# Linear and Polynomial Regression

Ada Boran Yılmaz

# 1. Introduction

In this project, we explore fundamental regression techniques, which are linear regression and polynomial regression, and apply them to two different datasets:

1. Dataset 1: A linear dataset generated by a known linear function plus Gaussian noise.
2. Dataset 2: A nonlinear dataset, where polynomial regression is used to capture its nonlinear behavior.

We examine three methods for linear regression on Dataset 1:
- scikit-learn's LinearRegression (Part 1.a)
- Manual Ordinary Least Squares (Part 1.b)
- Manual Gradient Descent (Part 1.c)

Then, for Dataset 2, we explore polynomial regression:
- scikit-learn's Polynomial Regression (Part 2.a) with polynomial degrees [1, 3, 5, 7]
- Manual Polynomial Regression for degree 3 (Part 2.b)

We measure performance using the Mean Squared Error (MSE) on the validation set. The goal is to compare how well different methods perform on linear vs. nonlinear data and understand the effect of the polynomial degree.

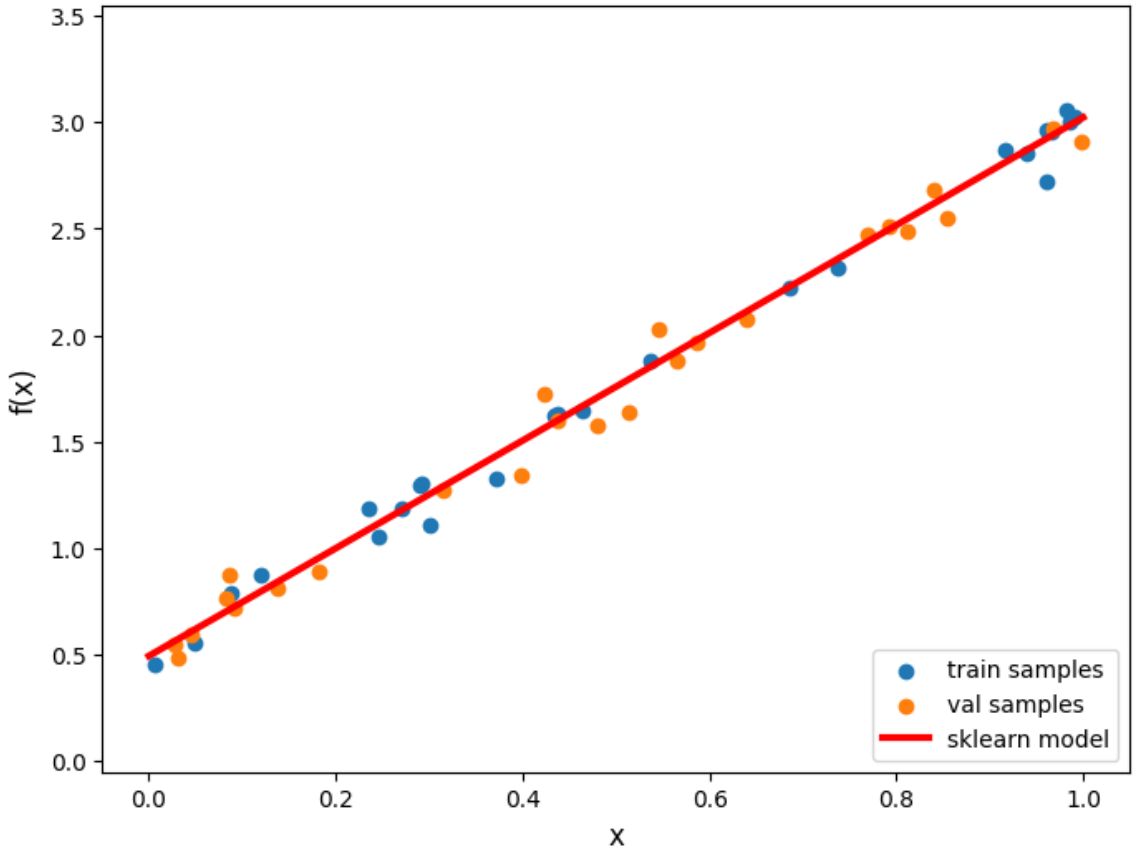
# 2. Part 1: Linear Regression on Dataset 1

