Diabetes Prediction Project — Report

Introduction

This project implements logistic regression from scratch and compares it with classical ML models to predict diabetes (Pima Indians dataset). The goal was learning: understanding the mathematics, building a clean pipeline, and producing an interview-ready repo.

Objectives

- Implement logistic regression (sigmoid, BCE, gradient descent) from scratch.

- Build a clean preprocessing pipeline handling invalid zero-values.

- Apply L2 regularization and measure performance.

- Compare with Decision Tree, Random Forest, and XGBoost (practical tuning).

- Document lessons learned and produce reproducible notebooks.

Dataset

Pima Indians Diabetes Dataset (768 samples, 8 numeric features). Target: Outcome (1 = diabetes, 0 = no diabetes). Notable issue: some features use 0 as a placeholder for missing values (Glucose, BloodPressure, SkinThickness, Insulin, BMI).

Preprocessing & Key Decisions

1. Split data into train and test BEFORE any preprocessing to avoid data leakage.

2. Replace biologically impossible zeros with NaN for: Glucose, BloodPressure, SkinThickness, Insulin, BMI.

3. Impute missing values using the median computed from the training set only.

4. Standard scale features (z-score) using means and stds from training set.

5. Save preprocessing as a reusable module (`preprocessing.py`).

Models Implemented

1. Logistic Regression (from scratch using NumPy): sigmoid, numerically-stable BCE, vectorized gradient descent, L2 regularization.

2. Decision Tree (sklearn) — default and tuned shallow tree.

3. Random Forest (sklearn).

4. XGBoost (xgboost) — tuned for small data (shallow trees, lower learning rate, subsampling, class weight).

Results (selected)

Logistic Regression (scratch):

- Accuracy: ~0.7446, Precision: ~0.6719, Recall: ~0.5309, F1: ~0.5931

Random Forest (default):

- Accuracy: ~0.7489, Precision: ~0.6769, Recall: ~0.5432, F1: ~0.6027

XGBoost (tuned):

- Accuracy: ~0.7662, Precision: ~0.6364, Recall: ~0.7778, F1: ~0.7000

Note: Small changes in hyperparameters (like n_estimators) had noticeable effects; tuned XGBoost improved recall and F1.

Challenges Faced

- Invalid zero values: Several features used 0 to mean missing. Fixing this was critical.

- Numerical stability: Sigmoid and BCE required careful handling (clipping probabilities, stable sigmoid) to avoid NaNs.

- Scaling & leakage: Learned the importance of fitting imputation and scaling only on training data.

- Small & noisy dataset: Limited signal meant model improvements were modest unless features or objective (thresholds) were adjusted.

- Overfitting with complex models: XGBoost/RandomForest needed careful regularization and shallow trees for this dataset.

Key Learnings

- Preprocessing matters more than small algorithmic tweaks—fixing missing values and scaling increased model stability.

- Implementing algorithms from scratch deepens understanding (derivation of gradients, effect of regularization).

- Evaluation must use precision/recall/F1 and not just accuracy, especially in medical tasks.

- For small datasets, ensembles and boosting may overfit; careful tuning helps but data quality/feature engineering are more effective routes.

Practical Recommendations

- For improving recall (reduce missed diabetics): adjust classification threshold and/or use class weighting (or scale_pos_weight in XGBoost).

- Try feature engineering: domain-informed ratios/interactions (e.g., Glucose/BMI), polynomial features, or binning.

- If more data is available, consider tree ensembles or neural networks with regularization.

How to Reproduce

1. Install requirements from requirements.txt.

2. Run notebooks in order:

  - 01_eda.ipynb (exploration)

  - 02_preprocessing_check.ipynb (validate preprocessing)

  - 03_logistic_regression_training.ipynb (train from-scratch model)

  - 04_tree_randomforest_experiments.ipynb

  - 05_xgboost_experiment.ipynb

3. All core logic is in `src/` (preprocessing.py, logistic_regression.py).

Conclusion

This project achieved its goal: building and understanding logistic regression from scratch, and applying practical ML tools to a real dataset. We improved the practical model by tuning XGBoost and handling class imbalance. The project is structured, reproducible, and interview-ready.