Machine learning - Project 1

Matthias Wyss, Donia Gasmi, Alexandre Huou
EPF Lausanne, Switzerland

***Abstract*—This report is part of Project 1 for the Machine Learning course at EPFL [1]. The objective of this work is to train and evaluate a model capable of predicting the likelihood of a person developing a cardiovascular disease based on their personal lifestyle factors. All methods are implemented from scratch using only standard Python libraries and NumPy.**

I. Introduction

Cardiovascular diseases (CVDs), including heart attacks, are a leading global cause of death. Advances in machine learning offer promising tools for early detection and prevention of CVDs. In this project, part of EPFL's Machine Learning course [1], we aim to predict the risk of developing coronary heart disease (MICHD) using clinical and lifestyle data from the 2015 Behavioral Risk Factor Surveillance System (BRFSS) Survey [2]. Our binary classifier outputs -1 for healthy individuals and 1 for those diagnosed with MICHD, implemented from scratch in Python and NumPy. We begin by analyzing and preprocessing the data, implementing the required ML methods, and ultimately comparing different model parameters through cross-validation to optimize an algorithm suitable for predicting new data.

II. Models and methods

A. Implementation of the 6 Required models

- `mean_squared_error_gd`: This function implements linear regression using gradient descent. It iteratively updates the model parameters to minimize the mean squared error (MSE) loss, which includes a factor of 0.5 for consistency with the lecture notes.
- `mean_squared_error_sgd`: This function updates model parameters using stochastic gradient descent (SGD) with a mini-batch size of 1, offering computational efficiency.
- `least_squares`: This function calculates the least squares solution for linear regression using the normal equations, without using `numpy.linalg.lstsq`.
- `ridge_regression`: This function implements ridge regression, a regularized form of linear regression that penalizes large weight values through a regularization term λ.
- `logistic_regression`: This method performs logistic regression using gradient descent for binary outcomes ($y \in \{0, 1\}$).
- `reg_logistic_regression`: This function implements a regularized form of logistic regression using gradient descent, incorporating a regularization term $\lambda\|w\|_2$ to prevent overfitting.

B. Data Analysis and Preprocessing Methods

Data exploration phase: The dataset consists of $328,135$ samples and 321 features, all containing numerical data. The test set includes $109,379$ individuals, intended for use in a forecasting competition hosted on AIcrowd.
The overall percentage of NaN values in our dataset is 44.79% (more details in Table 1).
We first manually selected 91 key features using the provided feature-background document. We took on the ones classified as 'calculated variables,' assuming they held meaningful information, since human intervention was imployed to calculate them. However, automated preprocessing techniques yielded better results. And so, we used the full dataset.

	$< 25\%$	$[25, 50]\%$	$[50, 75]\%$	$> 75\%$
Columns	44.548%	9.657%	9.346%	36.449%
Rows	0.000%	85.885%	14.115%	0.000%

Table I
NaN Values Percentages in Columns and Rows

Data preprocessing phase:
We experiment with various preprocessing techniques.

- **Convert Label -1 to 0**: Efficient computation of the loss and gradient using sigmoid and log operations.
- **Remove Single-Value Columns:** Columns with a constant value across all entries were removed, as they lack variance and do not aid in model learning.
- **Handle Class Imbalance**: Since only 8.83% of individuals experience a heart attack, the model could become biased towards predicting the majority class, resulting in poor performance. To mitigate this, we apply:
  - *Downsampling*: Reduce the majority class by removing random rows.
  - *Upsampling*: Increase the minority class by duplicating random rows.
  - *Data Augmentation*: Add noise to duplicated data to improve generalization and reduce overfitting.
- **Handle Missing Values (NaN)**: We test 2 approaches:
  - *Removal of Features*: We remove columns with more than 80% NaNs, eliminating 116 columns.
    Imputation: For remaining NaNs, continuous features are filled with the average and categorical ones are randomly filled based on their distribution.

- *Encoding Missingness*: We encode all NaNs as -1, allowing the model to interpret missingness as a unique indicator. This approach may capture patterns linked to the absence of specific information in health records.

- **Feature Scaling**: To handle the impact of variables with different scales, we experiment with two approaches:
  - *Normalization*: Rescals feature values to range $[0, 1]$.
  - *Standardization*: Centers features to have $\mu = 0$ and $\sigma = 1$
- **Handle Outliers**: The data being standarized ($\mu = 0, \sigma = 1$) We identify outliers using a z-score threshold of 3, and we test 2 approaches:
  - *Removing Outliers*: Rows exceeding a set percentage (30%) of outlier values are removed.
  - *Clipping Outliers*: We clip outliers to the threshold value. By capping extreme values, we retained the data without removing entire rows.
- **Principal Component Analysis (PCA)**: Reduce dataset dimensionality by retaining only the most significant components, based on a feature retention ratio that we meticulously hyper-tune (See Figure 1).

