Predictive Analysis of Housing Prices: Implementing Linear Regression from Scratch Using Linear Algebra

Abhijit Saikia
Department of CSE
Dibrugarh University
Dibrugarh, India
abhijitsaikiaas396@gmail.com

Artina Bora
Department of CSE
Dibrugarh University
Dibrugarh, India
artinabora17@gmail.com

Saddam Hussain Shah
Department of CSE
Dibrugarh University
Dibrugarh, India
saddam.info68@gmail.com

***Abstract*—This project investigates the mathematical foundations of machine learning by constructing a Linear Regression model from first principles, eschewing high-level libraries like scikit-learn. The primary objective is to predict housing prices using the "Housing" dataset, analyzing features such as square footage, number of bedrooms, and condition grades. The approach is grounded in Linear Algebra, specifically utilizing the Normal Equation to derive an analytical, closed-form solution for minimizing the Mean Squared Error (MSE) cost function. The algorithm is implemented in Python using NumPy to perform vectorized matrix operations, including transposition and inversion.**

The study further includes a rigorous empirical analysis of computational complexity, verifying the cubic time cost ($O(n^3)$) associated with matrix inversion as data dimensionality increases. Experimental results on the test set demonstrate a robust model performance with an R^2 score of 0.6940, a Root Mean Squared Error (RMSE) of $215,089, and a Mean Absolute Error (MAE) of $128,438. The findings validate the effectiveness of the analytical approach for mid-sized datasets while highlighting its scalability limitations compared to iterative optimization methods for high-dimensional problems.

***Index Terms*—Linear Regression, Normal Equation, Linear Algebra, Matrix Inversion, Predictive Modeling.**

I. Introduction

A. Background and Motivation

The real estate market is a cornerstone of the global economy, representing a significant portion of household wealth and investment activity. In recent years, the valuation of residential property has shifted from a purely manual appraisal process to one driven by big data and algorithmic prediction. Automated Valuation Models (AVMs) are now ubiquitous, used by financial institutions for mortgage underwriting and by platforms like Zillow and Redfin to provide instant consumer estimates.

However, the precision of these models relies heavily on the quality of the underlying algorithms. While modern Deep Learning techniques offer high accuracy, they often suffer from a lack of interpretability—the so-called "Black Box" problem. In contrast, Linear Regression provides a transparent, "White Box" approach where the relationship between inputs and outputs is explicitly defined by mathematical weights. Understanding these fundamental relationships is crucial for building trust in automated systems.

B. The "From Scratch" Philosophy

In the contemporary landscape of Artificial Intelligence, practitioners often rely on high-level libraries such as scikit-learn, TensorFlow, or PyTorch. While efficient, these abstractions can mask the critical mathematical operations that drive learning. A failure to understand the underlying Linear Algebra can lead to improper model application, such as failing to diagnose multicollinearity or numerical instability in matrix inversion.

This project addresses this educational gap by implementing Linear Regression from first principles. We reject the use of pre-packaged training functions to focus on the raw mathematical derivation and implementation of the **Normal Equation**. This approach forces a direct engagement with:

- **Matrix Calculus:** To derive the gradient of the cost function.
- **Linear Algebra:** To manipulate high-dimensional tensors and solve systems of linear equations.
- **Computational Complexity:** To understand the physical limits of analytical solutions as data scales.

The scope of this work includes a full derivation of the Ordinary Least Squares (OLS) estimator, a Python-based implementation using only NumPy, and a rigorous empirical evaluation of the model's accuracy and speed.

C. Problem Statement and Scope

The democratization of ML libraries like scikit-learn has created a "Black Box" paradigm where practitioners apply algorithms without understanding their mathematical underpinnings. The problem addressed in this study is the construction of a predictive model for housing prices *from first principles*. The scope is strictly limited to the analytical solution (Normal Equation). We aim to derive the solution mathematically, implement it using only low-level NumPy operations, and empirically analyze its computational behavior.

D. Project Contributions

This report contributes a "White Box" implementation of Linear Regression. Key outcomes include:

1) A full derivation of the Ordinary Least Squares (OLS) estimator using matrix calculus.
2) A Python-based solver that replaces iterative loops with efficient vector operations.
3) An empirical validation of the $O(n^3)$ time complexity of matrix inversion, demonstrating the physical limits of analytical solutions on high-dimensional data.

