Applying Binary Classification Techniques on Prediction of Heart Failure

Lily Qiang

This project focuses on using various binary classification techniques to predict whether a giver person is diagnosed of heart failure given a set of input features (e.g age, sex,chest pain, etc). 4 different types of machine learning models are developed for the binary prediction. Each model has its hyper-parameters being tuned on the training data set using k-fold cross validation technique, and then get trained on the entire training data set again to produce the best parameter. Finally, each model is evaluated on the test data set based on the performance, and the primary metric used for evaluation is classification accuracy. In turns out that the K-Nearest Neighbor has worst performance based on classification accuracy both on the development data set and test data set. On the other hand, logistic regression has best classification performance on both of the two types of data set.

Link to code: https://github.com/Lily-583/cs-467.

Link to code: dataset.

1. Introduction

Cardiovascular disease is one of the leading health killer globally. In 2019, it is found that 17.9 million people died from CVD, 85% among which is due to heart failure. However, the diagnosis of cardiovascular disease remains to be hard, as the triggering factors of this disease are complicated and can be coming from various aspects such as unhealthy food, tobacco, or overweight, etc. As early detection of CVD is critical in saving people's lives, this study is aiming at using various machine learning techniques to build different binary classification models in order to predict whether one is diagnosed of CVD. A value of 0 in the output will indicate one does not have CVD, whereas a value of 1 will indicate one is associated with CVD. Then a confusion matrix will be generated in order to compare and contrast the actual and predicted labels. I will also evaluate the prediction performance of each model by computing some evaluation metrics, such as accuracy. I will also compare those accuracy metrics generated by my models to see which one is the best at prediction. To summarize briefly, the logistic regression yields the best classification performance, whereas k-nearest neighbors yields the worst performance.

2. Related Work

Dissanayake et al (2021) conducted research on data collected from UCI repository's heart disease datasets. After applying feature selection over 303 instances with 75 features, 14 features were selected and used for the experiment purpose. Different supervised learning methods, including random forest, SVM, K-NN, logistic regression and naive Bayes were built and their corresponding performances are evaluated. However, there are several challenges faced by this study. First of all, the size of the dataset is limited, since only 303 instances are used. Therefore, a potential improvement is to use a larger dataset so models with better accuracy can be developed. Also, even though the study used cross-validation technique, it chose a relatively small number of subsets, which may be another reason leading to the loss of prediction accuracy. Also, the output varies greatly depending on the specific feature selection technique used, so another potential improvement for future research is to use hybrid techniques to extract most useful features.

In terms of comparison and contrast between my work and the paper above, the similarities lie in the fact that we both constructed k-nearest neighbors as 1 of the machine learning models, and we both used euclidean distance for measuring distance from neighbors. Also, for naive bayes algorithm, we both used Gaussian distribution for input features. The contrast between my work and the research above is that I primarily focused on tuning hyper-parameters, while the research above spent more time on data pre-processing. This is because we used slightly different dataset. All the examples in my dataset do not have missing features, and that saves me from spending a lot of time on data removal. However, a further improvement I could make on my experiment based on the previous research is to incorporate some feature selection techniques to further improve the classification accuracy.

3. Dataset and Evaluation

3.0.1. Dataset

The dataset used is the "Heart Failure Prediction Dataset" from Kaggle (2021). The input contains 11 independent attributes: (1) Age of patient (numerical); (2) Sex of patient (1 for men and 0 for women); (3) Level of chest pain (will

use integers from 0-3, where 3 indicates the highest level of pain); (4) Rest blood pressure (numerical); (5) Cholesterol level (numerical); (6) Fasting blood sugar level (will use 0 or 1 to show if passing the 120mg/dl threshold); (7) Resting electrocardiogram (will use integers from 0-2 to indicate level of severity, with 2 represents the highest); (8) Max heart beat rate (numerical); (9) Exercise-induced angina (0 for yes and 1 for no); (10) Oldpeak (numerical value to show level of depression); (11) ST-Slope (slope of peak exercise ST segment, will use integer from 1-3 to represent each type). The data set also contains the labels (in binary form, 0 and 1) to indicate if that individual has CVD or not. The outputs produced by my models will be in the form of binary (either 0 or 1) to indicate if each individual is diagnosed of heart disease or not. Then a confusion matrix will be generated to compare the predicted result with the actual labels.

