Machine Learning - Project 1

Shrinidhi Singaravelan - Salya Diallo - Fanny Ghez
Machine Learning CS-433, EPFL Lausanne, Switzerland

***Abstract*—With the help of machine learning, this study aims to predict cardiovascular disease risks based on personal lifestyle factors. The research explores different types of prediction models, trained over a large provided dataset, and then tested on a separate one. Diverse implementation approaches will be explored to enhance predictive accuracy using real-world data, and the findings will be analyzed to gain insights into the most effective models and features.**

I. Introduction

The global rise in Cardiovascular Diseases (CVDs), that includes conditions like heart attacks presents a substantial concern worldwide. As people's lifespans continue to increase, the elderly start getting more and more heart diseases and prevalent circulatory vessels. Luckily, technology growth allows Machine learning to give the potential to take proactive measures to prevent the onset of CVDs. Our project aims to use the power of Machine learning, using various functions developed during the course, to forecast an individual's risk of developing a CVD, utilizing their way of living as key parameters..

II. Data Analysis

A. Data set

For our project, we are given 3 main data-files : `y_train`, `x_train` and `x_test`. The training set focuses on 328,135 individuals, regarding to 322 features documented in `x_train` and the `y_train` file indicates by +/-1 values if the corresponding subject presents CVD condition. The testing set `x_test` contains data of 109,379 individuals, in order to implement our trained ML methods to predict labels.

B. Balancing

The provided data-set is imbalanced as it presents a prevalence of positive cases (approx. 92%). We don't consider this as critic but we will have to consider its effect upon the effectivity to capture the minority class.

C. Data set pre-processing

When looking at the data, we can easily notice that a lot of values are missing. We first need to replace all the empty cases by the median value computed over the whole column concerned. We also tried replacing it by the mean value but found it to be sensitive to outliers, we thus opted for the most robust and resilient alternative.

D. Data splitting

Before going any further trying to predict CVDs over our data set, we divided the database into two groups based on gender to account for potential variations in disease prevalence between men and women, and because it has no missing values. This way, with each of the following ML methods implemented, we will obtain different predictions between male and female individuals, as we split the data set accordingly before training our models.

E. Standardization

As previously seen in class and labs, standardization plays a valuable role in Machine Learning, particularly when dealing with features with varying scales. This way, every feature has the same scale and standardization prevent one feature to dominate the learning process because of its larger numerical range. It will also facilitate some methods, such as gradient descent, to converge more efficiently in a shorter amount of time. Finally, adding intercepts (a column of 1) to the obtained matrices is crucial as it allows the model to have a non-zero prediction.

III. Models and Methods

Six different ML Methods have been implemented by functions we have seen during the labs. All of these function definitions can be found in the `implementations.py` file.

- Linear regression using gradient descent (GD)
- Linear regression using stochastic gradient descent (SGD)
- Least squares regression using normal equations
- Ridge regression using normal equations
- Logistic regression using GD
- Regularized logistic regression using GD

These methods include a range of machine learning techniques, each tailored to specific data-driven tasks. Whether optimizing linear models, exploring regularization or diving into logistic regression, the functions encapsulated in `implementations.py` provide a flexible toolbox for diverse modeling needs of our project. In the following sections, we will examine the performance, nuances and results obtained with these methods, providing a comprehensive analysis of their effectiveness in addressing our prediction challenges.

IV. Results

A. Parameters fitting

- Linear regression: Both functions `mean_squared_error_gd` and `mean_squared_error_sgd` require two adjustable parameters `gamma` (step size) and `max_iters` (maximum number of iterations). Choosing a step size of 0.001 and 100 iterations in both cases seemed an appropriate trade-off between good accuracy and reasonable running time.
- Ridge regression : Regarding this method, choosing `lambda_` $= 0.1$ seemed to produce more accurate results.

- Logistic regression and Regularized logistic regression (RLR) : these 2 functions also need arguments `gamma` (step size) and `max_iters`. Once again, our choice went for 0.001 and 100 respectively. Additionally, the RLR method needed a `lambda_` parameter as well that we set on 0.01 for accuracy's sake. For RLR, more iterations were needed to gain a satisfying balance between accuracy and F1 score: we have tested 100, 400 and 1000.