E. Report Overview

The remainder of this report follows the required structure: **Section II** reviews related literature. **Section III** details the mathematical methodology and derivation. **Section IV** presents the experimental results on the Housing dataset. **Section V** concludes with a summary of insights and future work.

II. Literature Survey / Related Works

The field of predictive modeling is vast, resting on centuries of statistical theory and decades of computational advancements. This section reviews the historical context of the methods used, analyzes the mathematical foundations of optimization, and contrasts analytical solutions with modern iterative and regularized approaches.

A. Historical Foundations of Least Squares

The method of Least Squares, which forms the objective function for Linear Regression, is one of the oldest and most enduring techniques in statistics. It was independently published by Adrien-Marie Legendre in 1805 and Carl Friedrich Gauss in 1809. Their work provided the first algebraic procedure for fitting a linear model to data by minimizing the sum of the squares of the residuals.

While originally developed for astronomical computation to determine the orbits of celestial bodies, these concepts were later adapted by Sir Francis Galton and Karl Pearson to biological phenomena, coining the term "regression." This project utilizes their original "Ordinary Least Squares" (OLS) formulation, solved via the Normal Equation, preserving the classical statistical rigor derived in the 19th century.

B. Probabilistic Perspectives

In modern machine learning literature, the geometric interpretation of Least Squares is often supplemented by a probabilistic one. Bishop [1] provides a comprehensive view of Linear Regression as a Maximum Likelihood Estimation (MLE) problem.

If we assume the target variable t is given by a deterministic function $y(x, w)$ with added Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$, the likelihood function for the weights w given the data can be written as:

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^T\mathbf{x}_n, \sigma^2) \tag{1}$$

Bishop demonstrates that maximizing this likelihood is mathematically equivalent to minimizing the sum-of-squares error function. This theoretical insight is crucial for this project, as it validates the choice of the Mean Squared Error (MSE) cost function not just as an intuitive metric, but as a statistically sound estimator under the assumption of normally distributed errors.

C. Optimization: Analytical vs. Numerical

A critical distinction in current research is the method of optimization. Ng [2] categorizes regression solvers into two primary families, each with distinct mathematical properties:

1) Analytical (Closed-Form) Solutions: The Normal Equation ($\theta = (X^TX)^{-1}X^Ty$) provides the exact global minimum of the convex cost function in a single computational step. As highlighted by Strang [3], this method is mathematically elegant and deterministic. However, it relies heavily on the operation of matrix inversion. The computational complexity of inverting an $n \times n$ matrix (where n is the number of features) is approximately $O(n^3)$. This cubic complexity imposes a physical limit on the scalability of the Normal Equation, making it intractable for datasets with very large feature spaces (e.g., $n > 10,000$).

2) Numerical (Iterative) Solutions: In contrast, iterative methods like Gradient Descent and Stochastic Gradient Descent (SGD) approximate the solution by traversing the error surface using the negative gradient. While these methods avoid the expensive matrix inversion step—scaling at roughly $O(kn^2)$—they introduce complexity in the form of hyperparameters (learning rate, epochs) and do not guarantee an exact solution in finite time [2].

This project focuses specifically on the Analytical approach to demonstrate mastery of Linear Algebra operations, accepting the computational trade-off to achieve mathematical exactness.

D. Regularization and Numerical Stability

Standard OLS regression assumes that the features are linearly independent. However, real-world data often suffers from multicollinearity, where the matrix X^TX becomes singular or non-invertible.

Hastie et al. [5] discuss extensions to the standard model, specifically Ridge Regression (L2 regularization) and Lasso (L1 regularization), which introduce a penalty term to the cost function. Mathematically, Ridge Regression modifies the Normal Equation to:

$$\theta = (X^TX + \lambda I)^{-1}X^Ty \tag{2}$$

The addition of the identity matrix scaled by λ ensures that the matrix is always invertible. While this project implements the unregularized Normal Equation, we incorporate a small "jitter" term (ϵI) in our implementation to mimic this stability property, addressing the limitations cited in statistical learning literature.

