**Mini\_Project\_00\_Report**

Dilshan Kohilamulla(0491)

Table of Contents

[Executive Summary 3](#_Toc206748069)

[Objectives 3](#_Toc206748070)

[Approach 3](#_Toc206748071)

[Key Findings 3](#_Toc206748072)

[Implementation 4](#_Toc206748073)

[Algorithm Details 4](#_Toc206748074)

[Evaluation Metrics 5](#_Toc206748075)

[Design decisions 5](#_Toc206748076)

[Results 6](#_Toc206748077)

[Learning rate analysis 6](#_Toc206748078)

[Model comparison 6](#_Toc206748079)

[Performance metrics 7](#_Toc206748080)

[Challenges and Learnings 8](#_Toc206748081)

[Difficulties faced 8](#_Toc206748082)

[Solutions 8](#_Toc206748083)

[Insights 8](#_Toc206748084)

[Conclusion 9](#_Toc206748085)

[Summary 9](#_Toc206748086)

[Future improvements 9](#_Toc206748087)

# Executive Summary

## Objectives

The primary objective was to implement linear, polynomial, and logistic regression algorithms from scratch to predict student performance, specifically the CapstoneScore (continuous: 0-100, or binarized: ≥75 as success) from the zuu\_crew\_scores.csv dataset. The dataset includes 150 samples with features like EducationLevel, Attendance, TotalHours, AssignmentsCompleted, HackathonParticipation, GitHubScore, and PeerReviewScore. The goal was to demonstrate core machine learning concepts—gradient descent, feature engineering, and evaluation metrics—without relying on external ML libraries like scikit-learn for algorithm implementation, emphasizing educational clarity and practical understanding.

## Approach

The approach involved developing Python-based implementations using NumPy and Pandas across six Jupyter notebooks. Key steps included:

* **Data Preprocessing**: Standardized features (mean=0, std=1), dropped non-numeric columns, and binarized targets for classification.
* **Algorithms**: Linear regression (single and multi-feature), polynomial regression (up to degree 5), and logistic regression, all optimized via gradient descent (except polynomial multi-feature, using least squares). Custom functions computed costs (MSE for regression, log-loss for logistic) and gradients.
* **Evaluation**: Metrics included MSE, R², MAE, RMSE for regression; accuracy, precision, recall, F1, and AUC for classification. Train-test splits (80/20) and k-fold cross-validation (k=5) were used selectively.
* **Visualization**: Matplotlib plots for cost convergence, predicted vs. actual, ROC/PR curves, and feature importance.

## Key Findings

Single-feature linear and polynomial models (using Attendance) yielded low performance (R² ≈ 0.2, test R² ≈ 0.1), indicating weak predictive power. Multi-feature models improved significantly, with polynomial degree 3 achieving the highest cross-validated R² ≈ 0.40, though higher degrees showed overfitting. Logistic regression for binary classification achieved 73.33% accuracy (no split) and 63.33% on full data with threshold 0.5 (precision 0.7143, recall 0.6250, F1 0.6667, AUC ≈ 0.78). Higher learning rates (0.1-1) accelerated convergence for linear models, but polynomials required lower rates (0.0001) to avoid instability. Key predictors included Attendance and AssignmentsCompleted. The implementations highlighted the need for regularization and robust validation to prevent overfitting.

# Implementation

## Algorithm Details

Linear regression:

* **Single-Feature (Notebook 1)**: Modeled as y = β₀ + β₁x, where x is scaled Attendance. Cost function:

Gradients:

∂J/∂β₀ = (1/m) \* Σ(ŷ⁽ᵢ⁾ - y⁽ᵢ⁾), summed from i=1 to m

∂J/∂β₁ = (1/m) \* Σ(ŷ⁽ᵢ⁾ - y⁽ᵢ⁾) \* x⁽ᵢ⁾, summed from i=1 to m

Gradient descent updated β ← β - α \* ∇J, with learning rates [0.00001, 0.001, 0.1, 1] and 10,000 iterations (tolerance 0.000001).

* **Multi-Feature (Notebook 3)**: Generalized to y = Xθ, where X includes a bias column. A LinearRegressionGD class managed fitting with 1000 epochs, learning rate 0.01, and bias term.

#### Polynomial Regression

* **Single-Feature (Notebook 2)**: Created X\_poly = [1, x, x², ..., x^d] for degrees 1-3. Used gradient descent with learning rate 0.0001 to handle feature magnitude growth.
* **Multi-Feature (Notebook 4)**: Generated all polynomial terms up to degree d using itertools. Fitted via least squares (np.linalg.lstsq) for efficiency, as gradient descent was slower for high-dimensional features. Cross-validation (k=5) compared degrees 1-5.

#### Logistic Regression

* **Notebook 6**: Modeled binary classification with sigmoid:

σ(z) = 1 / (1 + e^(-z)), clipped between -500 and 500.

Cost (binary cross-entropy):

J(θ) = -(1/m) \* Σ[y \* log(h + ε) + (1-y) \* log(1-h + ε)], summed from i=1 to m, with ε = 10^(-15).

Gradients:

∇J = (1/m) \* Xᵀ(h - y)

Trained with learning rate 0.1, 1000 iterations. Predictions used threshold 0.5.

## Evaluation Metrics

 **Regression (Notebooks 2-4)**: Implemented MSE, MAE, RMSE, R² manually. Notebook 4 used cross-validation to average metrics.

