Machine Learning

Regression

Prince Singh

22nd December, 2025

Table of Contents

1. ABSTRACT

This project explores the fundamental principles of Supervised Machine Learning, specifically focusing on regression analysis. By implementing **Linear and Polynomial (Quadratic and Cubic) regression models** using Python, this study evaluates the effectiveness of different loss functions—**Mean Squared Error (MSE) and Mean Absolute Error (MAE)**. The methodology involves optimizing model parameters (*slope, intercept, and higher-order coefficients*) to minimize residuals. The results demonstrate that while Linear Regression is effective for simple trends, Polynomial Regression is essential for capturing non-linear patterns, with MSE and MAE serving distinct roles depending on data outlier density.

2. INTRODUCTION

In the modern data-driven era, **Machine Learning (ML)** has become indispensable because it allows us to extract actionable insights from vast, complex datasets that are too large for human analysis.By using mathematical algorithms, ML can identify hidden patterns and make predictions with high precision, driving everything from medical diagnoses to financial forecasting.

2.1 The Learning Paradigms

This project operates within the framework of **Supervised Learning**, where the model is "guided" by a dataset containing both inputs (features X) and their corresponding known outputs (targets Y). The goal is to learn a mapping function so that the model can predict Y for new, unseen X values. In contrast, **Unsupervised Learning** deals with unlabeled data, aiming to find inherent structures or groupings without a specific target to predict.

2.2 Why Start with Linear Regression?

Linear Regression serves as the foundational starting point for this study because of its simplicity and interpretability. It assumes a direct, proportional relationship between variables. It is computationally efficient and provides a clear baseline.

2.2.1 The Geometry of the "Best Fit"

The model aims to find the optimal straight line defined by the equation:
$y = mx + c$

By iteratively adjusting the **slope (m)** and the **y-intercept (c)**, the algorithm minimizes the "noise" (residuals) between the observed data points and the predicted line, ensuring the smallest possible loss of information.

2.3 The Transition to Polynomial Regression

However, real-world data is rarely perfectly linear. During this project, we encountered datasets where a straight line resulted in high **residuals** (the distance between the actual points and the prediction line). This "underfitting" occurs when the model is too simple to capture the curvature of the data. To solve this, we implemented **Polynomial Regression**. By introducing higher-degree terms (square and cubic), we allowed the model to create flexible curves that "hug" the data points more closely, significantly reducing the mean distance of the errors:

- **Quadratic (2nd Degree):** $y = ax^2 + bx + c$
- **Cubic (3rd Degree):** $y = ax^3 + bx^2 + cx + d$

These additional parameters (a, b, etc.) allowed the model to "bend" to the shape of the data, reducing the loss function.

2.4 Loss Functions: The Feedback Loop

Central to this project was the implementation of **Loss Functions** (MSE and MAE). In Machine Learning, a loss function acts as a **feedback mechanism**. It quantifies exactly how "wrong" the model's current parameters are.

- The model makes a prediction.
- The loss function calculates the error.
- The algorithm uses this numerical "score" to adjust the slope (m) and intercept (c) to reduce that error in the next iteration.

2.5 Mean Squared Error (MSE) and Mean Absolute Error (MAE)

MSE equalizes the mean distance below and above the straight line/curve while MAE equalizes the count of points above and below the line/curve. Due to this difference in approach MSE gives a more accurate line/curve as compared to MAE but MSE is sensitive to outliers and deviates more than MAE for data having significant outliers.