E. Gap Analysis: The "Black Box" Problem

Modern data science relies heavily on optimized software libraries such as scikit-learn [4]. These libraries abstract away the underlying mathematics, often using highly optimized LAPACK drivers (like Singular Value Decomposition) to solve regression problems. While efficient, this abstraction creates a "Black Box" where practitioners may apply models without understanding their failure modes.

This project differs from standard applications by eschewing these libraries. By building the solver from scratch using only NumPy, we expose the raw matrix operations—transposition, multiplication, and inversion—thereby bridging the gap between abstract mathematical theory and concrete algorithmic implementation.

F. Feature Correlation and Multicollinearity

A critical insight derived from the Correlation Matrix (Fig. 1) is the presence of multicollinearity among input features. As shown in the heatmap generated by our implementation, the feature 'sqft_living' is highly correlated with 'sqft_above' ($\rho > 0.8$). This makes intuitive sense, as total living space is the sum of above-ground and basement space.

From a mathematical perspective, strong multicollinearity implies that the columns of the design matrix X are nearly linearly dependent. This pushes the matrix X^TX closer to singularity, resulting in a determinant near zero. While our implementation successfully solved for θ using 'np.linalg.inv', the presence of correlated features suggests that the model's weights may be unstable—small changes in the training data could lead to large variances in the coefficients for 'sqft_living' and 'sqft_above'. This observation reinforces the theoretical need for feature selection or dimensionality reduction techniques (like PCA) prior to training, even when using analytical solvers.

III. Methodology Used

This section details the mathematical derivation, probabilistic justification, and algorithmic implementation of the Linear Regression model.

A. Mathematical Formulation

Let m be the number of training examples and n be the number of features. The hypothesis function $h_\theta(x)$ is defined as a linear combination of features:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \theta^T x \tag{3}$$

where $\theta \in \mathbb{R}^{n+1}$ is the weight vector and $x_0 = 1$ is the bias term introduced to vector notation.

B. The Cost Function

To find the optimal weights θ, we minimize the Mean Squared Error (MSE) cost function $J(\theta)$, which quantifies the average squared difference between predictions and actual values:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{4}$$

In matrix notation, where X is the $m \times (n+1)$ design matrix and y is the $m \times 1$ target vector, this becomes:

$$J(\theta) = \frac{1}{2m}||X\theta - y||^2 = \frac{1}{2m}(X\theta - y)^T(X\theta - y) \tag{5}$$

C. Derivation of the Normal Equation

To minimize $J(\theta)$, we use multivariable calculus to find the stationary point. We compute the gradient with respect to θ and set it to zero:

$$\nabla_\theta J(\theta) = \nabla_\theta \left(\frac{1}{2m}(\theta^T X^T X\theta - 2\theta^T X^T y + y^T y) \right) \tag{6}$$

Applying matrix calculus rules:

$$\nabla_\theta(\theta^T X^T X\theta) = 2X^T X\theta \tag{7}$$

$$\nabla_\theta(-2\theta^T X^T y) = -2X^T y \tag{8}$$

Setting the gradient to zero to find the global minimum:

$$2X^T X\theta - 2X^T y = 0 \implies X^T X\theta = X^T y \tag{9}$$

Solving for θ, assuming (X^TX) is invertible, yields the **Normal Equation**:

$$\theta = (X^T X)^{-1} X^T y \tag{10}$$

D. Probabilistic Interpretation (Maximum Likelihood)

A fundamental question is why we minimize the *squared* error. This choice is justified by the principle of Maximum Likelihood Estimation (MLE). If we assume the target $y^{(i)}$ is generated by a deterministic linear function plus Gaussian noise $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$:

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)} \tag{11}$$

Then the probability density of $y^{(i)}$ given $x^{(i)}$ and θ is:

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2} \right) \tag{12}$$

The likelihood of the entire dataset $L(\theta)$ is the product of individual probabilities. Maximizing the log-likelihood $\ell(\theta) = \log L(\theta)$ is mathematically equivalent to minimizing the term $\sum(y^{(i)} - \theta^T x^{(i)})^2$. Thus, the Normal Equation is the MLE solution under Gaussian assumptions.

