

TEchnical report

Implementation analysis



Mirco fernando - 0103

2025/08/22

Zuu Crew Academy

Foundations of Machine Learning

Contents

[Executive summary 2](#_Toc206759746)

[Objective 2](#_Toc206759747)

[Approach 2](#_Toc206759748)

[Key Findings 2](#_Toc206759749)

[Implementation 3](#_Toc206759750)

[Linear Regression 3](#_Toc206759751)

[Linear regression with Multiple features 5](#_Toc206759752)

[Polynomial regression 6](#_Toc206759753)

[Logistic Regression 7](#_Toc206759754)

[Results 8](#_Toc206759755)

[Challenges Faced 12](#_Toc206759756)

[Conclusion & Future Work 12](#_Toc206759757)

# Executive summary

## Objective

1. Implementation of all algorithms from scratch.

* Linear regression with single feature
* Polynomial regression with single feature
* Linear regression with multiple features
* Polynomial regression with multiple features
* Logistic regression single feature

1. Compare with Scikit-Learn’s implementation for benchmarking
2. Evaluate models using performance metrics
3. Perform comprehensive evaluation metrics for classification

## Approach

Part 1: Linear regression for a single feature, predict Capstone Score from Total Hours

Part 2: Linear regression for multiple features, predict Capstone Score from (Education Level, Attendance, Total Hours, Assignments Completed, Hackathon Participation, GitHub Score, Peer Review Score)

Part 3: Polynomial regression for a single feature, predict Capstone Score from Total Hours

Part 4: Polynomial regression for multiple features, predict Capstone Score from (Education Level, Attendance, Total Hours, Assignments Completed, Hackathon Participation, GitHub Score, Peer Review Score)

Part 5: Scikit learn Implementation for linear and polynomial regression comparing parameters and performance benchmarking

Part 6: Implementation of Logistic regression, predicting Binary classification of pass/fail (capstone\_score ≥ 75)

Part 7: Comprehensive Evaluation metrics for classification, Implementing all classification metrics : - Confusion Matrix, Accuracy, Precision, Recall, F-Score - ROC Curve and AUC calculation - Precision-Recall Curve - Threshold tuning analysis.

## Key Findings

Linear regression fits well when relationship is approximately linear, polynomial regression improves performance for nonlinear and large-scale data, Feature scaling is a must when dealing with multiple features.

Logistic regression achieved 70% accuracy but low precision/recall due to class imbalance.

Scikit-learn implementation is faster, more stable, and gives identical results when parameters match.

# Implementation

## Linear Regression

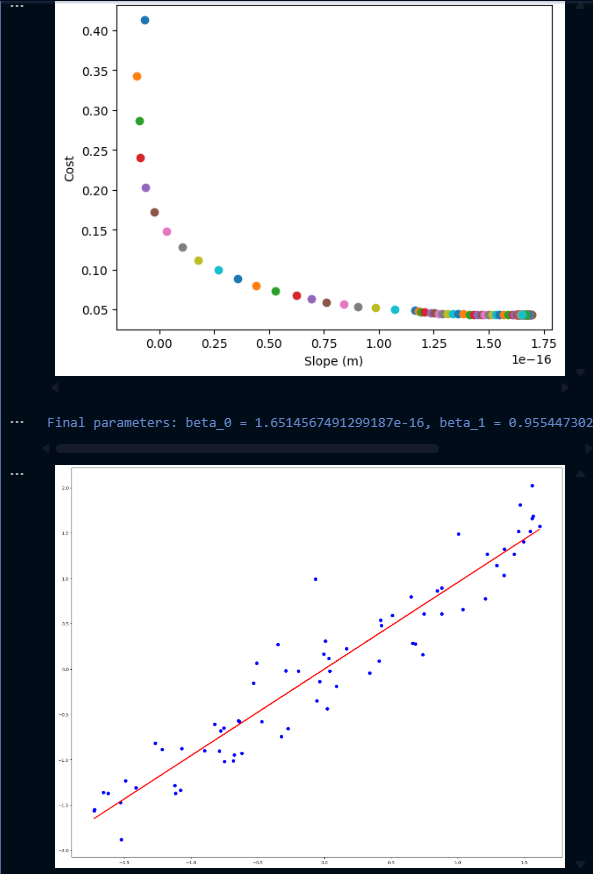
The goal is to predict the Capstone score which a continuous value with a single feature Total hour, this involves teaching a model using input data (Total hours) and corresponding output (Capstone Score)

Implemented from scratch using:

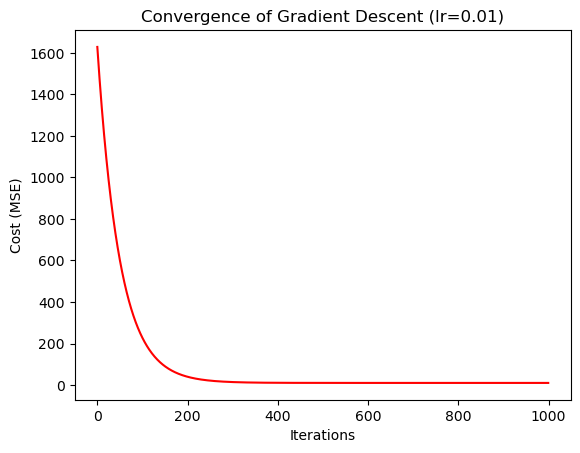
* Model Hypothesis
* Cost function: Mean Squared Error (MSE)
* Gradient descent optimization

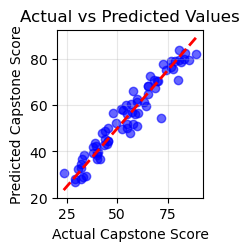
The gradient descent is optimized by learning rate = 0.1, iterations = 1000