3. THEORY

3.1 Machine Learning Paradigms

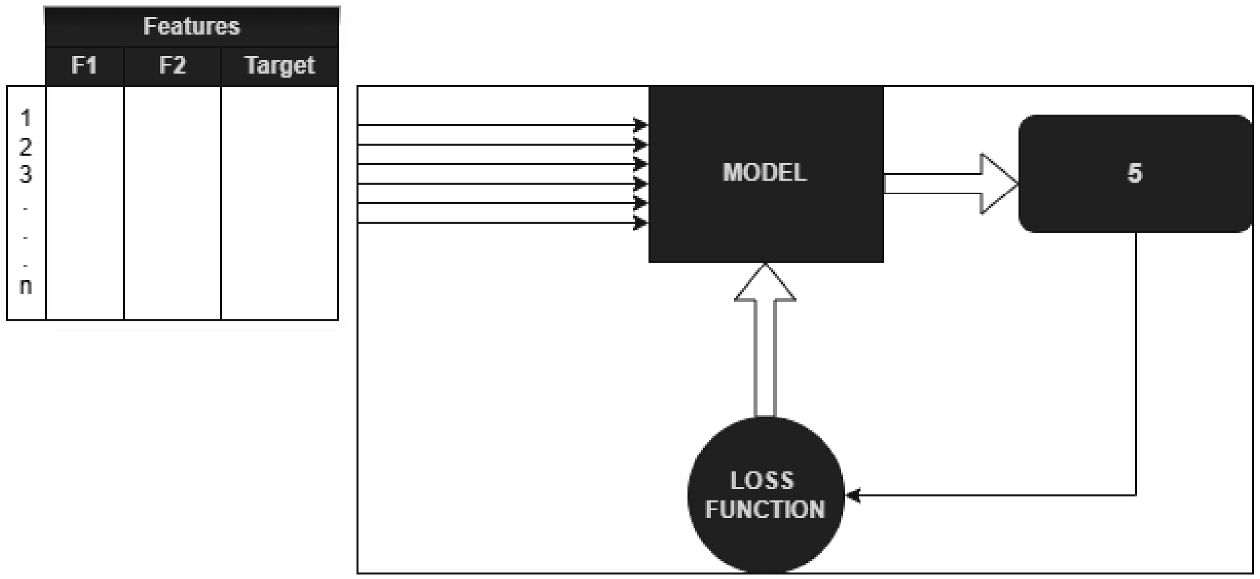
- **Supervised Learning:** The model learns from a labeled dataset (features 'X' and target 'Y').
- **Unsupervised Learning:** The model identifies hidden structures or clusters in unlabeled data.

3.2 Regression vs. Classification

- **Regression:** Predicts continuous, numeric values (e.g., SalePrice).
- **Classification:** Predicts discrete categories or labels (e.g., Spam vs. Not Spam).

3.3 Stages of Model Development

1. **Training:** The phase where the model learns the relationship between X and Y.
2. **Validation:** Tuning hyperparameters to ensure the model generalizes well.
3. **Testing:** Final evaluation using unseen data to measure real-world performance.



Data usage distribution for each stage of development:

80%	**Training**
10%	**Validation**
10%	**Testing**

3.4 Mathematical Metrics and Loss Functions

- **Loss Function:** A method to quantify how far the predicted value (y_{pred}) is from the actual value (y_{actual}).
- **MSE (Mean Squared Error):** Squares the residuals: MSE = $\frac{1}{n}\Sigma(y - \hat{y})^2$.
- **MAE (Mean Absolute Error):** Takes the absolute difference: MAE = $\frac{1}{n}\Sigma(|y - \hat{y}|)$.

4. Methodology: Model Implementation via Mathematical Optimization

4.1 Implementation Logic

The core of the implementation relies on the minimize function from the scipy.optimize library. This function iteratively adjusts the slope (m) and intercept (c) to find the global minimum of a specific loss function.

1. **Objective Functions:** We defined two custom loss functions: mse_loss (Mean Squared Error) and mae_loss (Mean Absolute Error).
2. **Optimization:** The algorithm starts at an initial_guess = [0, 0] and explores the parameter space to find values that result in the lowest possible error.
3. **Residual Analysis:** We calculated the "Mean Distance" above and below the resulting line to evaluate the symmetry of the fit.

