Predicting Heart Attacks: A Machine Learning Approach Using Real-World Data

Aziz Laadhar, Yassine Chaouch, Antoine Merel
Department of Computer Science, EPFL, Switzerland

***Abstract*—This project relies on data analysis and machine learning techniques to generate predictions on a data set of cardiovascular diseases.**

I. INTRODUCTION

Cardiovascular diseases, such as heart attacks, are a significant global health concern. Early detection and prevention of these diseases are crucial in reducing their impact. In this project, we applied machine learning techniques to predict the risk of developing myocardial infarction and coronary heart disease (MICHD) based on personal lifestyle factors. Using a real-world data set from the Behavioral Risk Factor Surveillance System (BRFSS), we aimed to build a model that could estimate the likelihood of developing MICHD. In this paper, we provide a concise overview of our methodology, including data preprocessing, feature engineering, and the implementation and evaluation of several machine learning models.

II. EXPLORING DATA ANALYSIS

A necessary first step is exploring and understanding the given data. In our case the data consists of a training set (328,135 instances) and a testing set (109,379 instances) each having 321 features taken from Behavioral Risk Factor Surveillance System's survey where respondents were classified as having coronary heart disease if they reported having been told by a provider they had MICHD or if they reported having been told they had a heart attack or angina.

III. DATA PREPROCESSING

Processing is key to reveal the information that lies within the data. In the following sections, we describe all the techniques that were used, tried and possibly dropped to improve the quality of the information fed to our model.

A. Data Cleaning

As illustrated in Figure1, The data came with a substantial amount of missing values. Therefore in the initial cleaning

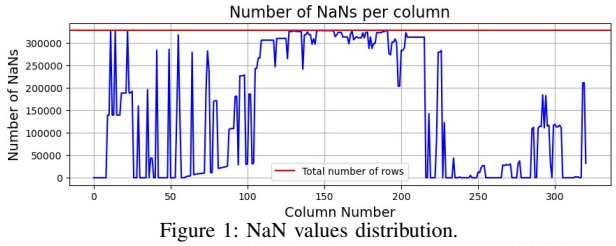

Figure 1: NaN values distribution.

phase, we began by eliminating columns exhibiting an excess of missing data, specifically those surpassing a 50% threshold.Notably, a substantial majority of our data features, approximately 70%, contain 12 or fewer distinct values as shown in Figure2.So, To address the gaps in the remaining dataset, we opted for imputation via median values. This choice was motivated by the robustness of the median to outliers and the prevailing categorical nature of our data.

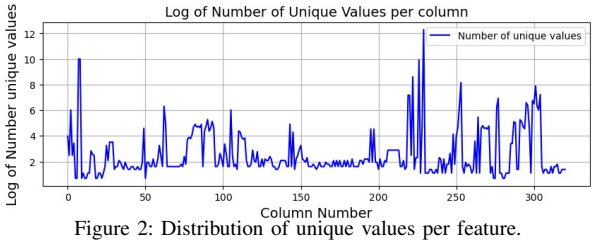

Figure 2: Distribution of unique values per feature.

B. Data Balancing

Additionally, a crucial observation we made pertains to the pronounced data class imbalance. A mere 9% of the instances belong to the positive class, while a substantial 91% are associated with the negative class, profoundly influencing the optimization of our model.

1) ***Oversampling & Undersampling:*** To reduce the effects of this imbalance, we first employed the Synthetic Minority Over-sampling Technique (SMOTE), creating synthetic samples of the minority class. This technique works by selecting a random minority class sample, finding its k-nearest neighbors, and generating new samples that lie on the line segments joining the selected sample with its neighbors. Concurrently with oversampling, we also implemented undersampling on the majority class, randomly selecting samples to reduce its size and balance the class distribution. These method ensures that the minority class holds more influence in model training, preventing model bias towards the majority class, and fostering a conducive environment for improved model training and performance.

2) ***Removing Outliers:*** In the process of data preprocessing, we initially used Z-scores to identify outliers with a Z-score greater than 5. However, this approach would have removed around 25% of the original positive samples, creating a significant data imbalance issue. As a more balanced alternative, we switched to the Interquartile Range (IQR) method, which resulted in the removal of less than 3% of the positive samples. This IQR-based approach preserved a better class balance, making it the preferred choice for our data preprocessing.

IV. FEATURE ENGINEERING

A. Features Correlation

To get the most significant features in the data we started by computing the correlation of each feature with y and

sorted them. Having highly correlated features in a data set is not desirable as it does not bring additional informations and complexifies our model. Hence, we computed the correlation factor between features and dropped the ones that were more than 90% correlated with another feature to keep only the most correlated with y. We then sorted the features according to their correlation with y and picked the first 40 features. Figure3 shows a sample of these features and their respective correlation with y.

