE): [2.00009699 3.49875935] Iterations: 65 Lowest loss: 0.9900974924174376

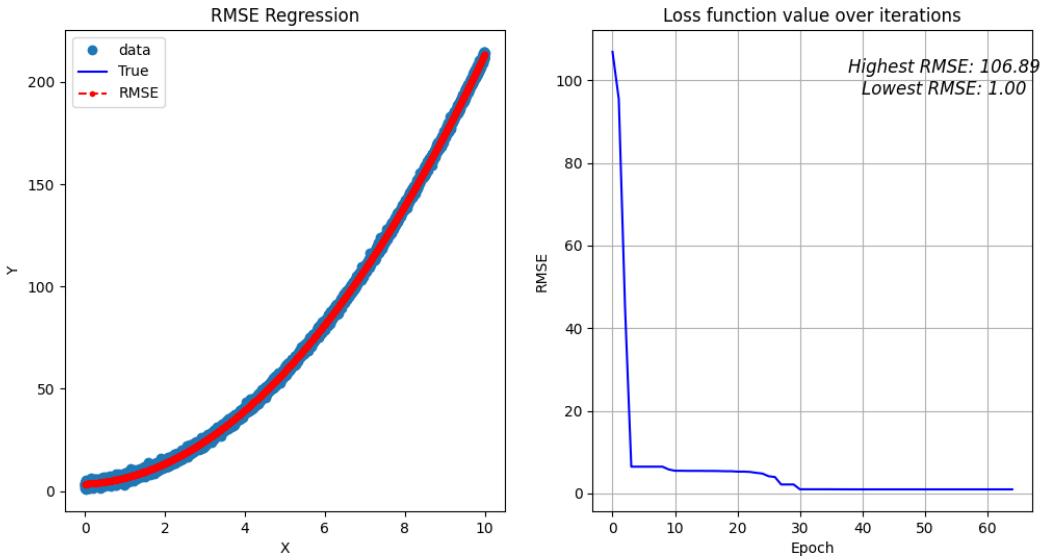


Parameters (MAE): [1.9995753 3.52783494] Iterations: 73 Lowest loss: 0.7937462111550543

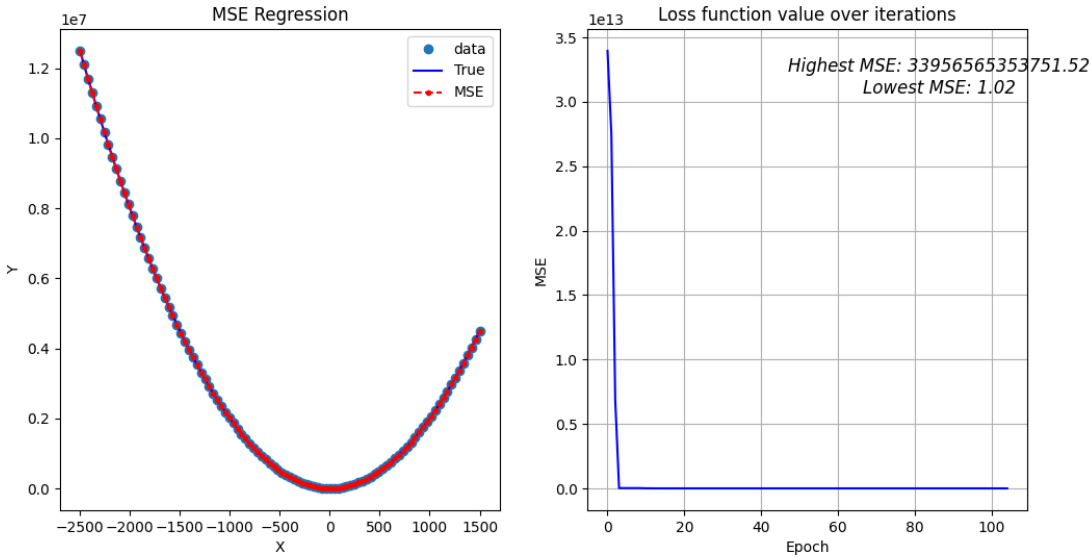


Parameters (RMSE): [2.00009699 3.49875935] Iterations: 65 Lowest loss: 0.995036427683649