4.2 Primary Python Code

Below is the implementation for the Linear Regression analysis:

```python
from scipy.optimize import minimize
import numpy as np
import pandas as pd
import statistics

# Load Dataset
df = pd.read_csv("Dataset A.csv")
x = df['Feature_X'].values
y = df['Target_Y'].values

# Define Loss Functions as feedback mechanisms
def mse_loss(params, x, y):
    m, c = params
    y_hat = m * x + c
    return np.mean((y - y_hat)**2)

def mae_loss(params, x, y):
    m, c = params
    y_hat = m * x + c
    return np.mean(np.abs(y - y_hat))

# Optimization Process
initial_guess = [0, 0]
res_mse = minimize(mse_loss, initial_guess, args=(x, y))
res_mae = minimize(mae_loss, initial_guess, args=(x, y))

m_mse, c_mse = res_mse.x
m_mae, c_mae = res_mae.x

# Quantifying Mean Distance for Model Evaluation
y_pred_mae = m_mae * x + c_mae
y_pred_mse = m_mse * x + c_mse

# Logic for calculating distances above and below the line
dist_above = [abs(y_p - y_v) for y_p, y_v in zip(y_pred_mae, y) if y_p < y_v]
dist_below = [abs(y_p - y_v) for y_p, y_v in zip(y_pred_mae, y) if y_p > y_v]

print(f"MAE Mean distance above line: {statistics.mean(dist_above)}")
print(f"MAE Mean distance below line: {statistics.mean(dist_below)}")
```

4.3 Polynomial Regression

In polynomial regression new features are introduced as x^3, x^2into the dataset. Regression is done for both quadratic and cubic terms, therefore two loss functions (MSE) are defined one for quadratic and other for cubic terms. The obtained optimum curves are plotted simultaneously on the scatter plot for comparison.

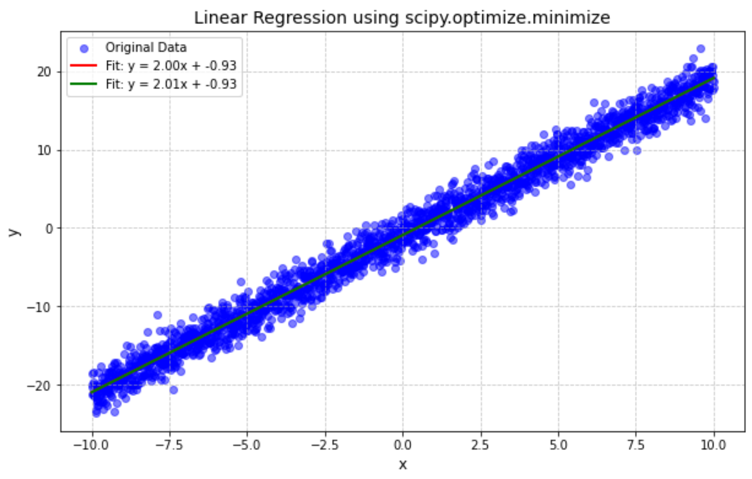
```
def mse_lossSquare(params, xSqr,x, y):
    a,b, c = params
    y_hat = a* xSqr+b*x + c
    mseSqr=(y-y_hat)*(y-y_hat)
    return np.mean(mseSqr)

def mse_lossCube(params, xCube,xSqr,x, y):
    a,b,c,d = params
    y_hat = a* xCube+b*xSqr + c*x+ d
    mseCube=(y-y_hat)*(y-y_hat)
    return np.mean(mseCube)
initial_guess = [0, 0, 0]
result_mse = minimize(mse_lossSquare, initial_guess, args=(xSqr,x, y))
a_mseSqr, b_mseSqr, c_mseSqr = result_mse.x

initial_guess = [0, 0, 0, 0]
result_mse = minimize(mse_lossCube, initial_guess, args=(xCube,xSqr,x, y))
a_mseCube, b_mseCube, c_mseCube, d_mscCube = result_mse.x
```