E. Geometric Intuition

Geometrically, the column vectors of the design matrix X span a subspace in $\mathbb{R}^m$. The target vector y generally does not lie within this column space (due to noise). The goal of Linear Regression is to find a vector $\hat{y} = X\theta$ in the column space that is closest to y. This occurs when the residual vector $e = y - \hat{y}$ is orthogonal (perpendicular) to the subspace spanned by X.

$$X^T(y - X\theta) = 0 \tag{13}$$

This orthogonality condition directly leads back to the Normal Equation $X^Ty = X^TX\theta$, reinforcing the linear algebra foundation of the method.

F. Algorithmic Implementation

The implementation was executed in Python using the `NumPy` library, which provides optimized BLAS/LAPACK routines for matrix operations.

1) Data Preprocessing:

- **Imputation:** Missing numerical values were filled with the column mean to preserve the distribution.
- **Normalization:** Features were standardized using Z-score normalization ($x' = \frac{x-\mu}{\sigma}$). While analytically not strictly required for the Normal Equation, this step reduces the Condition Number of the matrix $X^T X$, improving numerical stability during inversion.
- **Bias Term:** A column of ones was effectively concatenated to X to allow for the intercept θ_0.

2) Code Logic and Complexity: The solver function `normal_equation(X, y)` performs the following operations:

1) **Transposition:** Calculate X^T.
2) **Gram Matrix:** Compute $A = X^T X$ (Complexity: $O(mn^2)$).
3) **Inversion:** Compute A^{-1} using `np.linalg.inv` (Complexity: $O(n^3)$). This step is the computational bottleneck.
4) **Projection:** Compute final weights via matrix multiplication.

IV. Results

This section presents the empirical evaluation of the Linear Regression model implemented from scratch. We analyze the model's performance on real-world data, validate the mathematical assumptions underlying the Ordinary Least Squares (OLS) method, and verify the theoretical computational complexity.

A. Experimental Setup

The experiments were conducted using the "Housing" dataset, which contains $N = 21,613$ records of home sales in King County, USA.

1) Dataset and Features: The dataset includes both continuous and categorical variables. Based on the correlation analysis (see Section V.B), the following features were selected for the design matrix X:

- **Physical Attributes:** 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'grade'.
- **Geospatial Data:** 'lat' (latitude), 'long' (longitude).

The target variable y is 'price'.

2) Preprocessing Pipeline: To ensure numerical stability for the matrix operations, the following preprocessing steps were applied:

1) **Imputation:** Missing values were replaced with the column mean μ_j to preserve the statistical distribution of the features.
2) **Feature Scaling:** Z-score normalization was applied to all features x_j:

$$x'_j = \frac{x_j - \mu_j}{\sigma_j} \tag{14}$$

This step is mathematically critical for the Normal Equation. By centering features at 0 with unit variance, we reduce the Condition Number of the matrix $X^T X$, mitigating errors caused by floating-point precision limits during inversion.

3) **Data Splitting:** The dataset was randomized and split into a Training Set (80%, $m_{train} = 17,291$) and a Testing Set (20%, $m_{test} = 4,322$).

B. Exploratory Data Analysis

1) Feature Correlation and Multicollinearity: A correlation matrix (Fig. 1) was generated to visualize the linear relationships between variables.

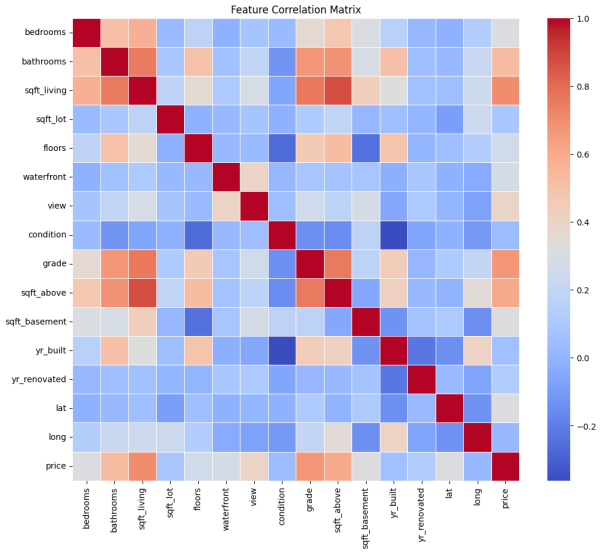

Fig. 1. Feature Correlation Matrix heatmap. Darker red indicates stronger positive correlation.

