```
# напишіть функцію гіпотези лінійної регресії у векторному вигляді
import numpy as np

def hypothesis(X, theta):
    return np.dot(X, theta)


# створіть функцію для обчислення функції втрат у векторному вигляді
def compute_loss(X, y, theta):
    m = len(y)
    predictions = hypothesis(X, theta)
    loss = (1 / (2 * m)) * np.dot((predictions - y).T, (predictions - y))
    return loss[0]


# реалізуйте один крок градієнтного спуску
def gradient_descent_step(X, y, theta, alpha):
    m = len(y)
    predictions = hypothesis(X, theta)
    gradient = (1 / m) * np.dot(X.T, (predictions - y))
    theta_new = theta - alpha * gradient
    return theta_new


# знайдіть найкращі параметри w для датасету прогнозуючу ціну на будинок залежно від площі, кількості ванних кімнат та кількості спалень
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# URL датасету
url = 'https://drive.google.com/uc?export=download&id=1-rAa4XT4_fI0dOBlMNuE6a7jB0wln_Qo'

# Завантажуємо датасет
data = pd.read_csv(url)

# Вибираємо ознаки та цільову змінну
features = data[['area', 'bedrooms', 'bathrooms']]
target = data['price']

# Нормалізуємо дані
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Ділим дані на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(features_scaled, target, test_size=0.2, random_state=42)

# Додаємо стовпець з одиницями для зміщення (bias)
X_train = np.hstack((np.ones((X_train.shape[0], 1)), X_train))
X_test = np.hstack((np.ones((X_test.shape[0], 1)), X_test))

# Ініціалізуємо параметри моделі
theta_gd = np.zeros(X_train.shape[1])

# Параметри градієнтного спуску
learning_rate = 0.01
num_iterations = 50000

# Функція для одного кроку градієнтного спуску
def gradient_descent_step(X, y, theta, learning_rate):
    m = len(y)
    predictions = X.dot(theta)
    errors = predictions - y
    gradient = X.T.dot(errors) / m
    theta -= learning_rate * gradient
    return theta

# Використовуємо градієнтний спуск для знаходження найкращих параметрів
for i in range(num_iterations):
    theta_gd = gradient_descent_step(X_train, y_train, theta_gd, learning_rate)

# Виводимо знайдені параметри для градієнтного спуску
print("Найкращі параметри:", theta_gd)
```

```
Найкращі параметри (градієнтний спуск): [4736303.1615296   749023.05142652  265604.85843734  714016.44847747]
```

```
# знайдіть ці ж параметри за допомогою аналітичного рішення
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
# URL датасету
url = 'https://drive.google.com/uc?export=download&id=1-rAa4XT4_fI0dOBlMNuE6a7jB0wln_Qo'

# Завантажуємо датасет
data = pd.read_csv(url)

# Вибираємо ознаки та цільову змінну
features = data[['area', 'bedrooms', 'bathrooms']]
target = data['price']

# Нормалізуємо дані
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Ділим дані на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(features_scaled, target, test_size=0.2, random_state=42)

# Додаємо стовпець з одиницями для зміщення (bias)
X_train = np.hstack((np.ones((X_train.shape[0], 1)), X_train))

# Знаходимо параметри за допомогою аналітичного рішення
theta = np.linalg.inv(X_train.T @ X_train) @ X_train.T @ y_train

# Виводимо знайдені параметри
print("Найкращі параметри:", theta)
```

```
Найкращі параметри: [4736303.16152964  749023.05142651  265604.85843734  714016.44847748]
```

```
#порівняйте отримані результати
print("Найкращі параметри (градієнтний спуск):", theta_gd)
print("Найкращі параметри (аналітичний метод):", theta_analytical)
```

```
Найкращі параметри (градієнтний спуск): [4736303.1615296   749023.05142652  265604.85843734  714016.44847747]
Найкращі параметри (аналітичний метод): [4736303.16152964  749023.05142651  265604.85843734  714016.44847748]
```