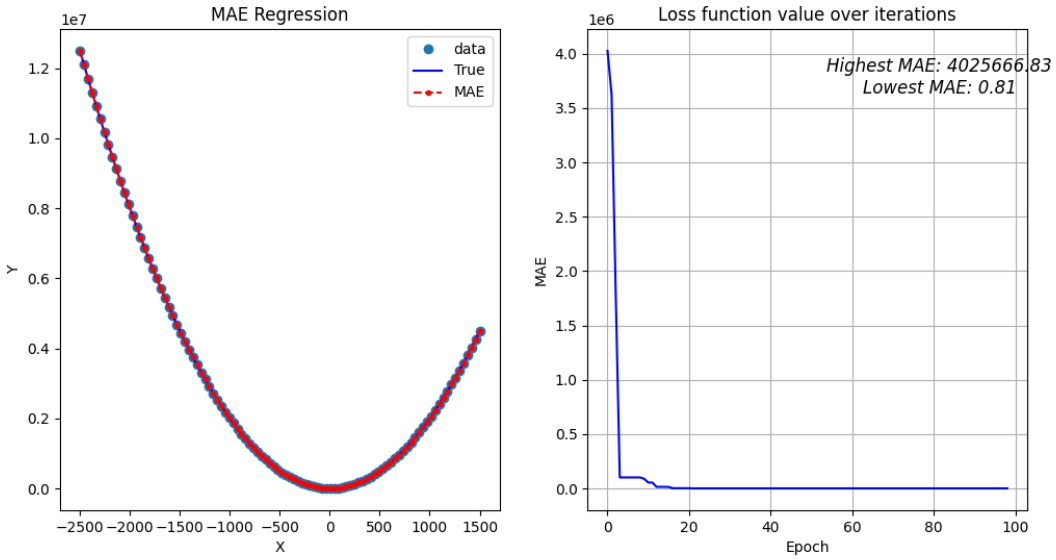
При збільшенні кількості спостереженб параметри знайдені більш точно.



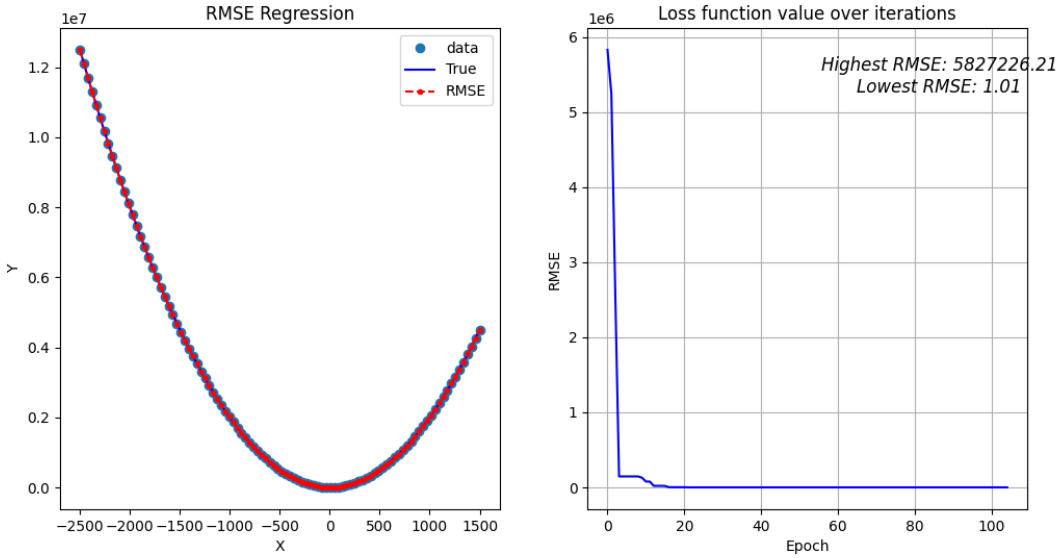
Експеримент 2. Для даних з частини 2 на 100 спостереженнях - тесту змінюю сігму в згенерованних данних на діапазон [-2500; 1500]



Parameters (MSE): [1.99999996 3.40638296] Iterations: 105 Lowest loss: 1.0200336026506593



Parameters (MAE): [1.9999999 3.60783813]   Iterations: 99 Lowest loss: 0.8122347025911651



Parameters (RMSE): [1.99999996 3.40638296] Iterations: 105 Lowest loss: 1.0099671294901926

Моделі довше сходились і параметри знайдені достатньо точно.

Експеримент 3. Inital Guess

Для данних з частини 2 на 100 спостереженнях і данних на діапазону [-2500; 1500]

initial_guess = np.array([-13,85])

Parameters (MSE): [1.99999996 3.40637924]   Iterations: 109 Lowest loss: 1.0200336031292039

Parameters (MAE): [1.9999999 3.60784934] Iterations: 103 Lowest loss: 0.8122346064380369

Parameters (RMSE): [1.99999996 3.40637924] Iterations: 109 Lowest loss: 1.0099671297271036

Експеримент 4. Optimization algorithms

Для початкових даних з $y = 2 * x + 3.5$ з loss = MSE тестую наступні алгоритми:

Nelder-Mead

- Зійшлась за 55 ітерацій
- Highest MSE: 89.29567546107769
- Lowest MSE: 1.0241037044214218
- Parameters: 2.00058069; 3.3425008

COBYLA

Parameters: [2.00162701 3.33480732]

Lowest MSE: 1.0241188968588417

dogleg

ValueError: Jacobian is required for dogleg minimization

ValueError: Hessian is required for dogleg minimization

...

Lets try to use a different optimization method that doesn't require the Jacobian, such as the BFGS method.

BFGS

Parameters: [2.00057341 3.34251044] Lowest MSE: 1.0241037032533469 Iterations: 7

Цей спосіб дуже швидко шійшовся показавши достатньо високу точність.