UNIVERSITÄT KLAGENFURT

Heart Failure Prediction with Feed Forward Neural Network

Processing of missing data by neural networks

Machine Learning and Deep Learning

Author:
Ahmed Kafrana (ID: 12038320)

Date: March 10, 2022

1 Introduction

Cardiovascular diseases (CVDs) are the leading cause of death globally, taking an estimated 17.9 million lives each year. CVDs are a group of disorders of the heart and blood vessels and include coronary heart disease, cerebrovascular disease, rheumatic heart disease and other conditions. More than four out of five CVD deaths are due to heart attacks and strokes, and one third of these deaths occur prematurely in people under 70 years of age. People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidaemia or already established disease) need early detection and management wherein a machine learning model can be of great help for it's very important to have the capability of predicting if a patient could have an heart attack considering his CVDs.
The aim of this project is to find the best supervised model to predict if a patient could have an heart failure or not on the 'heart.csv' dataset found on kaggle at the following link which contains 11 features that can be used to predict a possible heart disease. The most import metrics we look at in order to decide which model perform the best is the recall as in medical issues is much more important to know who can have an heart attack then who will not and the recall returns the ability of a classifier to find all positive instances that, in our case, is the heart failure. In order to do that we use two classifiers fitted on our dataset which are a Support Vector Machine and a Feed Forward Neural Network.

2 Literature overview

Numerous works has been done related to disease prediction systems using different data mining techniques and machine learning algorithms in medical centres. We mainly referred to the paper 'Heart failure prediction using machine learning techniques' from Sahoo et al.[1] in which, as we did, they took the dataset on kaggle, removed unnecessary values and implemented many machine learning techniques to solve our same issues choosing an SVM with radial bases kernel as the best model, reaching an accuracy of 85.2% .
In the section of this paper related to the literature overview, they also showed other techniques used to solve the saim issues. The most interesting for our work are:

- Chang et al. in the paper 'A New Hybrid XGBSVM Model: Application for Hypertensive Heart Disease' [2] proposed a method for the prediction of heart disease which is based on a hybrid neural network that combines two-way long-term memory and selfencoding network. Using the data of 35,332 hypertensive patients, the model was used to predict renal disease in hypertensive patients and the final prediction accuracy was 89.7%.

- Gudadhe et al. in the paper 'Decision support system for heart disease based on support vector machine and Artificial Neural Network' [3] proposed a system by using Support Vector Machine and multilayer perceptron neural network architecture for identifying the heart disease. They divided the database into

two categories by using the Support Vector Machine to show the presence of heart disease or absence of heart disease. They achieved with an accuracy of 80.41%, whereas the artificial neural network classifies the heart disease data into 5 risk classes with an accuracy of 97.5%. In this project we implemented two different techniques that are SVM and Feed Forward Neural Network as they seems to be the ones that performed better in the papers in analysis.

3 Dataset

3.1 Dataset overview

The dataframe shows the record of 918 people with 11 features (independent variables) that are connected to heart disease and 1 target variable named 'HeartDisease' that has 2 values: 1 for people having it and 0 for people who do not. In this dataset, 5 heart datasets are combined over this 11 common features which makes it one of the largest heart disease dataset available so far for research purposes.
In particular we have the following features:

- Age : age of the patient [years]
- Sex : sex of the patient [M: Male, F: Female]
- ChestPainType : chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: NonAnginal Pain, ASY: Asymptomatic]
- RestingBP : resting blood pressure [mm Hg]
- Cholesterol : serum cholesterol [mm/dl]
- FastingBS : fasting blood sugar [1: if FastingBS $<$ 120 mg/dl, 0: otherwise]
- RestingECG : resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of $>$ 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]
- MaxHR : maximum heart rate achieved [Numeric value between 60 and 202]
- ExerciseAngina: exercise-induced angina [Y: Yes, N: No]
- Oldpeak : oldpeak = ST [Numeric value measured in depression]
- ST Slope : the slope of the peak exercise ST segment [Up : upsloping, Flat: flat, Down: downsloping]
- HeartDisease : output class [1: heart disease, 0: Normal]

3.2 EDA and preprocessing

Exploratory Data Analysis on the dataset are made by my team mate, who also imputed missing values using deep learning and removed outliers. Anyway, in order to build classification models, it's important to know that the dataset has no missing value and the target variable is almost well balanced as we have 507 people who had the heart disease, that is the 55%, and 410 who didn't, which is the 45%. For this reason we don't need to use any technique to balance the dataset such as oversampling or undersampling and we don't need to use a cost matrix to balance the loss weights during the training of the models. Afterwards we need to normalize the features of our dataset; this is made with a function that recognise if the feature type is a categorical or numeric variable. If it's numerical, the function will normalize it with a min-max scaler with the following formula:

$$X_n - \frac{X - X_l}{X_h - X_l}$$

where X is each value, X_n is the new value and X_h, X_l are respectively the higher and the lower value in that feature so that every value will stay between 0 and 1. If the variable is categorical we use one hot encoding to separate each category in a single new feauture that can takes values 0 or 1 where 0 indicates non existent while 1 indicates existent. This two operations will help the classifiers to better predict the target variable for now every value in the dataset stays between 0 and 1. Afterwards we split the dataset into train and test sets with percentage of 70% for training and 30% for testing our classifiers.