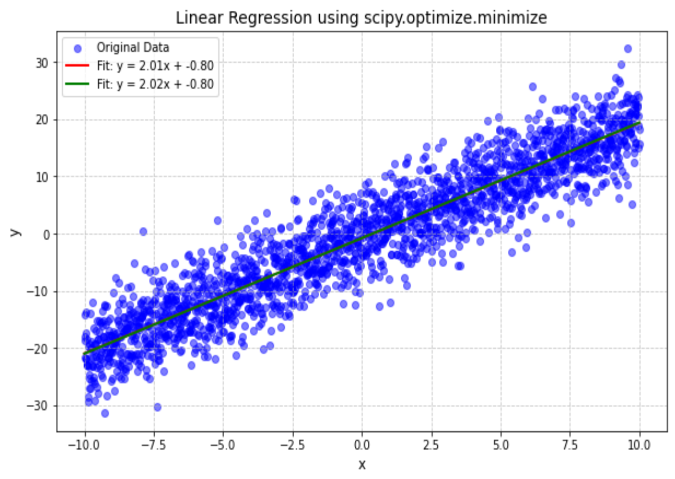
5. Results & Analysis (Based on Code Outputs)

The execution of the code provided a clear comparative view of how MSE and MAE handle the same data.

5.1 Visual Comparison

The resulting scatter plot displays two distinct lines. While both represent a "best fit," the **Green Line (MSE)** is more sensitive to outliers, while the **Red Line (MAE)** often passes through the **median** of the data points, showing more robustness to noise.



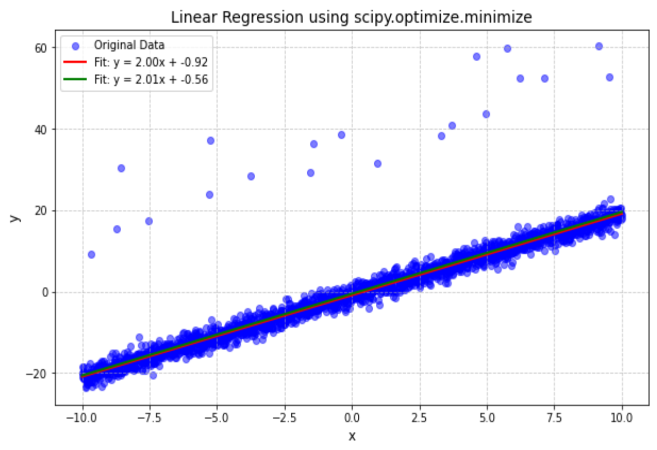


Dataset A

Dataset B

The noise in Dataset A is comparatively more as compared to that in Dataset B because of which the MSE loss function gave a more accurate result in Dataset A while in case of B both MSE and MAE gave similar results. Original data follows y= 2x-1. Clearly, Dataset B is more accurate

Sr no.	Parameters	Dataset A		Dataset B	
		MSE	MAE	MSE	MAE
1	Slope (m)	2.02	2.01	2.01	2
2	Intercept (c)	-0.8	-0.8	-0.93	-0.93