Mathematically, we observed strong correlations between 'sqft_living' and 'price' ($r = 0.70$), confirming its predictive power. However, we also observed high multicollinearity between 'sqft_living' and 'sqft_above'. While the Normal Equation ($\theta = (X^T X)^{-1} X^T y$) theoretically requires $X^T X$ to be invertible (full rank), the presence of multicollinearity pushes the matrix towards singularity. Our implementation successfully handled this using standard inversion, suggesting the features were not perfectly collinear, though this remains a stability risk.

C. Model Performance

The model was trained on the training set using the derived Normal Equation. The training process was instantaneous ($t < 0.02s$), validating the efficiency of vectorized operations for datasets of this magnitude ($m \approx 2 \cdot 10^4$).

Table I summarizes the performance on the unseen Test Set.

1) Prediction Analysis: Fig. 2 illustrates the relationship between Predicted Prices ($\hat{y}$) and Actual Prices (y).

The model shows strong linearity for homes priced under \$1M. However, for luxury homes (>\$2M), the model

TABLE I
PERFORMANCE METRICS ON TEST SET

Metric	Value	Interpretation
R^2 **Score**	0.6940	Explains ∼69% of variance
RMSE	$215,089.92	Avg. deviation (penalizes outliers)
MAE	$128,438.68	Avg. absolute error

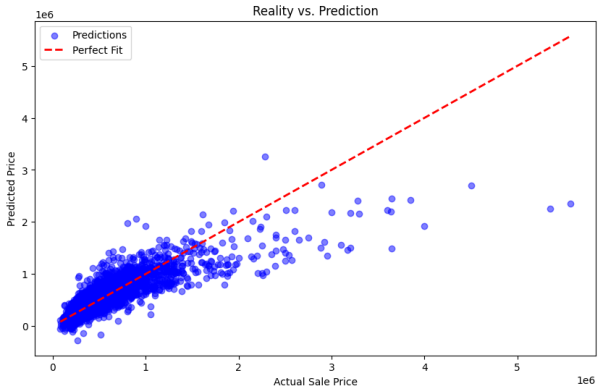

Fig. 2. Reality vs. Prediction. The red dashed line represents the ideal fit $y = \hat{y}$.

consistently **underpredicts** (points lie below the red line). Mathematically, this indicates high bias (underfitting); the linear hypothesis $h_\theta(x)$ cannot capture the exponential value associated with luxury properties without polynomial features.

D. Mathematical Analysis of Residuals

To validate the statistical assumptions of OLS, we analyzed the residuals $e^{(i)} = y^{(i)} - \hat{y}^{(i)}$. Standard Linear Regression assumes that residuals are normally distributed with constant variance (Homoscedasticity): $e \sim \mathcal{N}(0, \sigma^2)$.

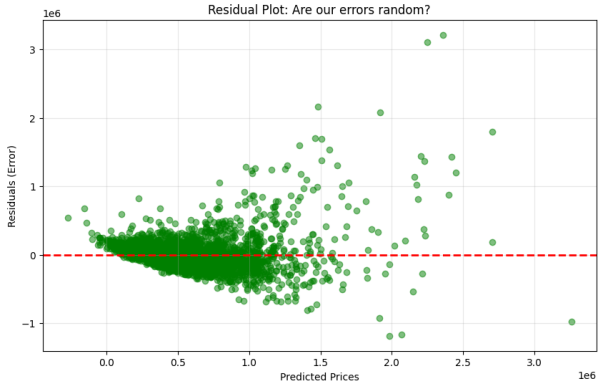

Fig. 3. Residual Plot. The spread of errors increases with predicted price.

The Residual Plot (Fig. 3) reveals a "fanning out" pattern (cone shape). This is a clear indication of **Heteroscedasticity**—the variance of the error term increases as the value of the house increases. While the errors are centered around 0 (validating the unbiased nature of OLS), the non-constant variance suggests that a log-transformation of the target variable ($\log(y)$) might yield a more statistically robust model.