 **Classification (Notebook 7)**: Added confusion matrix, accuracy, precision, recall, F1, ROC (via threhold sweep), and AUC (trapezoidal rule). Tested thresholds [0.3, 0.5, 0.7].

## Design decisions

 **No External ML Libraries**: Emphasized fundamentals, using NumPy for math and Matplotlib for plots.

 **Standardization**: Applied before polynomial expansion to prevent numerical issues.

 **Fixed Hyperparameters**: Learning rates and iterations were hardcoded for simplicity, with experimentation in Notebook 1.

 **No Regularization**: Focused on baseline models, noting overfitting as a limitation.

 **Plots**: Included cost convergence, predicted vs. actual, and ROC/PR curves for interpretability.

# Results

## Learning rate analysis

In single-feature linear regression (Notebook 1), learning rates [0.00001, 0.001, 0.1, 1] were tested. Final costs:

* 0.00001: 1885.2590 (slow convergence).
* 0.001, 0.1, 1: ≈303.0199 (rapid convergence, <1000 iterations).

Plots showed higher rates stabilized faster, with no divergence, due to the simple loss landscape. Polynomial regression (Notebook 2) used 0.0001 to handle larger gradients from polynomial terms, converging in <1000 iterations.

## Model comparison

 **Single-Feature Linear (Notebook 1)**: No explicit train-test metrics, but cost ≈303 suggests moderate fit.

 **Single-Feature Polynomial (Notebook 2)**: For degrees 1-3:

* Degree 1: Train MSE 580, R² 0.17; Test MSE 664, R² 0.05.
* Degree 2: Train MSE 556, R² 0.21; Test MSE 629, R² 0.10.
* Degree 3: Train MSE 554, R² 0.21; Test MSE 630, R² 0.10.
* Test R² plateaued, indicating overfitting and weak predictive power of Attendance alone.

 **Multi-Feature Linear (Notebook 3)**: Truncated metrics, but plots suggest improved fit (R² est. 0.25). Coefficients showed Attendance, AssignmentsCompleted as dominant.

 **Multi-Feature Polynomial (Notebook 4)**: Cross-validated R²:

* Degree 1: ~0.25
* Degree 2: ~0.35
* Degree 3: ~0.40 (peak)
* Degree 4: ~0.38
* Degree 5: ~0.32 (overfitting).
* Bar plot confirmed degree 3 as optimal.
* **Logistic Regression (Notebook 5)**: Accuracy 73.33% on full data (no split), cost 0.4753.

## Performance metrics

* **Regression**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Degree** | **Train MSE** | **Train R²** | **Test MSE** | **Test R²** |
| Single-Feature Linear | 1 | 580 | 0.17 | 664 | 0.05 |
| Single-Feature Poly | 2 | 556 | 0.21 | 629 | 0.10 |
| Single-Feature Poly | 3 | 554 | 0.21 | 630 | 0.10 |
| Multi-Feature Linear | 1 | ~400 (est.) | ~0.25 | ~450 (est.) | ~0.20 |
| Multi-Feature Poly | 3 | - | 0.40 | - | ~0.35 |

* **Classification (Notebook 7, thresholds)**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Threshold** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| 0.3 | 0.6267 | 0.7051 | 0.6250 | 0.6627 |
| 0.5 | 0.6333 | 0.7143 | 0.6250 | 0.6667 |
| 0.7 | 0.6400 | 0.7237 | 0.6250 | 0.6707 |

* **AUC**: ≈0.78 (ROC plot, Notebook 7).

Plots (cost convergence, predicted vs. actual, ROC/PR) confirmed stable training and moderate performance, with multi-feature polynomial degree 3 outperforming others.

# Challenges and Learnings

## Difficulties faced

 **Numerical Stability**: Logistic regression faced overflow in sigmoid (solved by clipping) and log(0) in cost (mitigated with ε=10^(-15)). Polynomial features amplified gradients, requiring low learning rates.

 **Overfitting**: Polynomial models (degrees >3) showed test R² decline (Notebook 4), due to lack of regularization. No train-test split in Notebooks 1 and 5 inflated metrics.

 **Dataset**: Small size (150 samples) caused high variance; single-feature models had low signal (R² ≈0.2).

 **Implementation**: Manual k-fold (Notebook 4) was labor-intensive; redundant code across notebooks (e.g., scaling) reduced efficiency

## Solutions

 Clipping and ε for stability.

 Cross-validation (Notebook 4) for degree selection.

 Standardization before polynomial expansion.

 Manual metric functions ensured correctness.

## Insights

 Standardization and low learning rates are critical for polynomial models.

 Multi-feature models capture more variance but risk overfitting without regularization.

 From-scratch coding clarified gradient descent mechanics and library internals.

 Validation (train-test, cross-val) is essential for generalization

# Conclusion

## Summary

The project implemented linear, polynomial, and logistic regression from scratch, achieving moderate performance (R² ≈ 0.40 for polynomial degree 3, classification accuracy ≈ 63-73%). Multi-feature models outperformed single-feature, with Attendance and AssignmentsCompleted as key predictors. Logistic regression showed balanced precision/recall, with AUC ≈ 0.78.

## Future improvements

 Add regularization (L2/Ridge) to penalize high-degree polynomial coefficients.

 Implement adaptive optimizers (e.g., Adam) or learning rate schedules.

 Universal train-test splits and cross-validation for all models.

 Explore feature selection or ensembles for better performance.

 Test on larger datasets and automate hyperparameter tuning (e.g., grid search for learning rates, degrees).