## 2.1 Part 1.a: Using scikit-learn's Linear Regression

Method:
- Generated Dataset 1 (N=50) from a linear function $y = 2.5 x + 0.5$ plus Gaussian noise.
- Split into 50% train and 50% validation sets (25 samples each).
- Fitted a LinearRegression model and predicted on the validation set.

Results:
- Validation MSE: 0.00795462682779033
- Regression Coefficients: (Intercept and slope were close to 0.5 and 2.5, respectively.)

Figure 1:




## 2.2 Part 1.b: Manual Ordinary Least Squares (Pseudo-inverse)

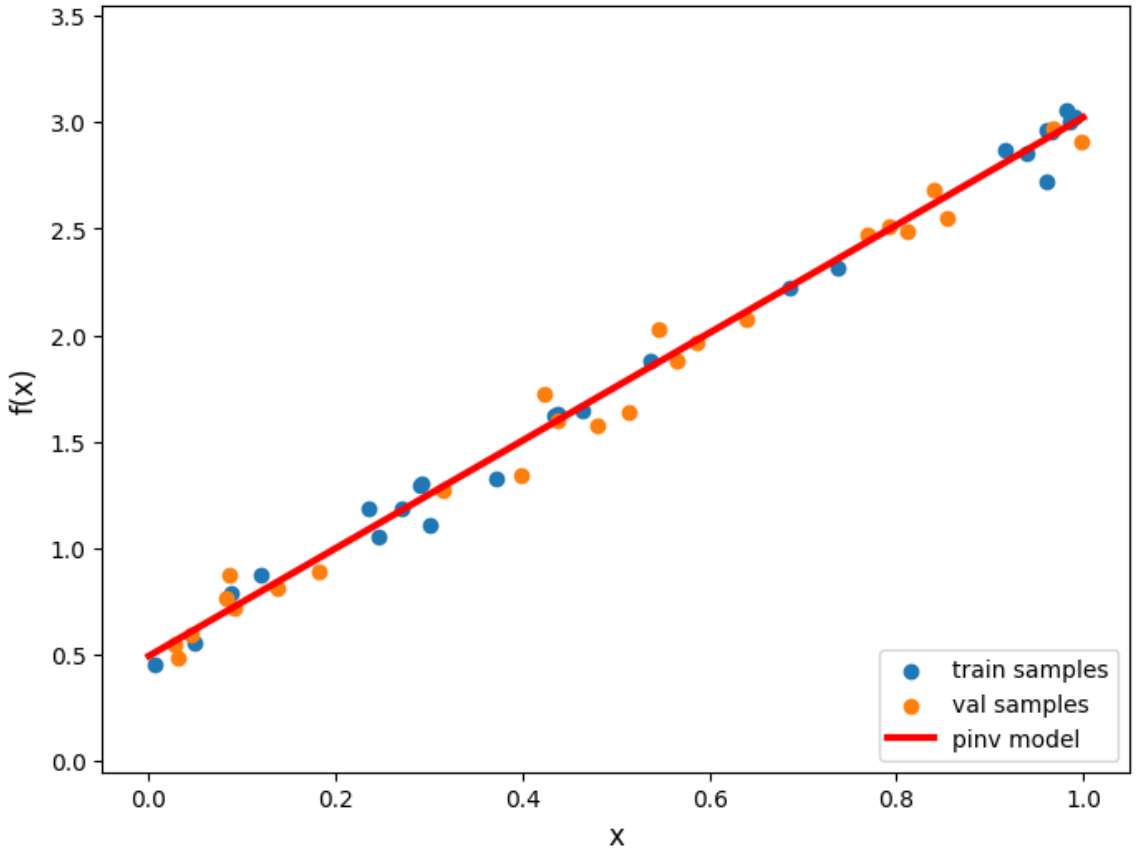
Method:

- Constructed an extended data matrix X by adding a column of ones for the intercept.
- Computed $w = (X^T X)^{-1} X^T y$ via pinv(X).
- Predicted on validation set and calculated MSE.