3.3 Missing Values

There were not any missing values in our dataset. However, we did it as an extra mile. The strategy was to delete some values of the dataset, then try to fill them using a neural network.
We used a mask that goes over the whole dataset and convert 40% of the values into nan values using this method.

```python
def produce_NA(X, p_miss, mecha="MCAR", opt=None, p_obs=None, q=None):
    """Generate missing values for specifics missing-data mechanism and proportion of missing values.
    Parameters:
    X : torch.DoubleTensor or np.ndarray, shape (n, d)
        Data for which missing values will be simulated.
        If a numpy array is provided, it will be converted to a pytorch tensor.
    p_miss : float
        Proportion of missing values to generate for variables which will have missing values.
    mecha : str,
            Indicates the missing-data mechanism to be used. "MCAR" by default, "MAR", "MNAR" or "MNARsmask"
    opt: str,
            For mecha = "MNAR", it indicates how the missing-data mechanism is generated: using a logistic regression ("logistic"), quantile censorship ("quantile") or logistic regression for generating a self-masked MNAR mechanism
    p_obs : float
            If mecha = "MAR", or mecha = "MNAR" with opt = "logistic" or "quanti", proportion of variables with *no* missing values that will be used for the logistic masking model.
    q : float
        If mecha = "MNAR" and opt = "quanti", quantile level at which the cuts should occur.
    Returns
    ----------
    A dictionnary containing:
    'X_init': the initial data matrix.
    'X_incomp': the data with the generated missing values.
    'mask': a matrix indexing the generated missing values.s
    """
    to_torch = torch.is_tensor(X) ## output a pytorch tensor, or a numpy array
    if not to_torch:
        X = X.astype(np.float32)
        X = torch.from_numpy(X)

    if mecha == "MAR":
        mask = MAR_mask(X, p_miss, p_obs).double()
    elif mecha == "MNAR" and opt == "logistic":
        mask = MNAR_mask_logistic(X, p_miss, p_obs).double()
    elif mecha == "MNAR" and opt == "quantile":
        mask = MNAR_mask_quantiles(X, p_miss, q, 1-p_obs).double()
    elif mecha == "MNAR" and opt == "selfmasked":
        mask = MNAR_self_mask_logistic(X, p_miss).double()
    else:
        mask = (torch.rand(X.shape) < p_miss).double()

    X_nas = X.clone()
    X_nas[mask.bool()] = np.nan
    return {'X_init': X.double(), 'X_incomp': X_nas.double(), 'mask': mask}
```

Figure 1: A method for converting 40% of values into nan

After having the data with 40% nan values, we want to fill in these values. The following are the ways to do so:

- Deleting Rows with missing values
- Impute missing values for continuous variable
- Impute missing values for categorical variable
- Using Algorithms that support missing values
- Prediction of missing values
- Imputation using Deep Learning

Here we are using a neural network to fill in the missing data and solve the classification problem using the predicted data. There is a paper that proposes propose a general, theoretically justified mechanism for processing missing data by neural networks. The idea is to replace typical neuron's response in the first hidden layer by its expected value. This approach can be applied for various types of networks at minimal cost in their modification. Moreover, in contrast to recent approaches, it does not require complete data for training. Experimental results performed on different types of architectures show that this method gives better results than typical imputation strategies and other methods dedicated for incomplete data.

4 Processing of missing data by neural networks

4.1 Introduction

In this paper, we introduce a general, theoretically justified methodology for feeding neural networks with missing data. Our idea is to model the uncertainty on missing attributes by probability density functions, which eliminates the need of direct

completion (imputation) by single values. In consequence, every missing data point is identified with parametric density, e.g. GMM, which is trained together with remaining network parameters.
The main advantage of the proposed approach is the ability to train neural network on data sets containing only incomplete samples (without a single fully observable data). This distinguishes this approach from recent models like context encoder, denoising autoencoder or modified generative adversarial network, which require complete data as an output of the network in training. Moreover, this approach can be applied to various types of neural networks what requires only minimal modification in their architectures.

