Machine Learning Programing Project Report

Background

According to World Health Organization (2021), cardiovascular diseases (CVDs) is one of the leading causes of death worldwide. It is estimated that there are 17.9 million people died from CVDs in 2019, accounting for 32% of global deaths. However, Most of the CVDs can be prevented if they can be detected as earlier as possible for people to adopt health behaviors. A good prediction on detection data can further help doctors to draw conclusions, especially in the current big data era. Thus, to develop a model that can predict CVD death is meaningful and helpful.

Target

Build a model that can predict death event of heart failure caused by cardiovascular diseases (CVDs) via different machine learning methodologies.

Data description

The dataset downloaded from Kaggle contains 299 cases and 13 features. Following by some steps of inspection, it was clear that the dataset contains neither missing values nor duplicated values. Features of age, ejection fraction, serum creatinine and time were most corelated to the feature of death event. As the feature "time" was meaningless in this dataset, it was not considered as an input. Out of 299 cases, 96 cases were death events, accounting for nearly 50%. There were 194 males and 105 females in the dataset, aging from 40 to 95.

Methodologies, experiments, results and error analysis

ML1——According to the dataset, it is clear that this is a binary classification problem with the abovementioned 4 features as inputs and the death event feature as an output. Therefore, I applied for methodologies of Logistic Regression, Decision Tree and SVM in machine learning 1. (done in jupyter notebook)

Logistic Regression. After defining the input and the output for the model, the dataset was split into a train set and a test set. A pipeline was made to build the model. Due to no missing values, the "imputer" step was skipped. However, the ranges of some features' figures were too large which might cause unnecessary overweight. Considering there were no extreme outliers, standardization was adopted to scale the data. The model generated 0.68 accuracy score when testing, predicting 32 out of 42 for not dead cases and 9 out of 18 for dead cases. It didn't seem to be a very good accuracy. Hence, a dummy was applied for comparison. The dummy resulted in 0.54 score which was lower than the model's. Furthermore, the cross-validation score of the model was also calculated, with a 0.68 mean value and a 0.08 standard deviation which was nearly 0 reflecting the mean value to be valuable. Thus, the model was acceptable.

Decision Tree. Considering the number of the cases (299, not too large), the decision tree was made in maximum three layers depth. From the decision tree, we could get the information as follows:

1. There were 299 samples in the dataset, dividing into two classes with the number of 203 and 96. According to previous inspection, it was known that 96 samples were labeled "death" and 203 "not death".
2. The feature of serum creatinine was set in the first node of the decision tree indicating that it contains most of samples.
3. When serum creatinine level was lower than or equal to 1.815, 61 out of 251 samples were in the dead class. The entropy was 0.8 which was lower than the previous one, 0.9 in the first node, further indicating the confirmation of the classification. When serum creatinine level was higher than 1.815 but lower than or equal to 2.05, 35 out of 48 samples were labeled "not death".
4. When ejection fraction level was lower than or equal to 32.5, nearly half of the 71 samples were in the dead class and when serum creatinine level was further lower than or equal to 0.85, then 59 out of 71 were in the not-dead class.

Thus, for those whose ejection fraction level was lower than 32.5, it was better to keep the level of serum creatinine lower than 0.85. Otherwise, it may cause death event (entropy of 0.65).

SVM. After defining the input and the output, a pipeline was set to build the model. Both standardization and normalization were applied to the pipeline but the results showed that normalization was better for the result. Thus, MinMaxScaler was imported and kept in the notebook. Another comparison between LinearSVC and SVC was also implemented. LinearSVC resulted in a mean of 0.68 and a std of 0.065 in cross validation score, while SVC generated 0.71 and 0.045 for mean and std respectively. Therefore, SVC was finally adopted to the model. In order to select a better model, some hyperparameters of the model were inspected and fine tunned:

1. kernel in SVC: by plotting the performance between 3 different kernels and their performances in accuracy, RBF turned out to be the best kernel for the model;
2. gamma in SVC: with kernel RBF, gamma of 10.0 performed the best accuracy;
3. C parameter in SVC, with RBF and 10.0 gamma, selected the best C to be 0.1.

Using the best hyperparameters mentioned above, the accuracy of the model prediction turned out to be 0.7 which was higher than the logistic regression model.

At last, a learning curve was learnt and plotted indicating that it was not necessary to augment the dataset.

ML2——CNN, RNN and LSTM (done in google colab)

The dataset was simply inspected at the beginning of this notebook, as it was already done once in the previous one. Two new important steps were implemented though, namely splitting the dataset into 3 sets (train, validation and test) and changing the digit type of the dataset into float 32.

CNN. When building the model, a lot of problems popped up. One of the most outspoken ones was that the accuracy curves of train and validation remained unchanged along all the epochs when I used softmax as the activation function for the output. After changing it into sigmoid function, because of the binary classification dataset, the results showed that the model actually learnt something. Another prominent problem was that when I tried to calculate the accuracy score, f1 score and classification report, it always returned an error as in "*ValueError: Classification metrics can't handle a mix of continuous and binary targets"*. Considering that the output of the dataset is binary (0 and 1), while the prediction of the model was an array of floats (dtype = float32), I added a threshold to squash the prediction outcome into binary integers and it finally worked. The accuracy of the model was 0.71 which was close to the SVM model result.

RNN. The RNN model got the best result so far with an accuracy score of 0.77 optimized by Adam and limited by early stopping. When applying for SGD as its optimizer, the accuracy curve of the model showed the model learnt basically nothing as the validation accuracy remained horizontal along with epochs. Thus, Adam turned out to be a better choice.

LSTM. Fixing a LSTM model was difficult. At the first beginning, one code line of input layer, one code line of hidden layers and one code line of output layer were created to observe the result. Adam was directly applied for as the optimizer. An accuracy score of 0.73 was obtained. So far, all the neural networks got a better performance outcome than the logistic regression model and even the SVM model. More hidden layers were added in the LSTM model to inspect the difference. But it didn't return a better outcome.

Dataset URL:
https://www.kaggle.com/andrewmvd/heart-failure-clinical-data?select=heart_failure_clinical_records_dataset.csv

Reference

https://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)