The slope changes when the learning rate is modified



## Linear regression with Multiple features

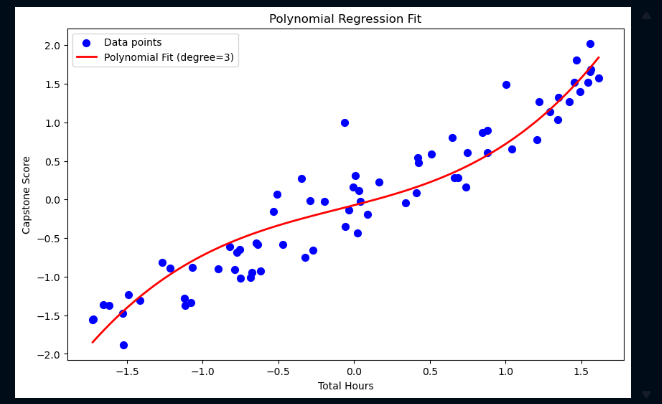


 **Same as Linear regression with single feature the model hypothesis is modified to implement multiple features and feature scaling is done for input values.**

Feature scaling is implemented and X bias was made for beta values

## Polynomial regression

Same as Linear regression, Instead of fitting a straight line(slope), we fit a polynomial curve by adding powers of X, extending to **polynomial regression** using feature expansion (degree = 2, 3).



A graph with blue dots and a red line

AI-generated content may be incorrect.The polynomial curve changes when the degree is changed so the optimal degree would be 3 when comparison

## Logistic Regression

This algorithm is a classification models this predicts if something is True or False, instead of fitting a straight line into the data, Logistic regression fits an S-Shaped curve to the data using the sigmoid function

A graph of a person with a number of points

AI-generated content may be incorrect.

A graph with a blue line

AI-generated content may be incorrect.

**Design Decisions**

* Normalization applied to features for faster convergence.
* Bias column added manually to handle intercept term.
* Epsilon smoothing added in cost function to prevent log(0) errors.
* Manual implementation of metrics to understand the math behind them.

# Results

**Linear Regression**

When the learning rate was set **too high**, the gradient descent updates overshot the minimum, causing the cost function to **diverge** instead of decreasing.

When the learning rate was **too low**, the model converged very slowly, requiring thousands of iterations before reaching a stable point.

An **optimal learning rate ≈ 0.1** achieved the best values, converging steadily and fitting the best line

Polynomial regression curve changes when the degree is changed the best fitting degree value is 3

Higher-degree polynomials risk **overfitting**, so degree = 3 was chosen as the balance between accuracy and generalization.

**Logistic Regression**

* Accuracy: Achieved ≈ 70%, which initially seems reasonable. However, accuracy alone can be misleading in imbalanced datasets (more students fail than pass).
* Precision & Recall:
  + Precision = 0.17, only ~17% of predicted “Pass” students were actually correct.
  + Recall = 0.19, the model identified only ~19% of actual passing students.
  + This shows the model is biased toward predicting Fail, likely due to imbalance in the dataset.
* F1 Score: The F1 score was 0.18, confirming poor balance between precision and recall.
* ROC Curve & AUC:
  + The ROC curve demonstrated limited separation power, with AUC ≈ 0.5–0.6.
  + An AUC close to 0.5 indicates the model is only slightly better than random guessing.
  + This highlights that Total Hours alone is not a strong predictor of passing the course.
* Threshold Sensitivity:
  + Using the default 0.5 decision threshold produced low recall.
  + Lowering the threshold (e.g., 0.3–0.4) could improve recall, but at the cost of even lower precision.

**Scikit-learn Comparison**

**Scikit-learn Comparison**

* **Convergence:**
  + sklearn’s LinearRegression uses the **Normal Equation** (closed-form solution) and converges instantly without iterative gradient descent.
  + sklearn’s LogisticRegression uses optimized solvers that are numerically stable and much faster.
* **Performance Metrics:**
  + sklearn produced **identical parameter estimates and predictions** compared to the manual implementation (when learning rate and iterations were tuned properly).
  + This validated the correctness of the from-scratch implementation.
* **Numerical Stability:**
  + Manual implementation occasionally encountered **NaN values** due to log(0) in the cost function.
  + sklearn handles these edge cases internally using numerical safeguards.

**A screen shot of a graph

AI-generated content may be incorrect.Linear regression with sklearn**

**A screen shot of a graph

AI-generated content may be incorrect.Polynomial Regression with Sklearn**

# Challenges Faced

**Challenges**:

* Understanding the math behind gradient descent and cost functions in all algorithms.
* Debugging NaN issues in logistic regression due to log(0).
* Handling imbalanced dataset (many students fail).
* Had to manually add an **X bias column** to include the intercept term.
* Sometimes had to **standardize Y** for plotting smooth curves (X\_curve, Y\_curve) so visuals aligned properly.

**Solutions**:

* Added epsilon to prevent log(0).
* Normalized features when using multiple values.
* Verified results by comparing with sklearn.

**Learnings**:

* Gradient descent requires careful learning rate tuning.
* The **bias term** is essential for correct predictions.
* Accuracy alone can be misleading precision, recall, F1, and AUC give better insight.
* Implementing from scratch builds intuition for how sklearn functions work internally.
* Implementing from scratch gave me intuition on why libraries like Scikit-learn automate these preprocessing steps.

# Conclusion & Future Work

* **Summary**:
  + Successfully implemented Linear and Logistic Regression from scratch.
  + Learned the importance of gradient descent, normalization, and evaluation metrics.
  + sklearn confirms correctness of implementation.
* **Future Improvements**:
  + Handle class imbalance (oversampling/undersampling).
  + Add regularization (L1/L2 penalty).
  + Extend to multi-class classification tasks.
  + Try more advanced optimization methods