# Machine Learning Project 1 - Coronary Heart Disease Detection

Jan Fornieles Zaguirre, Jan Gambús Vinyeta, Cyriac Grégoire
*CS-433, EPFL*, https://github.com/CyriacGregoire/ML_proj1.git

***Abstract*—In this report we describe the use of a weighted and penalized logistic regression with gradient descent allowing to predict (with some precision) whether an individual will suffer a heart attack based on their answer to the 2015 BRFSS survey. Preprocessing of the data,feature engineering and selection was also done to implement the models.**

## I. Introduction

In our quest to predict the fatal diseases, we have written a two models, which differ only in a couple of approaches, and gave the same F1 score of $0.437$ on AIcrowd, one had a slightly higher accuracy. We will mainly describe the best model (referred to as John in the following) but when some differences with the second best model occur (which we will call Fox), that are worth mentioning, we will do so.
We have structured our procedure into two parts, the first explains the preconditioning of the large dataset in order to make it suitable to our approach. In the second we implement the weighted and penalized logistic regression with gradient descent. All the code can be found on our GitHub repository.

## II. Models and Methods

### A. Preconditioning the data

*1) Categorical vs Numerical features and Cleaning the NaNs:* The First step in our preprocessing was to isolate the categorical features from the numerical ones.
We simply consider a categorical feature any feature that has less than 10 different values and will deal with them differently than with the numerical ones in future steps.

Secondly we observe, that there is a very large number of features which have NaN entries (Not a Number - corresponds to unanswered questions). This poses two problems, the first being that the features with a high content of NaNs give us very little information while reducing computational cost, the second being we need numbers to implement any ML algorithm.
In order to deal with the first point, we simply remove all the features that have a content of NaNs above a threshold. The threshold we found most efficient for this (by a simple grid search) was 0.4 - note that Fox uses 0.75 but the difference in the F1 score that we obtain by increasing it from 0.3 to 0.75 is only $1.6395 \cdot 10^{-4}$. Figure 1 shows that this corresponds to only very few features (and those features give very little information).

To deal with the second point we replace the NaN values with $-1$ so that we keep the information that this is unanswered but may still use the features. (All other features take positive values.) Note that Fox takes a different approach here, we impute NaN values with the mean value of the feature for numerical features (We have also tried the median but it performed worse.) and with the mode in the case of the categorical features.

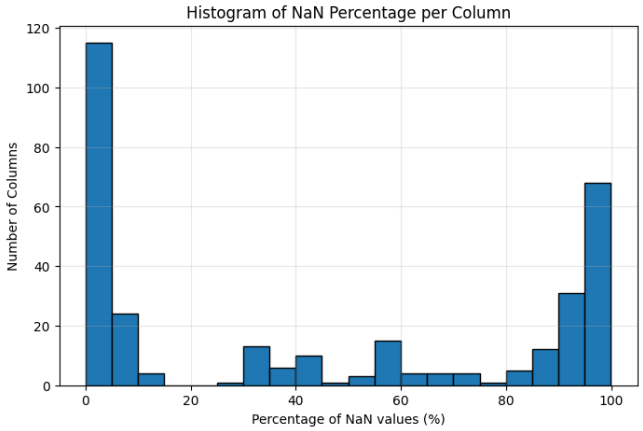


Figure 1. Number of features as a function of the NaN content.

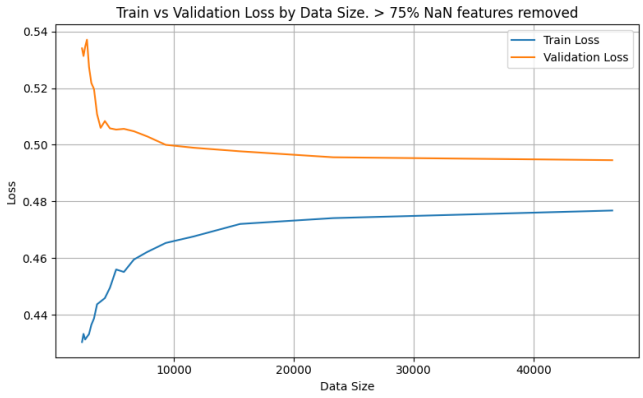


Figure 2. Training Curve - Loss as a function of the training set size - before the implementation of one hot encode.

