MICHD prediction: Machine learning project 1

Hanwen Zhang, Zihan Yu, Qiaochu Wang
CS-433, EPFL, Lausanne

***Abstract*—Machine learning is a powerful tool to predict the risk of a person developing a certain disease. And the chance of getting MICHD, a cardiovascular disease, is becoming a big concern for the elderly. In this paper, we tackle this task by preprocessing the data and applying binary classification techniques. We present our attempt to predict whether a man will develop MICHD given a certain clinical and lifestyle situation, a vector of features.**

I. Introduction

Manifest Ischemic Coronary Heart Disease (MICHD) is a kind of cardiovascular disease. As people are living longer, cardiovascular diseases are prevalent in this growing population of older adults. Here we try to develop a machine learning algorithm to determine the risk of a person developing MICHD based on features of their personal lifestyle factors using the data provided by the Behavioral Risk Factor Surveillance System (BRFSS). In order to make the best prediction, we first cleaned the data and did some feature processing. After that, we tried several machine learning algorithms for binary classification and managed to pick the model that maximized the accuracy and F1 score. We then explain which models we choose and conclude by analyzing the results we obtain.

II. Data analysis and clean

A. Data observation

We started by looking into the data file. We found that we have 32k samples and 321 features, which is too big for human observation. Meanwhile, it can be observed that there are many missing values inside the data. Clearly, we need to deal with these missing values before we feed the data into our model.

B. Categorize data

After noticing that all values are positive, we replaced all nan values with -1. We first plotted histograms of all the features in order to have an idea of the distribution of each. One of our important findings is that there are pairs of columns, e.g. 'CTELENUM & CTELNUM1' and 'PVTRESD1 & PVTRESD2', complementing each other, i.e. when 'CTELENUM' is equal to 1, the value of 'CTELNUM1' is -1, and vice versa. Furthermore, the indices of different pairs are matching, i.e. a sample with 'CTELENUM' 1 will have 'PVTRESD1' 1 and vice versa. Both facts are true for all pairs. Both facts are illustrated in figure 1. Hence, we categorize the data by splitting the data into 2 categories with the value of 'CTELENUM & CTELNUM1'.

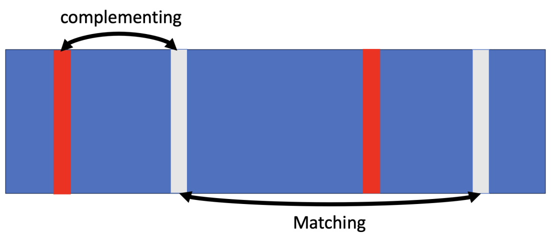

Figure 1. Illustration of pairs in the features

C. Discard irrelevant features

Now for both categories, we found that many features are now all 1 or -1 (nan). Therefore, for each category, we can discard these irrelevant features. We define the relevance as follows: if a feature has 90% of its values as 1, or conversely, 90% of its values as -1, then it is considered irrelevant and can be discarded.

D. Fill missing values

After selecting the relevant features, we drew the box plots to see the distribution of the features. Still, we spotted many features with -1 (nan) values and wanted to replace them with some reasonable values like median or mean value. We decided to replace these missing values with the median of the non-nan values for the particular feature, with the hope of not being affected by outliers.

III. Feature engineering

A. Data standardization

After plotting the box plots, we found that the value ranges of different features vary greatly. Therefore, we use Z-score standardization to standardize our data by rescaling it to mean 0 and standard deviation 1, to make the ranges consistent and help improve the model.

B. Polynomial expansion

First, we append an all-1 feature as a bias term. As some of our models like ridge regression are linear, we expand the polynomial degree of every feature to provide non-linear information. The intuition is, of course, that zwe may be able to find some hyperplane splitting the data in higher dimensional space.

C. *Resampling*

With an unbalanced data set (over 90 percent of y are -1), we find that all our models tend to give predictions in the majority class, leading to poor f1 scores. Therefore, we want to make the data balanced by adopting oversampling and downsampling techniques.

D. *Cross-validation*

To find the best hyperparameters such as λ, expansion degree d, and sampling rate s, we used k-fold cross-validation as illustrated in the lecture, to utilize all data for training and testing.

IV. Models

At first, we tried all models on the original data set. We found that they all get good accuracy but poor f1 score. Indeed we found that all models tend to predict almost -1. To optimize the f1 score, we tried different λ in the regularization terms for ridge regression and regularized logistic regression. What was confusing is that we can get a reasonable f1 score approaching 0.4 if we set λ to be very big, like 7.0 for ridge regression, which goes against our intuition of λ small. We realized that it was due to the unbalanced data and decided to adopt the resampling technique.