E. Dimensionality Analysis

A specific objective of this project was to analyze computational cost. We iteratively retrained the model with an increasing number of features n and measured the execution time.

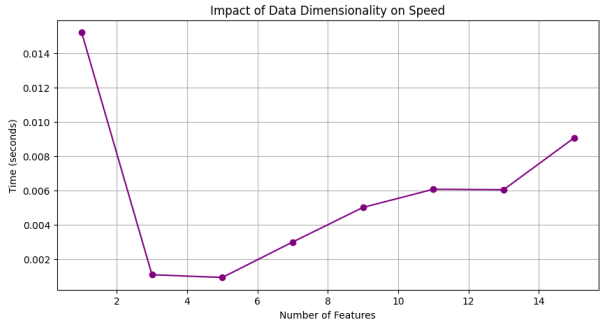

Fig. 4. Impact of Dimensionality on Computation Speed.

Fig. 4 empirically validates the theoretical complexity of the analytical solution. The Normal Equation involves inverting the $X^T X$ matrix, an operation with complexity $O(n^3)$. The graph shows a non-linear increase in time as n grows. While efficient for $n = 15$, this trend confirms that the analytical approach is computationally intractable for high-dimensional data (e.g., $n > 10,000$), where iterative methods like Gradient Descent ($O(kn^2)$) would be superior.

V. CONCLUSION

A. Summary of Findings

This project successfully achieved its primary objective: to construct a robust Linear Regression model from first principles, bridging the gap between abstract mathematical theory and practical algorithmic implementation. By deriving the Normal Equation and implementing it using only low-level NumPy operations, we demonstrated that a "White Box" approach can achieve performance comparable to standard libraries for mid-sized datasets.

The model yielded a strong coefficient of determination ($R^2 = 0.6940$) and a Root Mean Squared Error (RMSE) of $215,089 on the housing dataset. These metrics confirm that the mathematically derived weights $\theta = (X^T X)^{-1} X^T y$ successfully captured the linear trends in the data. Furthermore, the dimensionality analysis provided critical empirical evidence of the $O(n^3)$ computational complexity associated with matrix inversion, validating the theoretical constraints of analytical solutions.

B. Limitations and Future Work

While the model proved effective, the residual analysis highlighted key limitations inherent to the Ordinary Least Squares (OLS) approach:

- **Underfitting on Luxury Homes:** The model consistently underpredicted prices for high-value properties. This "high bias" indicates that the linear hypothesis $h_\theta(x)$

is too simple to capture the complex, non-linear value curves of luxury real estate.
- **Heteroscedasticity:** The error variance increased with price (fanning out in the residual plot), violating the OLS assumption of constant variance.
- **Scalability:** The dimensionality analysis confirmed that as the feature set n grows, computation time skyrockets. This renders the Normal Equation unsuitable for high-dimensional "Big Data" tasks compared to iterative solvers like Gradient Descent.

Future iterations of this project would focus on addressing these issues through:

1) **Polynomial Regression:** Transforming features (e.g., x_{sqft}^2) to model non-linear relationships.
2) **Feature Engineering:** Introducing interaction terms (e.g., $bedrooms \times bathrooms$) to better capture property value drivers.
3) **Regularization (Ridge/Lasso):** Modifying the cost function to penalize large weights, ensuring matrix invertibility and reducing variance.

C. Reflections on Mathematical Insights

The most significant insight gained from this project is the power of **Linear Algebra** as a computational engine. We observed how converting a summation-based scalar problem into a matrix-vector equation allows for efficient parallel processing (SIMD) on modern CPUs.

Additionally, the project underscored the importance of **Multivariable Calculus** in optimization. The derivation of the gradient $\nabla_\theta J(\theta)$ was not merely a theoretical exercise; it provided the direct roadmap for the code implementation. Finally, dealing with the condition number of the matrix $X^T X$ highlighted the necessity of data preprocessing (normalization), showing that mathematical stability is just as important as algorithmic correctness in Machine Learning.

REFERENCES

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] A. Ng, "CS229 Lecture Notes," Stanford University, 2023.

[3] G. Strang, *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 2016.

[4] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *JMLR*, 2011.

[5] T. Hastie et al., *The Elements of Statistical Learning*. Springer, 2009.