4.2 Network architecture

Adaptation of a given neural network to incomplete data relies on the following steps:

- Estimation of missing data density with the use of mixture of diagonal Gaussians. If data satisfy missing at random assumption (MAR), then we can adapt EM algorithm to estimate incomplete data density with the use of GMM. In more general case, we can let the network to learn optimal parameters of GMM with respect to its cost function.

- Generalization of neuron's response. A missing data point (x, J) is interpreted as the mixture of degenerate Gaussians F_S on S = Aff[x, J]. Thus we need to generalize the activation functions of all neurons in the first hidden layer of the network to process probability measures. In consequence, the response of n(·) on (x, J) is given by n(F_S).

The rest of the architecture does not change, i.e. the modification is only required on the first hidden layer.

4.3 Experiments

The paper used three architectures to evaluate their model. And, we used shallow radial basis function network (RBFN) as it is the most suitable for binary classification.
Radial basis function network: RBFN can be considered as a minimal architecture implementing our model, which contains only one hidden layer. We used cross-entropy function applied on a softmax in the output layer. This network suits well for small low-dimensional data.
The heart failure date was prepared to be fed into this model for evaluation. After applying the mask to get missing values, we exported the dataframe into comma delimited text file. Analogical cross-validation procedure is applied. The number of RBF units was selected in the inner cross-validation from the range 25, 50, 75, 100. Initial centers of RBFNs were randomly selected from training data while variances were samples from N(0, 1).

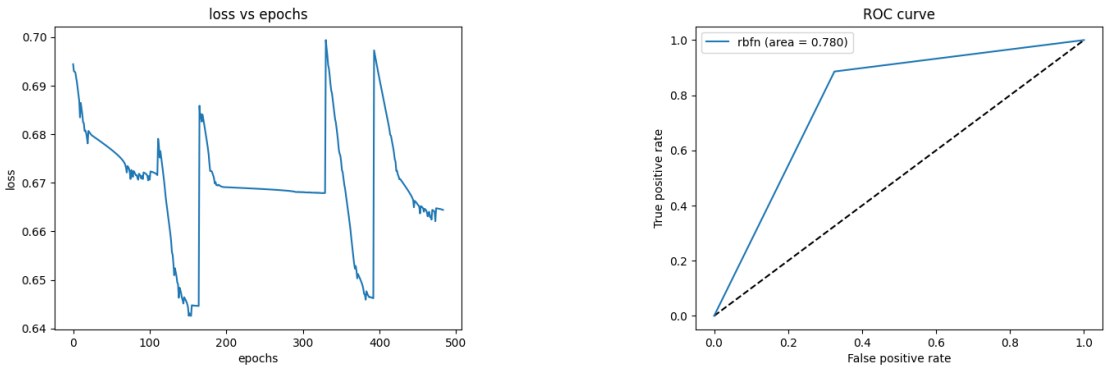
In this model a generalized neuron's activation ReLu function were used at the first layer. We used the sigmoid as activation function in the last layer as it's a binary classification so that we transform the final output of the net in a score between 0 and 1 which is the predicted probability that the current observation has target=1. We used the stochastic gradient descent optimizer (SGD) with a learning rate starts from 0.25 and decays to 0.0001 and 32 for batch size. Then we got 0.8431 for train accuracy, 0.8633 for validation accuracy and 0.8002 for test accuracy.

4.4 Model Evaluation

Our model is evaluated based on accuracy, loss vs epoch graph and ROC curve. ROC curves (receiver operating characteristic curve) are typically used in binary classification to study the output of a classifier. They feature true positive rate on the Y axis, and false positive rate on the X axis. This means that the top left corner of the plot is the "ideal" point - a false positive rate of zero, and a true positive rate of one. This is not very realistic, but it does mean that a larger area under the curve (AUC score) is usually better.