Results:

- Extended X_train shape: (25, 2)
- Extended X_val shape: (25, 2)
- Validation MSE: 0.007954626827790374

Figure 2:




## 2.3 Part 1.c: Manual Gradient Descent

Method:

- Initialized weights ($w_0$, $w_1$) to zeros.
- Used a learning rate (lr = 0.1) and iterated M = 1000 steps, updating weights based on the gradient of the MSE.
- Tracked the MSE over iterations.

MSE During Training:

- Step 1: 2.0573
- Step 100: 0.0313
- Step 200: 0.0075
- Step 300: 0.0035
- …
- Step 1000: 0.0026

Figure 3:

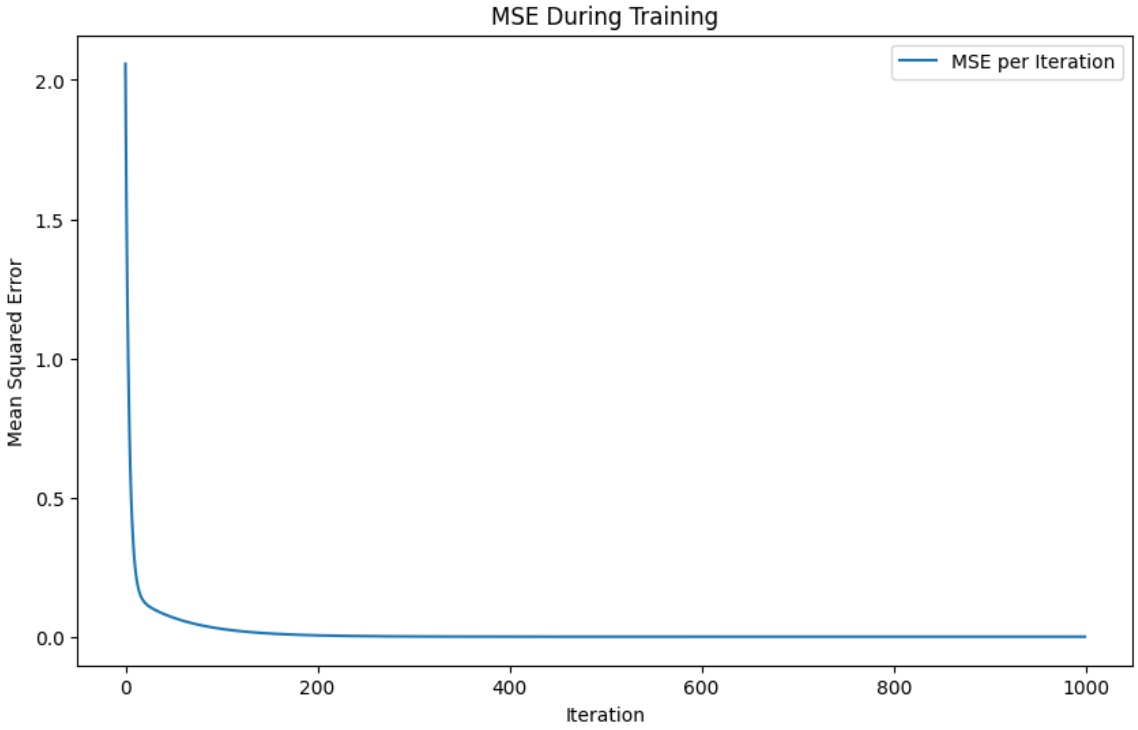
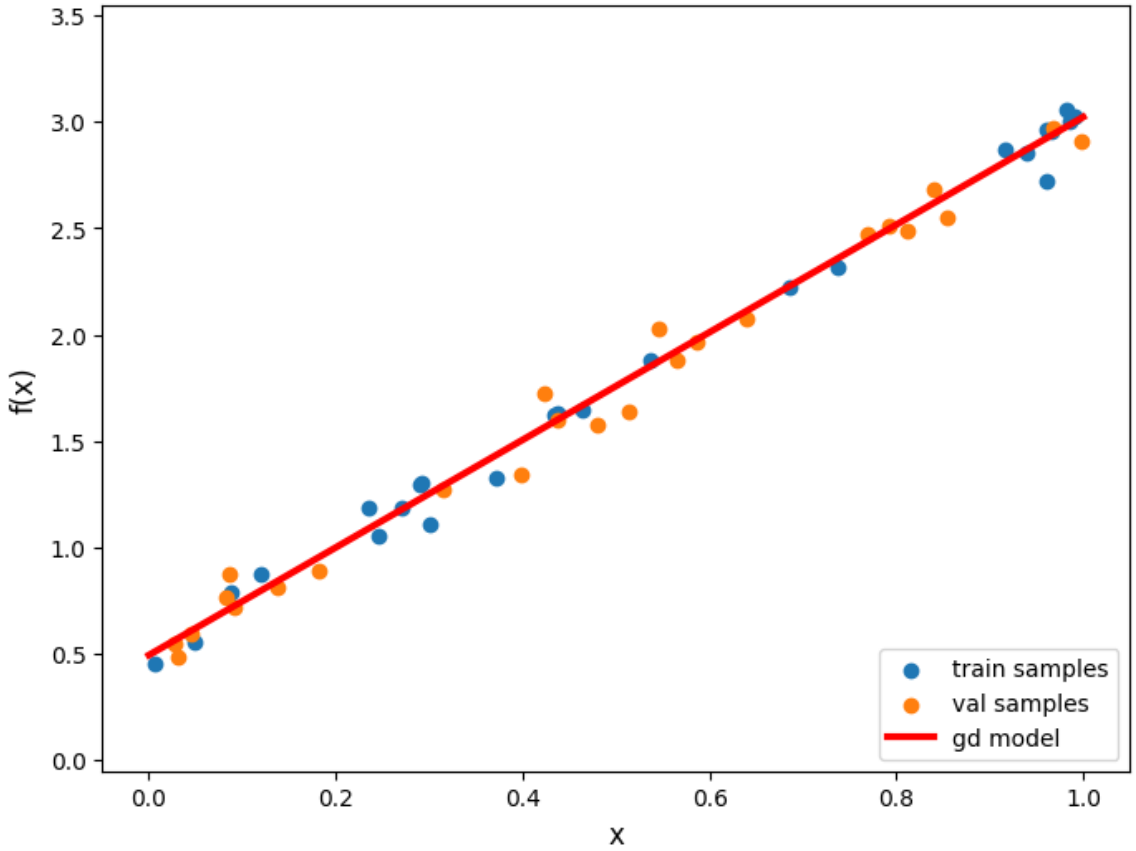



Figure 4:



Discussion: The final MSE (~0.0026) is very close to the results from Parts 1.a and 1.b, indicating that the gradient descent solution converged to a similar optimum. Minor differences are attributed to numerical precision.