3.0.2. Evaluation

As of evaluation, there are 918 observations totally in my dataset. Among those 918 observations, 508 are classified as having heart failures, and the remaining 410 are people with no heart failures. Since I have a limited amount of data, I decide to use cross validation for the training and development dataset together. To begin with, I firstly selected 20% percent data for the test set (184 data), and for the remaining 734 data, I used cross validation to tune the model and select the best hyperparameters. I will use k-1 folds for training set, and 1 remaining fold for development set. As for the evaluation metrics, I will use accuracy as the main metric for evaluation purposes, since I have a relatively balanced dataset and all classes are of equal importance.

4. Methods

4.0.1. Baseline

I used logistic regression with L2 regularization as the baseline. The inputs are (1) Age of patient (numerical); (2) Sex of patient (1 for men and 0 for women); (3) Level of chest pain (will use integers from 0-3, where 3 indicates the highest level of pain); (4) Rest blood pressure (numerical); (5) Cholesterol level (numerical); (6) Fasting blood sugar level (will use 0 or 1 to show if passing the 120mg/dl threshold); (7) Resting electrocardiogram (will use integers from 0-2 to indicate level of severity, with 2 represents the highest); (8) Max heart beat rate (numerical); (9) Exercise-induced angina (0 for yes and 1 for no); (10) Oldpeak (numerical); (11) ST-Slope (slope of peak exercise ST segment, will use integer from 1-3 to represent each type). After running the logistic regression algorithm on the testing data set, a binary output (in 1 or 0) will be generated to indicate the predicted outcome, that is, whether the person is predicted to have heart disease or not. This predicted outcome will then be compared with the actual outcome for evaluating the accuracy of the model. I used cross-validation on the training and development data set all together. Since I have a limited number of data, I used a relatively large number of iterations (iteration=1000) to run the logistic regression. The hyper-parameters here are the regularization(penalty) and fit intercept. The regularization term, or penalty, is either L1 or L2, which are used to improve the performance and accuracy of logistic regression model. The fit_intercept is a boolean variable that specifies if a constant should be added to the decision function. I used 12 fold cross-validation on the training set to select the best combination of hyper-parameters using grid search. I used classification accuracy as the criteria for selecting the best hyper-parameters. After tuning the best sets of hyper-parameters, I trained the model on the entire training set, and finally evaluated the model's performance on test data set. I used classification accuracy as my main evaluation metrics.

4.0.2. K-Nearest Neighbors

I used the k-nearest neighbor as 1 of my machine learning model for doing the heart failure classification. The input features of the data set are exactly the same as those used in the baseline method above. Since I have limited data, I again used cross validation set to tune the hyper-parameter and used the test data set to evaluate the final performance of the model. I used 12 as the number of folds. The hyper-parameter I experimented on is the number of neighbors. I tested n from 1 to 31 as the number of neighbors. For each number of n, I run the cross validation algorithm, again with 11 folds as training set and 1 fold as development set. For the k nearest neighbor algorithm, for each test data point, the k nearest neighbor are selected using euclidean distance, and finally the test data point is classified as the same class of its majority neighbors. Again I used accuracy as the evaluation metrics for selecting the best hyper-parameter. Finally the model tuned from train+development set is applied on the test set, and the classification accuracy on the test data set is also calculated for performance evaluation.

The algorithm for running k-nearest neighbors is as follows:

1. Start with the data point that needs to be classified. Calculate the euclidean distance between the data point and all the other n existing data points. The euclidean distance between 2 points p,q is defined as below.

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

2. Sort the n calculated distance from greatest to smallest.

3. Take the first k distance from the sorted list.

4. Find the majority label of the example corresponding to the k distance, and assign the current data point to that label.