4.4.1 Trial 1

Learning rate 0.025 with 25 of hidden neurons for first layer and batch size 75



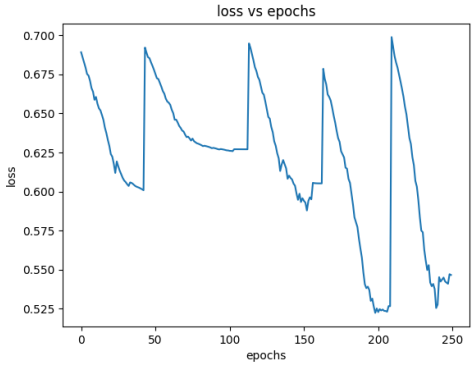
(a) Loss

(b) ROC

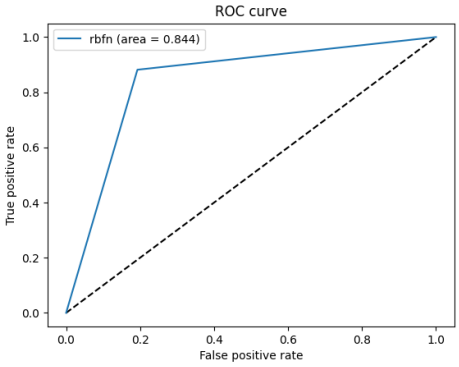
Figure 2: Trial 1

4.4.2 Trial 2

Learning rate 0.25 with 50 of hidden neurons for first layer and batch size 125



(a) Loss

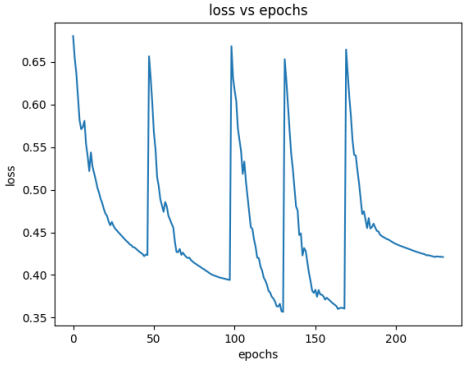


(b) ROC

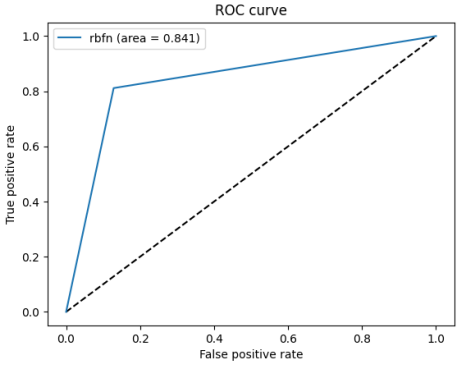
Figure 3: Trial 2

4.4.3 Trial 3

Learning rate 0.25 with 25 of hidden neurons for first layer and batch size 75



(a) Loss

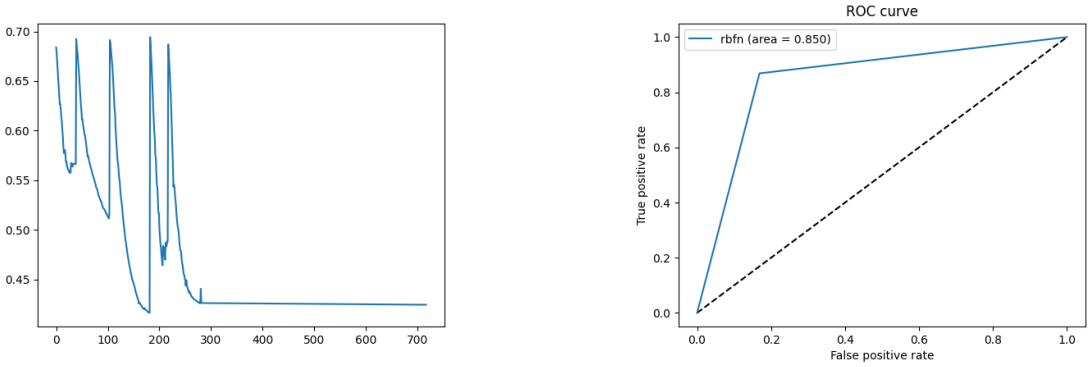


(b) ROC

Figure 4: Trial 3

4.4.4 Trial 4

Learning rate 0.25 with 25 of hidden neurons for first layer and batch size 32



(a) Loss **(b)** ROC

Figure 5: Trial 4

5 Conclusion

All models performe very well with accuracy 0.93 for the SVM with polynomial kernel of degree=3 and also 0.93 for the FFNN and 0.84 for RFBN model. The new model (RFBN) has a bit lower accuracy than the others but keeping in mind that the 40% of the data is missed, we think that it's a good result and the model was relatively able to classify the dataset.

6 References

[1] Prasanta Kumar Sahoo and Pravalika Jeripothula. Heart failure prediction using machine learning techniques. SSRN, December 15, 2020.

[2] Wenbing Chang, Yinglai Liu, Xueyi Wu, Yiyong Xiao, Shenghan Zhou, and Wen Cao. A new hybrid xgbsvm model: Application for hypertensive heart disease. IEEE Access, 7:175248–175258, 2019.

[3] Mrudula Gudadhe, Kapil Wankhade, and Snehlata Dongre. Decision support system for heart disease based on support vector machine and artificial neural network. In 2010 International Conference on Computer and Communication Technology (ICCCT), pages 741–745, 2010.

[4] sklearn. Gridsearch documentation, 2022. Last accessed 22 February 2022.

[5] https://papers.nips.cc/paper/2018/file/411ae1bf081d1674ca6091f8c59a266f-Paper.pdf