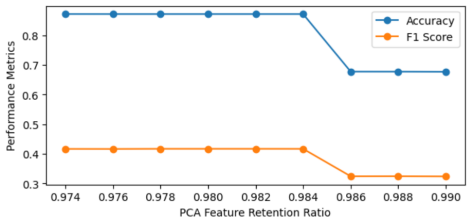

Figure 1. Effect of PCA on Model Performance.

- **Add Bias Term**: Allow the model to capture data patterns that don't intersect at the origin.

After cross-validation, we select the following:
1. Label Conversion 2. Removing Single-Value Columns 3. Handling NaN Values (Encoding Missingness with -1) **4. Standardization 5. Class Imbalance Handling** (Upsampling the minority class from 8.83% to 20%) **6. Principal Component Analysis (PCA)** (98% of features retained)

III. Results

A. Model and hyper-parameters selection

We evaluate our models using accuracy and F1 score as metrics, prioritizing F1 as it is crucial in the medical context for minimizing false negatives.
We fix Maximum Iterations at 1000 for all models.

We use 5-fold cross-validation to select the best parameters for each model, then identify the top performer overall. Consistent performance across folds (Table 3) indicates robust generalization on unseen data (small standard deviations).

Hyperparameters	Values
λ (Penalty weight)	$[10^{-4}, 10^{3}]$ (8 values in logspace)
γ (SGD step size)	$[10^{-4}, 10^{3}]$ (8 values in logspace)
Non-log boundaries	$[0, 0.29]$ (in increments of 0.01)

Table II
TESTED HYPERPARAMETERS VALUES

Method	Accuracy	F1-Score	γ	λ
least_squares	0.859 $\pm$(1.3e$-$3)	0.416 $\pm$(4.9e$-$3)	n/a	n/a
mean_squared_error_sgd	0.651 $\pm$(1.6e$-$3)	0.303 $\pm$(1.1e$-$3)	3e$-$4	n/a
ridge_regression	0.722 $\pm$(0.5e$-$3)	0.347 $\pm$(1.6e$-$3)	n/a	1e$-$5
logistic_regression	0.831 $\pm$(34e$-$3)	0.329 $\pm$(4e$-$4)	1e$-$4	0
reg_logistic_regression	0.633(8.1e$-$3)	0.285(3.6e$-$3)	1e$-$4	1

Table III
MODELS' PERFORMANCE SUMMARY

The least squares model, having the highest F1 score, is selected as the best model. When optimizing Least Squares we also use cross validation to hyper-tune the decision threshold for optimal performance.

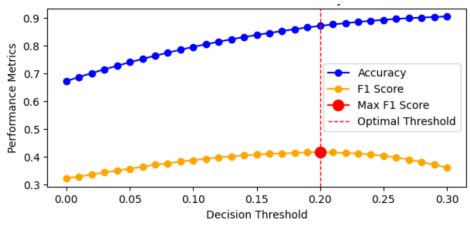

Figure 2. Decision Threshold Tuning.

We use this best model on the training set, obtaining optimal parameters and tracking the training loss. Using these parameters, we predict binary labels on the test set and convert back our output from 0 and 1 to -1 and 1.

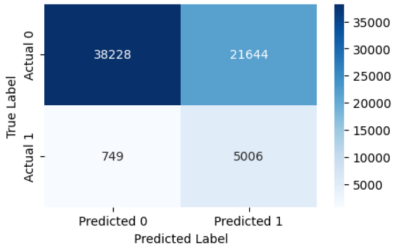

Figure 3. Final Model: Confusion Matrix

IV. Discussion Summary

One key challenge we faced is data leakage. Preprocessing before the train-test split allows validation data to influence results. To fix this, we integrated preprocessing within cross-validation, isolating each training fold from its test fold and ensuring a fair model evaluation. This project highlighted the importance of data cleaning, preprocessing, balancing, and parameter tuning for building an effective prediction model.

REFERENCES

[1] N. Flammarion and M. Jaggi, "Machine learning cs-433." [Online]. Available: https://www.epfl.ch/labs/mlo/machine-learning-cs-433/

[2] U. D. of Health Human Services, "2015 brfss survey data and documentation." [Online]. Available: https://www.cdc.gov/brfss/annual_data/annual_2015.html