4.0.3. DECISION TREE

I used decision tree as another machine learning model for doing the heart failure classification. The input features of the data set are same as those used in all the other machine learning algorithms. I again used k-fold cross validation to tune the hyper-parameters first, and then used the test set to measure the ultimate performance of the model. The number of folds I used is always 12, which is consistent across all the different machine learning models I used. I chose 2 hyper-parameters to tune, which are "min sample leaf" and "criterion". The options for "min sample leaf" is [1,5], and the options for "criterion" are "gini" and "entropy". I used "accuracy" as my scoring criteria for selecting the best hyper-parameters. After determined the best set of hyper-parameters using "GridSearchCV", which is a built-in function that tries all the combinations of hyper-parameters on the cross validation set, I trained the model on the entire training data, and evaluated its performance on the test data.

- "Min sample leaf" is defined as "the minimum number of samples required to be at a leaf node".
- The "gini" criteria is used to measure the probability of a specific variable being wrongly classified when it is randomly chosen. p is the empirical probability of class c within the current node. While building the decision tree, the feature with the lowest gini index will be preferred at the current node. Below is the formula for calculating gini index.

$$gini = 1 - \sum_{c=1}^{C} p_c(1 - p_c)$$

- The "entropy" criteria measures the impurity in a group of examples. It is another way to determine which feature should be based on to split the data at the current node. Below is the formula for calculating entropy. After knowing how to calculate entropy, we could in turn calculate the information gain.

$$entropy = -\sum_{c=1}^{C} p_c(log_2(p_c))$$

- The "information gain (IG)" is a metric associated with entropy. The feature with the highest information gain is preferred.

$$IG = entropy(parent) - avg\_entropy(children)$$

The algorithm for constructing the decision tree is as follows:

1. Start with the root node. All training data is under the root.
2. For each feature, split the training data by the value of that feature. Then compute the gini index/information gain of that feature.
3. Select the feature that has the highest information gain/lowest gini index at that node.
4. Partition the data into child nodes based on that feature.
5. For each child node, if it is "pure"(all data under it are from the same class), label it as a leaf node. Otherwise, repeat the above procedure recursively.

4.0.4. NAIVE BAYES

I used Naive Bayes as the last classifier for heart disease classification. Again, the input features of the data set are same as those used in all the other machine learning algorithms. I again used 12-fold cross validation to tune the hyper-parameters. For Naive Bayes, there is only 1 hyper-parameter I tune, which is "var_smoothing". The "var_smoothing" is a user-defined value that is added to a given feature's distribution's variance. This widens the Gaussian distribution curve, and accounts for more samples that are away from the sample mean. I again used "accuracy" as my selection criteria when choosing the best value of the hyper-parameter. The values I chose from are 9 discrete values ranging from 1e-10 to 0.01. After figuring out the best value of the hyper-parameter, I run the algorithm on the test set, and evaluated the performance of the model based on classification accuracy. Even though the ski-learn has built-in function for constructing the Naive Bayes classifier, below is a list of steps of the algorithm.

1. prior: Since the training examples are binary classified, there are only 2 types of prior: p(y=1) and p(y=0), where y=1 denotes training examples labeled as having heart failure, and y=0 denotes those labeled as not having heart failures.

$$p(y = 1) = \frac{p(y = 1)}{p(y = 1) + p(y = 0)}$$

$$p(y = 0) = \frac{p(y = 0)}{p(y = 1) + p(y = 0)}$$

2. conditional probability of features: As we assume the presence of a particular feature in class in independent of the presence of another feature, we can make following derivations:

$$p(x_i|y, x1, ..., x_i, x_i, ..., x_n) = \frac{p(x_i|y)}{y}$$

Therefore, we have

$$p(x_1, x_2, ..., x_n|y) = p(x_1|y)p(x_2|y)...p(x_n|y)$$

And finally, we could compute

$$p(y|x_1, x_2, ..., x_n) = \frac{p(y)\prod_{i=1}^{n} p(x_i|y)}{p(x_1, ..., x_n)}$$

4.0.5. Improvements from Midterm Report

In midterm report, I only constructed 2 machine learning models, namely logistic regression and K nearest neighbors. In final report I further improved the classification accuracy of logistic regression by re-selecting the hyper-parameter for tuning. Previously I chose the number of folds in cross validation as a hyper-parameter to tune, but it is not the best one to play with, and it also makes the training process being inconsistent from other models. Therefore I chose the regularization strength as the hyper-parameter to tune, and the classification accuracy is increased. In addition to that, in this final report I built another 2 models, decision tree and naive bayes, and both of them have more than 80% of classification accuracy on the test dataset.