B. Running time

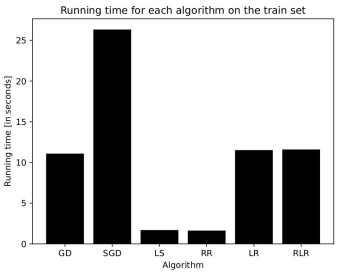

Fig. 1. Running time of each algorithm using the parameters given in the section *Parameters fitting*

The running time of each function can be found in **Fig. 1**. This plot clearly illustrates that methods based on matrix computations (LS and RR) are notably more time-efficient than their iterative counterparts.
The second method is significantly the slowest one. This is due to the stochastic nature of SGD with a mini-batch size of 1, resulting in a more noisy training process, which may require more iterations to converge to a satisfactory solution. Larger mini-batches would provide more stable updates.
Least squares and ridge regression tend to have faster running times compared to gradient-based optimization methods because they directly compute the model parameters using the normal equations. These equations have closed-form solutions, making them more efficient. In contrast, gradient-based methods require iterative updates, which introduce additional computational overhead.

C. Accuracy

Method	Accuracy
Linear regression using GD	84.3%
Linear regression using SGD	55.8%
Least squares regression using normal equations	78.7%
Ridge regression using normal equations	78.8%
Logistic regression using GD	90.3%
Regularized logistic regression using GD	89.0%

TABLE I
ACCURACY OF THE DIFFERENT IMPLEMENTED METHODS

Logistic regression methods stand out as the top-performing methods, achieving an accuracy around 90%. This model excels in binary classification tasks, making it a robust choice for predictive accuracy. For RLR, we used the results obtained when doing 1000 iterations. A small predominance of LR can be observed: this may illustrate the fact that the logistic loss explains better our data set. On the other end of the spectrum, linear regression with SGD, despite its computational simplicity, lags behind with an accuracy of 55.8%. The noisy updates and reliance on small mini-batch sizes contribute to its lower accuracy.
Overall, all of these 6 different methods achieved accuracy levels superior to 50% (and even more than 75%, except for linear regression using SGD). This translates the effectiveness of all ML algorithms used in this project for classification task, despite a very large data set counting many features.

V. Discussion

A. Data pre-processing

The careful handling of missing values, standardization, and other pre-processing steps we operated not only ensures data quality but also significantly influences the results obtained. It sets the foundation for accurate model training. Proper data pre-processing is the cornerstone of reliable, robust, and interpretable machine learning outcomes. A large improvement in accuracy was made by standardization when tackling the case of the standard deviation being zero. We replaced it by a very small value ($1e^{-6}$) to avoid division by zero.
Nevertheless, while extensive data pre-processing efforts can enhance method accuracy, the imbalanced distribution of values in the training data set (strong prevalence of positive cases) remains a limiting factor, contributing to potential errors in predictions.

B. Parameter fitting

This step was crucial for obtaining an optimal accuracy score. Due to the high dimensionality of the data set (322 features), parameter adjustments for functions requiring fine-tuning (detailed in *section IV-A*) e.g., step size and max iterations were performed manually and incrementally. Concerning finding the best fitted `lambda_` parameter for Ridge Regression, we used a method of 4-fold cross-validation but did not get a better result since we obtained the same accuracy.

C. Code modularization

Through the use of code modularization, we managed to gain in time efficiency and accuracy when defining functions for computing MSE and logistic loss.

D. Model efficiency

Based on the obtained results of sections *IV-B* and *IV-C*, we observe an obvious trade-off between computation efficiency and accuracy (i.e. the most accurate functions tend to be slower). In this case of study, the prediction program is made for medical conditions. As it relies more on accuracy than time efficiency, the most appropriate method would be the one obtaining the best accuracy result : logistic regression. While this method may not provide the fastest execution time, it falls within the range of a few tens of seconds, which remains quite suitable for an application in the medical field.

VI. Summary

This project emphasizes the critical role of data analysis and pre-processing in machine learning. Through the use of different optimization techniques, we successfully constructed several models with highly favorable outcomes. Ultimately, logistic regression emerged as the model with the highest final accuracy.