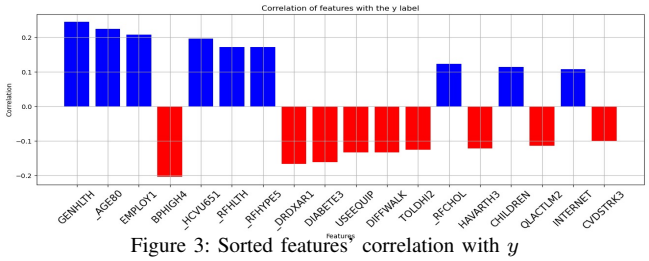

Figure 3: Sorted features' correlation with y

B. One Hot Feature Encoding

Through our inspection of the features Codebook we notice that we can divide the features into two categories:

1) ***Categorical Features:*** To handle categorical variables in our model, we selected the features that have less than 8 values and used one-hot encoding on them. This method turns categories into a binary format that our machine learning model can understand. By changing categories into binary vectors, we kept the unique nature of each category while giving our model a format it could work with easily. We also noticed that for these type of features the values 7 and 9 were assigned to non significant values (Don't know, Didn't answer) therefore we proceeded to remove them.

2) ***Non-Categorical Features:*** Although these features are not inherently categorical, they hold significant importance in enabling the model to make accurate predictions. As such, we concatenated them to the one-hot-encoded matrix.

C. Feature Expansion

To uncover hidden feature interactions, we expanded our feature set through pairwise multiplications, subsequently narrowing it down to the top 190 correlated features with the target. This approach increased the model's ability to capture complex interactions while mitigating the risks of high dimensionality and overfitting.

V. Models & Results

A. Models

To train our model, we tried different different linear regression algorithms and choose the one gave the best results as shown in Table I with Smote ration=0.8 and Undersample ratio=4 .

Lin. Regression Algorithm	Iters	λ	γ	Acc.	F1
Mean Square GD	3000	-	0.001	0.846	0.405
Mean Square SGD	3000	-	0.001	0.830	0.380
Least Squares	-	-	-	0.855	0.419
Ridge Regression	-	-	1	0.852	0.416
Logistic Regression	3000	-	0.001	0.869	0.406
Reg. Logistic Regression	10000	0.001	0.01	0.899	0.364

Table I: Results Table

B. Algorithm Choice

Our choice was motivated by the need to handle potential overfitting and improve the model's generalization performance. Reg. Logistic Regression technique is particularly useful when dealing with high-dimensional datasets, as it helps to select the most relevant features and reduce the impact of noisy or irrelevant ones. This can be also seen in Table I where Reg. Logistic Regression out-preformed the other algorithms in terms of accuracy but had a smaller F1 score than the other algorithms which needed more optimisation to increase:

1) ***Threshold Selection:*** We refined our model's classification threshold through the 'select_best_threshold' method, enhancing its performance across key metrics, particularly the F1 score as it can be seen in Figure 4.

2) ***Weighted Loss:*** To further mitigate class imbalance effects, we introduced a weighted loss function in logistic regression. We assigned higher weights to the minority class, emphasizing correct classification and penalizing misclassifications of this class more severely. So, in Regularised Logistic Regression algorithm instead of minimizing the regular $-log$ likelihood, we use a weighted version for which improvements can be seen in Table II

$$\sum_{i=1}^{N} [w_0 y_i \log(y_i) + w_1 (1 - y_i) \log(1 - y_i)] \quad ; w_0 > w_1$$

Lin. Regression Algorithm	λ	γ	Acc.	F1
Reg. Log. Regression	0.001	0.01	0.899	0.364
Weighted Reg. Log. Regression	0.001	0.01	0.853	0.413

Table II: Results Table

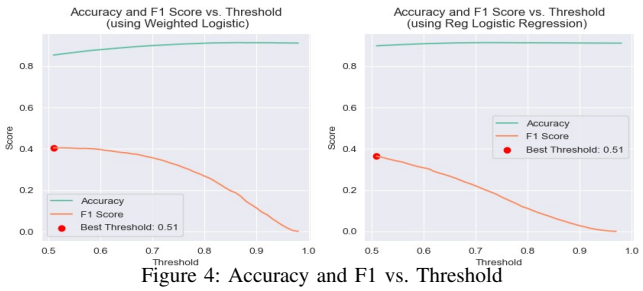

Figure 4: Accuracy and F1 vs. Threshold

VI. Results

Using Regularised Logistic Regression with the parameters shown in table I, we achieved a latest F1 score of 0.420 with an accuracy of 0.853 on AIcrowd (AYA group).

VII. Conclusion & Future Work

In conclusion, this project illustrates the potential of machine learning in healthcare, specifically in predicting the risk of MICHD. One could explore additional feature engineering techniques, such as higher degree polynomial features to capture more complex relationships in the data. Exploring ensemble models like Random Forest or Gradient Boosting may also offer improved predictive performance due to their ability to model non-linear patterns and interactions. These enhancements have the potential to refine our predictions, ultimately contributing to more accurate and reliable risk assessments in healthcare settings.