5. Experiments

5.0.1. Logistic Regression

The hyper-parameter is the type of regularization (l1 or l2) and fit_intercept (whether a constant will be added to the decision function). For each different combinations of the 2 hyper-parameters, the classification accuracy is calculated on the dev set to determine the best combinations of hyper-parameters. Since I used 12-fold cross validation, the actual classification accuracy is the average classification accuracy across the 12 dev sets. From the graph, it turns out that fit_intercept=False and penalty=l1 produces the best classification accuracy on the dev set, which is 87.04%. The classification accuracy of the same model on the test data set is 85.32%.

TP	FP	TN	FN
69	8	88	19

Table 1. Confusion matrix for logistic regression on test dataset.

5.0.2. K-Nearest Neighbors

The hyper-parameter is the number of the nearest neighbors whose labels will be considered when assigning label to the current data point. For each of the 2 hyper-parameter, the classification accuracy is calculated on the dev set to determine the best one. Since I used 12-fold cross validation, the actual classification accuracy is the average classification accuracy across the 12 dev sets. From the graph, it turns out that k=7 produces the best classification accuracy on the dev set, which is 74.11%. The classification accuracy of the same model on the test data set is 64.13%.

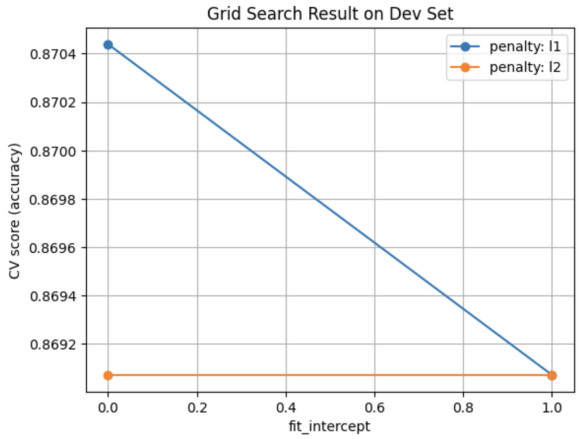

Figure 1. The relationship between the classification accuracy on dev set and different combinations of fit intercept and regularization of logistic regression

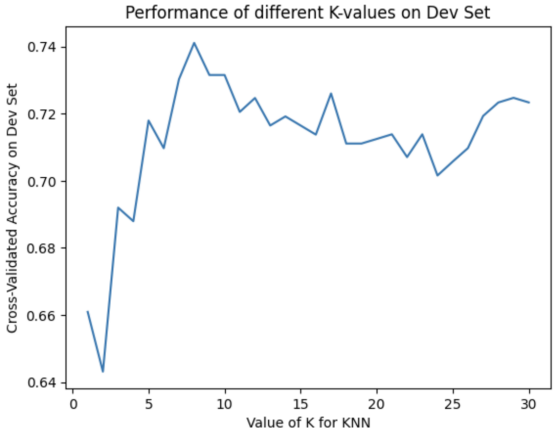

Figure 2. The relationship between the number of neighbors used and the cross validation accuracy on the dev set of K-nearest neighbors

TP	FP	TN	FN
55	22	63	44

Table 2. Confusion matrix for k-nearest neighbors on test data set.

5.0.3. Decision Tree

The hyper-parameter is the minimum number of samples at leaf nodes and criterion (gini index and entropy). For each different combinations of the 2 hyper-parameters, the

classification accuracy is calculated on the dev set to determine the best combinations of hyper-parameters. Since I used 12-fold cross validation, the actual classification accuracy is the average classification accuracy across the 12 dev sets. From the graph, it turns out that criterion=entropy and min_sample_leaf=3 produces the best classification accuracy on the dev set, which is 81.20%. The classification accuracy of the same model on the test data set is 83.15%.