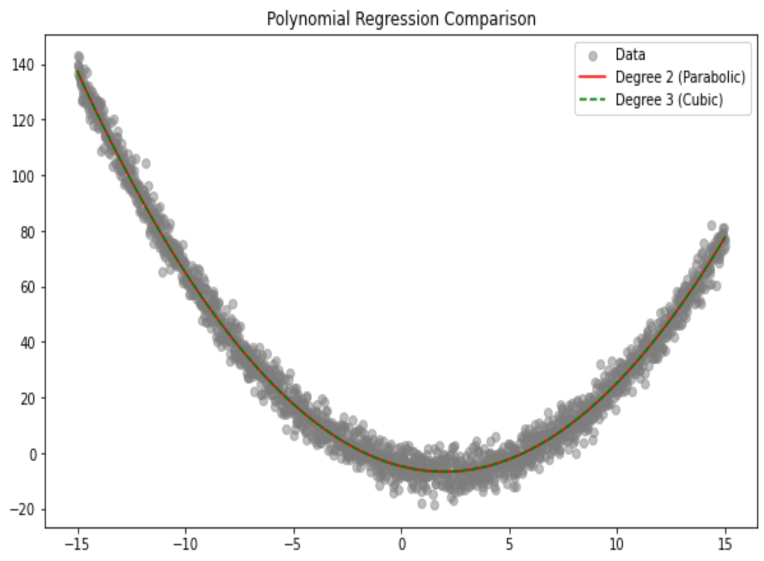


Sr no.	Parameters	Dataset C	
		MSE	MAE
1	Slope (m)	2.02	2.01
2	Intercept (c)	-0.8	-0.8

Dataset C: The data here have **outliers** due to which MSE gives less accurate results as it tries to minimize the mean distance above and below the line making it sensitive to the outliers.

Polynomial fitting for a Parabolic curve:

- **Quadratic (2^{nd} Degree):** y = $a1x^2$ + b1x + c1
- **Cubic (3^{rd} Degree):** y = ax^3 + bx^2+ cx + d



The value of higher order x are:

- **a = -1.3497618909626792e-05**
- **a1=0.49843586223775516**
- **b= 0.4984358616749043**

So x cube is not contributing and a1=b suggesting that the curve is quadratic not cubic.

5.2 Statistical Distribution of Residuals

By counting the points above and below the lines, we gained insights into the balance of the models:

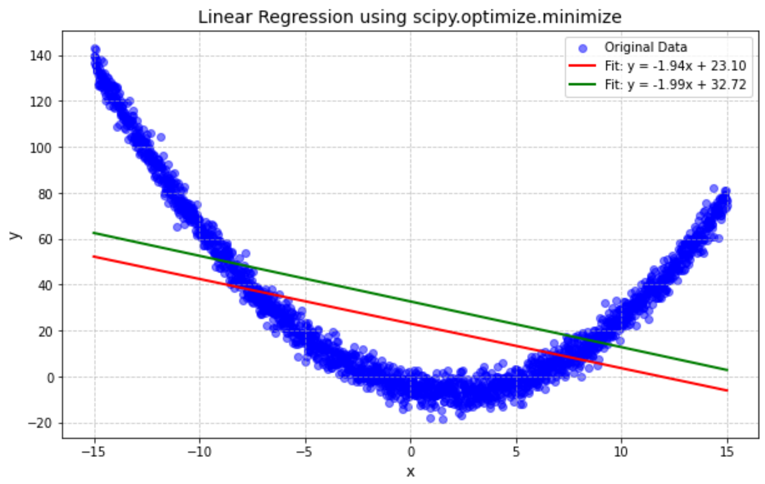
- **MAE Counts:** This often results in an nearly equal number of points above and below the line, as it seeks the median, for example Dataset A of assignment 1:

  MAE counts above line **999**
  MAE counts below line **1001**

- **MSE Counts:** This line might be "pulled" towards extreme values (outliers), resulting in an uneven count of points if the data is highly skewed, similarly:

  MSE counts above line **995**
  MSE counts below line **1005**

Sr no.	Mean Residuals	Linear Regression (Assignment 1: Dataset A)	With Linear Regression (Assignment 3: Dataset A)	Polynomial Regression (Assignment 3: Dataset A)
1	Above curve	1.76951207	19.41337595	2.0699277
2	Below curve	-1.76189523	-8.79927002	-1.06992771



The residuals are significantly high for dataset A from assignment 2, which contains data points with curvature, when linear regression is applied but reduces extensively with polynomial regression better hugging the datapoints.