We first tried oversampling. We implemented a trivial SMOTE: each time select some sample with prediction 1 (the minority class), choose one of its k-nearest neighbors, and create a new sample based on this neighbor with some tiny perturbation. Note that the process is very slow because of the calculation of distance. We managed to find some optimized $\lambda = 10^{-6}$ and oversampling ratio $s = 5.0$ (which means we add 5 times more samples to the minority class). The process consumed more than 7 hours, yet the result is just of f1 score 0.395 and accuracy 0.813. Due to the limited time and hardness of reproduction, we decided to adopt the downsampling technique.

Our downsampling method is easy: given a ratio s and the number of samples in the minority class n, we keep all samples in the minority class while selecting only $s * n$ random samples from the majority class. As one can see, the process is much faster.

We decided to train two different models for the two categories of data. The algorithms we tried are ridge regression (rr) and regularized logistic regression (rlg). Over the resampling set, we use cross-validation to test different parameters λ and downsampling ratio s and get the corresponding weight. Given the weights, we tested the results in the original dataset which are shown in table I and II.

Finally, We choose the model with the best performance. We get $s_1 = 2.0$ and $\lambda_1 = 10^{-8}$ for the first category and $s_2 = 3.5$ and $\lambda_2 = 10^{-5}$ for the second. The models are both ridge regression. On AIcrowd, we get an f1 score of 0.415 and an accuracy of 0.866 for the test set.

Table I
F1 SCORE OF RR AND RLR FOR CATEGORY 1

s	λ								
	10^{-10}	10^{-9}	10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}
0.5	.323 / .347	.323 / .347	.323 / .346	.322 / .347	.322 / .347	.322 / .347	.322 / .347	.321 / .347	.320 / .346
1.0	.384 / .380	.384 / .380	.384 / .380	.384 / .380	.384 / .380	.384 / .380	.382/ .380	.381 / .380	.380 / .380
1.5	.411 / .395	.411 / .395	.411 / .395	.411 / .395	.411 / .395	.411 / .395	.410 / .395	.410 / .395	.409 / .395
2.0	.422 / .397	.422 / .397	.422 / .397	.422 / .397	.422 / .397	.422 / .397	.421 / .397	.420 / .397	.420 / .397
2.5	.414 / .388	.414 / .388	.414 / .388	.414 / .388	.414 / .388	.413 / .388	.413 / .388	.412 / .388	.412 / .388
3.0	.392 / .376	.392 / .376	.392 / .376	.392 / .376	.392 / .376	.391 / .376	.390 / .376	.388 / .376	.388 / .376

Table II
F1 SCORE OF RR AND RLR FOR CATEGORY 2

s	λ								
	10^{-10}	10^{-9}	10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}
1.5	.353 / .339	.353 / .339	.353 / .339	.353 / .339	.353 / .339	.353 / .339	.352 / .339	.351 / .339	.348 / .338
2.0	.372 / .352	.372 / .352	.372 / .352	.372 / .352	.372 / .352	.372 / .352	.370 / .352	.369 / .352	.367 / .351
2.5	.382 / .362	.382 / .362	.382 / .362	.382 / .362	.382 / .362	.382 / .362	.382 / .362	.381 / .362	.378 / .362
3.0	.391 / .366	.391 / .366	.391 / .366	.390 / .366	.390 / .366	.390 / .366	.388 / .366	.386 / .366	.385 / .366
3.5	.393 / .368	.393 / .368	.393 / .368	.393 / .368	.393 / .368	.394 / .368	.392 / .368	.390 / .368	.388 / .368
4.0	.389 / .369	.390 / .369	.389 / .369	.389 / .369	.389 / .369	.388 / .369	.386 / .369	.385 / .369	.385 / .369

We also tried the polynomial expansion technique. However, we found that any expansion to even $d = 2$ will do harm to our performance, so we just choose $d = 1$, i.e. add a bias term. We think it is because the features are already rich and expansion will get too rich features for the simple model.

For the regularized logistic regression model, we set the learning rate 0.001 and max-steps 2000. We tried different parameters in the range $0.0001 \to 0.01$ and $1000 \to 20000$. However, these had little impact on the performance. We think it is because the model already converged in our initial parameters.

V. Results

We found that the ridge regression model achieves the best average f1 score from the cross-validation process. Finally, we got an f1 score of 0.415 and an accuracy of 0.866.

VI. Discussion

In the feature engineering process, we also tried to see the 'importance' of the features by plotting their correlation with the prediction. We found that many features like 'IMONTH' have almost zero correlation with the prediction, which is intuitive. However, because of the fear of losing information and also the limit of time, we chose not to discard these features with low correlation. A different strategy could be that we only choose features with high correlation and do a polynomial expansion on them to train, which may get better performance.

For the oversampling technique, we found that although the f1 score was improved after the oversampling, the accuracy was harmed. Also, it got worse performance than our downsampling. A more non-trivial oversampling algorithm may work better.