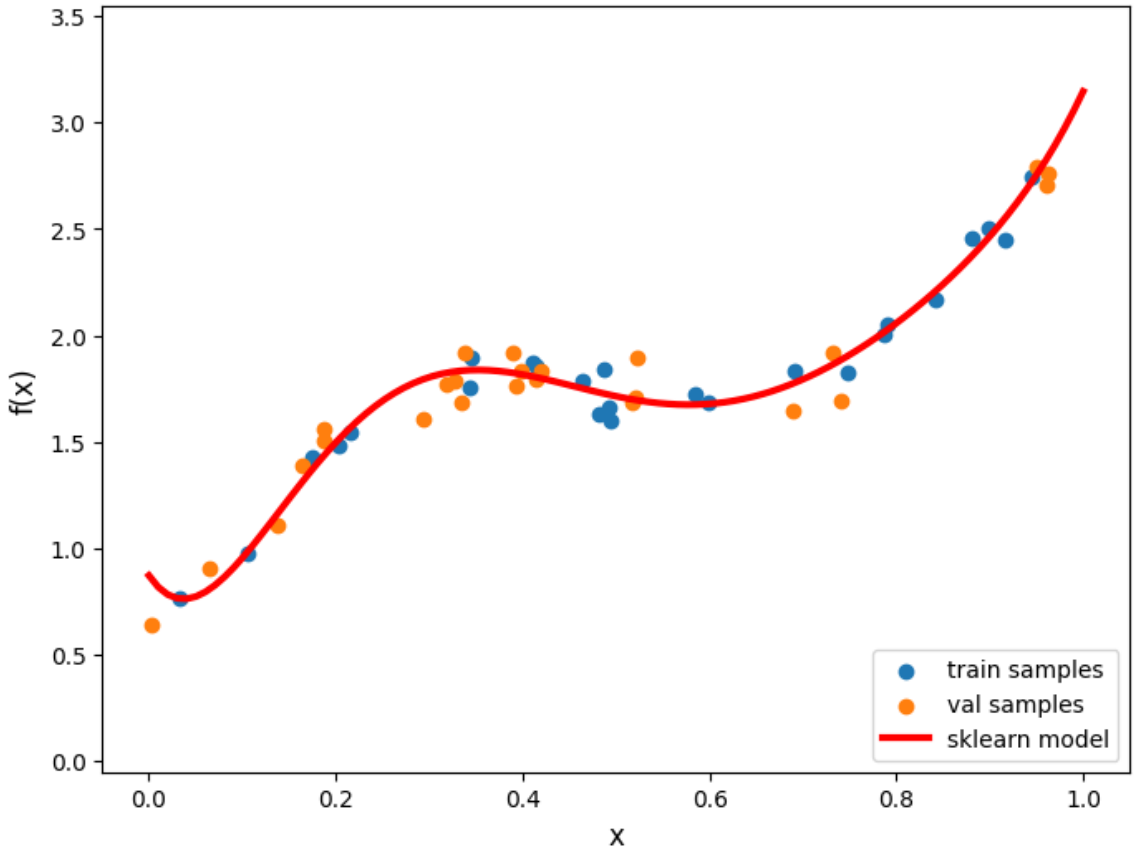
## 3. Part 2: Polynomial Regression on Dataset 2

### 3.1 Part 2.a: scikit-learn Polynomial Features + Linear Regression

Method:
- Loaded dataset2_data.npy and dataset2_labels.npy with shapes (50,1).
- Split data 50%-50% into training and validation sets.
- For polynomial degrees [1, 3, 5, 7], used PolynomialFeatures and LinearRegression to fit and predict.

Results (Validation MSE):
- Degree 1: 0.06362852709262727
- Degree 3: 0.012059223742868292
- Degree 5: 0.007475463079281102
- Degree 7: 0.011549227838827407

Figure 5:




Discussion: Increasing the degree improves the fit up to a point (degree=5) beyond which overfitting may occur.