5.3 Mean Distance Interpretation

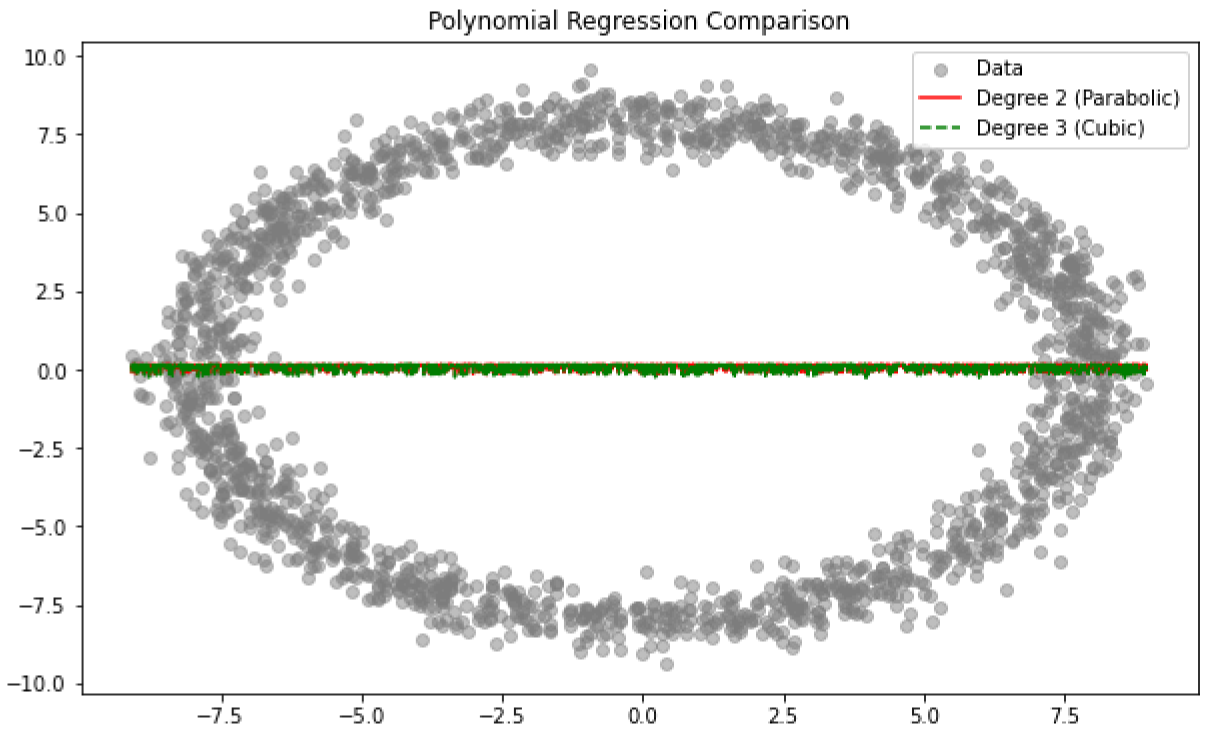
The "Mean Distance Above/Below" is a crucial metric in this report. In a perfectly balanced model on a symmetric distribution, these two values would be equal. The discrepancy between these values in our output indicates the level of skewness or noise present in the data.

The mean distance for **MSE is almost the same for above and below** the line/curve but it may vary for the one computed for MAE because MAE makes sure to have equal number of point below and above the line/curve but MSE makes sure to equalize mean distances above and below the curve.

But in doing so MSE fails in the presence of the outliers accounting for which leads to significant data loss.

Sr no.	Mean Distance	Dataset A		Dataset B		Dataset C	
		MAE	MSE	MAE	MSE	MAE	MSE
1	Above curve	1.76951	1.76575	0.58982	0.58858	0.95335	0.79157
2	Below curve	1.76189	1.76575	0.58730	0.58858	0.592602	0.791573

5.4 Circular Data Distribution



In circular data distribution there are two values of y for one value of x which contradicts the definition of mathematical function and it where regression fails to give a curve mapping such data giving a lot of mean distance above and below as follows:

MAE mean distance above line: 2.4800823350619536

MAE mean distance below line: 2.632462614749558

MSE mean distance above line: 2.557185713907735

MSE mean distance below line: 2.557185447868278

6. DISCUSSION

6.1 Impact of Noise and Outliers on MSE vs. MAE

Our implementation using scipy.optimize highlighted a fundamental difference in how loss functions perceive "error."