*2) Feature selection:* Once we have dealt with the NaNs we will modify the categorical features to be more suitable to analysis. This is done via the one_hot_encode function that takes in the data and splits a categorical feature of $k$ possible values into $k-1$ categorical features each taking values in $\{0,1\}$ to indicate which answer was given. The main advantage of this transformation is that this adds more features (and hence more parameters) to the model, allowing more complexity and avoiding underfitting of the data. Indeed looking at the learning curve before (2) and after (3) this modification, we notice that the learning which previously got stuck after a given data size (i.e. lacking in complexity - this model has high bias) is now behaving well

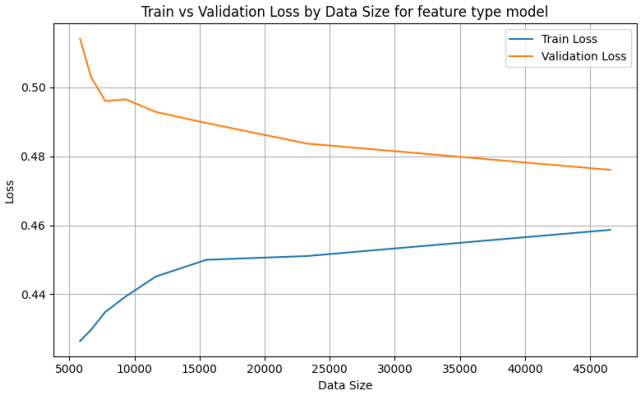


Figure 3. Training Curve - Loss as a function of the training set size - after the implementation of one hot encode.

(i.e. keep learning more with more data - we have a higher variance).
Finally we compute the correlation matrix to determine which feature's information is actually readable in another one. Here if two features (categorical and/or numerical) have a correlation above a threshold we keep only the one with the highest feature target correlation previously computed. Here we set the threshold at $0.9$ (determined via trials).

### B. The Learning Model(s)

Since the goal of this project is a classifying task, we have found the use of a weighted version of logistic regression to be most efficient, i.e. optimizing the log-likelihood that our data is observed. We therefore minimize:

$$\mathrm{L}(x,y) = \frac{1}{N \cdot \sum_{i=1}^{N} \eta_i} \sum_{i=1}^{N} \eta_i \left( log(1 + e^{x_i^T w}) - y_i \cdot x_i^T w \right) + \lambda \left\| w \right\|_2^2,$$

by using gradient descent. We use $\lambda = 10^{-8}$ and $\eta_i = 6.5 \cdot \mathbb{1}_{\{y_i=1\}} + \mathbb{1}_{\{y_i=0\}}$. To make a decision for each data point, we compute the probability $\mathbb{P}(y_i = 1|x_i) = (1 + e^{-x_i^T w})^{-1}$ of getting a heart attack for a particular point and decide to classify it as $1$ if this probability is above $\alpha = 0.7476$.
Note that while John uses $6.5 : 1$ as weights, which corresponds to a balance towards the majority class, Fox uses the exact ratio of ones to zeros in the training set (i.e. a perfect balance), we found through trials that a very large range of weights works almost equivalently well, provided we adjust the threshold $\alpha$ accordingly.
The use of the weighted logistic regression is due to the large imbalance of the data. Indeed about $91\%$ of values are zeroes which means that predicting more zeroes will give better accuracy. To solve this problem we put a weight $\eta_i$ on each data point that makes zeros 'count much less' than the ones. We have first tried using a simple undersampling of the data (oversampling was discarded due to a higher computational cost), where we select a subset of the data that conserves the $9 : 1$ ratio of ones and zeros. This was had the big disadvantage of not allowing us to use all the data at hand.

One of the main difference between John and Fox is the following: In order to get a validation set John split the data into three parts, a training set, a validation set and a testing set composed of $70\%$, $15\%$ and $15\%$ of the data respectively. It is important here that we sample each subset in a way that conserves the original class balance of the data, i.e. roughly $91\%$ zeroes and $9\%$ ones. Fox uses the classical $5$-fold cross validation, since both models are so close, and this is the main difference, we conclude that in this case the two methods give very similar results. This is likely due to the large size of the dataset.

### C. Ablation Analysis of the Model

| Added steps to the Model | F1 Score |
|---|---|
| Cross - Validation | 0.0318 |
| Balancing | 0.3940 |
| Remove features of low NaN content | 0.4071 |
| One Hot Encode | 0.4246 |
| Weighted Logistic Regression | 0.4261 |
| Inter-correlation of features | 0.4277 |

Table 1. Added steps to the models and the F1 score of the model containing all steps before it.

We finish with an Ablation analysis of the Fox Model as found in [1]. Here we start with the simplest model i.e. a simple logistic regression with $5$-fold cross validation and imputing NaNs with means. Then we add more and more steps until reaching the full Fox model and record each F1 scores computed at testing time. The results are found in table II-C. We note that the best improvement came from the balancing of the data this is reasonable since else we are likely to select a training set composed of more that $90\%$ zeros in which case it gets harder to learn anything other than: 'Predict 0!'. The second best improvement was the One Hot Encode, as explained above this allowed us to learn using more data.

## III. Conclusion

We conclude that understanding the data and its structure as well as the correct feature engineering is crucial to writing a working classifying model. Moreover we have found that various approaches worked almost equivalently well at many steps of our process and hence that many variations of the Model could be used with only minimal losses or improvements to the performance. This shows that there is a limit to how well logistic regression can perform on this kind of dataset. Using more complex models such as Neural Networks may improve the results.

## REFERENCES

[1] A. Y. Ng, "Advice for applying machine learning," https://cs229.stanford.edu/materials/ML-advice.pdf.