## 3.2 Part 2.b: Manual Polynomial Regression (Degree 3)

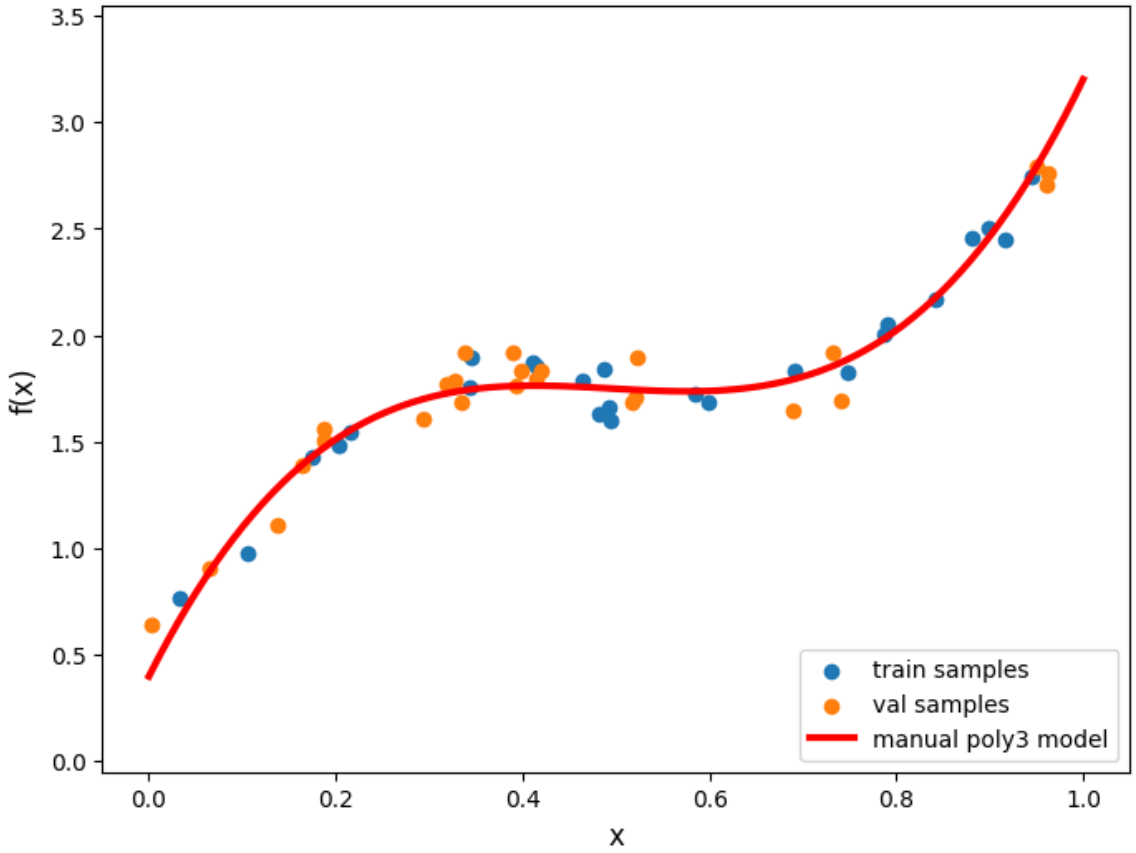
Method:

- Constructed extended data matrices for training and validation with columns [1, x, $x^2$, $x^3$].
- Computed regression coefficients using the pseudoinverse.
- Predicted on the validation set and calculated MSE.

Results:

- X_train_poly3 shape: (25, 4)
- X_val_poly3 shape: (25, 4)
- Validation MSE: 0.01205922374286855

Figure 6:




## 4. Discussion and Conclusion

1. Dataset 1 (Linear):

- All three methods (sklearn, manual pseudo-inverse, gradient descent) produced nearly identical fits and similar MSE values (~0.008).
- The gradient descent method converged to a solution comparable to the closed-form solutions.

2. Dataset 2 (Nonlinear):

- A linear model (degree=1) resulted in a high MSE (0.0636), indicating underfitting.
- Increasing the degree to 3 and 5 significantly reduced the MSE (0.0120 and 0.0075 respectively), with degree 5 providing the best balance.
- Degree 7 led to a slightly higher MSE (0.0115), suggesting overfitting.

3. Underfitting vs. Overfitting:

- A model that is too simple (low degree) underfits the data, while an excessively complex model (high degree) may overfit, capturing noise.

- In this case, a polynomial degree of 5 appears optimal for Dataset 2.

In conclusion, polynomial regression effectively captures nonlinear trends, and careful tuning of the degree parameter is essential to balance underfitting and overfitting. Meanwhile, for linear relationships, all standard methods yield similar results.