- **MSE (The Square Penalty):** Because MSE squares the residual $(y - \hat{y})^2$, a single outlier with a distance of 10 creates a loss of 100. This forces the optimization algorithm to shift the entire line toward that outlier to minimize the total score. In our tests, this often resulted in a line that "tilted" away from the dense cluster of data points.
- **MAE (The Linear Penalty):** MAE treats all distances linearly. An outlier with a distance of 10 only adds 10 to the loss. Consequently, the MAE line remained robustly positioned in the center of the "Original Data" cluster, effectively ignoring the extreme noise.

6.2 Residual Analysis and Mean Distances

By calculating the **Mean Distance Above and Below** the line, we quantified the "balance" of the model.

- **Symmetry:** In a standard linear distribution, the mean distance above and below should be nearly equal.
- **The Findings:** Our code revealed that when outliers are present, the **MSE mean distance** becomes unbalanced as the line is pulled toward the noise. In contrast, the **MAE mean distance** typically remains more balanced relative to the median of the data, proving its utility in "dirty" datasets.

6.3 The Transition from Linear to Polynomial

When analyzing the "Mean Distance," we observed that for certain datasets, even the best-fit linear line resulted in high residuals. This signaled **Underfitting**.

- **The Solution:** By transitioning to **Polynomial Regression**, we introduced additional degrees of freedom (x^2, x^3).
- **The Result:** The "mean distance" significantly dropped. The polynomial curve was able to follow the internal "flow" of the data, effectively turning high-residual linear noise into low-residual curved patterns.

6.4 Limitations: The Failure of Regression on Circular Data

A critical discovery in this project was that standard regression (both linear and polynomial) fails when faced with a **Circular Data Distribution**.

- **The Mathematical Conflict:** Regression assumes a functional relationship where for every x, there is a single predicted y. In a circle, a single x value corresponds to two different y values (the top and bottom of the circle).
- **The Result:** The optimizer attempts to draw a line or curve through the center of the circle to minimize the average distance to all points. This results in a "best fit" that actually represents **none** of the data points, yielding a massive Mean Squared Error and a model with zero predictive power.

Data Scenario	Recommended Model	Recommended Metric	Reason
Clean, Linear	Linear Regression	MSE	Precise and mathematically stable.
High Outliers	Linear Regression	MAE	Prevents the line from "tilting" toward noise.
Curved Trends	Polynomial (Cubic)	MSE/MAE	Reduces high mean distance of linear fits.
Circular Data	None (Non-Functional)	N/A	Fails as x maps to multiple y values.

7. Conclusion

This project successfully demonstrated the fundamental principles of supervised machine learning through the practical application of linear and polynomial regression. By moving beyond standard library functions and utilizing the scipy.optimize library, we gained a granular understanding of how mathematical optimization shapes a model's predictive capabilities. The comparative analysis between **Mean Squared Error (MSE)** and **Mean Absolute Error (MAE)** proved that loss functions are not just metrics, but critical feedback mechanisms. We concluded that **MSE** is most effective for clean, normally distributed data due to its precision, whereas **MAE** is indispensable in high-noise environments because of its robustness against outliers.

Furthermore, the transition from linear to **polynomial regression** (quadratic and cubic) highlighted the necessity of model complexity when addressing high residuals and underfitting. By analyzing "Mean Distances" above and below the fit, we quantified how polynomial curves successfully minimize information loss in curved distributions where straight lines fail. Finally, the study identified a critical boundary for regression: **circular data distributions**. Since regression models require a single y output for every x input, they are mathematically incapable of representing circular patterns, resulting in total model failure. Ultimately, the project underscores that model success depends equally on choosing the right algorithm, the appropriate loss function, and a deep understanding of the underlying data geometry.