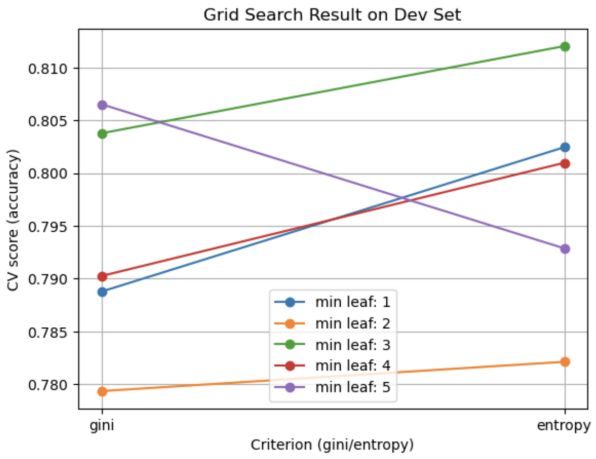

Figure 3. The relationship between the classification accuracy on dev set and different combinations of criterion and number of min leaf of decision tree.

TP	FP	TN	FN
68	9	85	22

Table 3. Confusion matrix for decision tree on test data set.

5.0.4. Naive Bayes

The hyper-parameter is the value of var_smoothing (a value added to the given feature's distribution variance). For each value of the hyper-parameter, the classification accuracy is calculated on the dev set to determine the best one. Since I used 12-fold cross validation, the actual classification accuracy is the average classification accuracy across the 12 dev sets. From the graph, it turns out that var_smoothing=1e-06 produces the best classification accuracy on the dev set, which is 86.36%. The classification accuracy of the same model on the test data set is 84.78%.

TP	FP	TN	FN
67	10	89	18

Table 4. Confusion matrix for Naive Bayes on test data set.

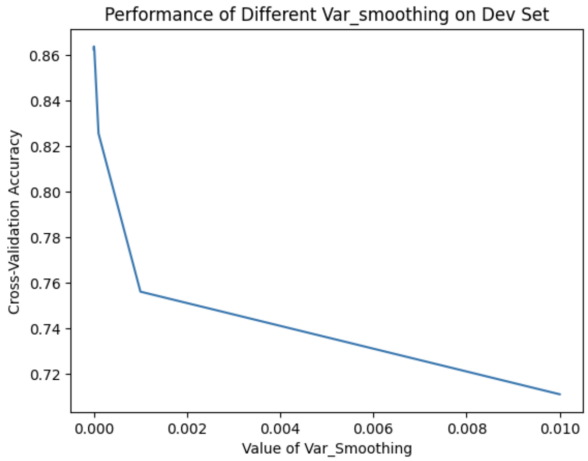

Figure 4. The relationship between the classification accuracy on dev set and different values of var_smoothing on naive bayes.

5.0.5. Performance Comparison

After comparing the classification accuracy of the development set and test set for both models, it turns out that the logistic regression model always has a better classification performance in terms of both classification accuracy on development set and test set. The K-nearest neighbor model has the worst performance on both development set and test set. Naive bayes and decision tree models both have relatively good classification performance on the test data set.

6. Discussion

6.0.1. Error Analysis

It turns out that most of the classification errors for the 4 machine learning models occur when the "oldpeak" (numerical value measured for depression) has a small positive value between 0-3, and the "fasting blood sugar" has a value of 0 (meaning the fasting blood sugar is less than 120mg/ml), and when the resting electrocardiogram (RestingECG) has a value of 0 (meaning it's normal). There could be several explanations for this. First, some features (such as the level of depression) may be thought to be leading factors for heart failure, but in reality, the effects those features exert on the output are not as significant as they are thought to be. Another explanation is that some other features, such as the RestingECG, has even amounts of different values spread across the dataset, and also have no direct bearing on the outcome (the heart failure). That is, there are an equal number of people with and without heart failure having normal RestingECG. Next is the specific error analysis on each model. As for logistic regression, I only used the algorithm 'liblinear', which is the library for large linear

classification. However, other types of algorithm, such as 'newton-cholesky', may produce better result on the data set. As for k-nearest neighbors, the primary source of error is the high dimension of features, and the limited size of training data. Both of those 2 factors lead to the distribution of neighbors being 'sparse', which result in difficulties for finding closest neighbors. As for decision tree, the error in classification may be due to the fact that the data is noisy. In other words, the training process may build up a tree of large size in order to adapt to the noise in the data. However, that leads to over-fitting and poor performance on testing data set. As for naive bayes, even though I made the assumption that the features are conditionally independent from each other, in reality it may not always be the case. That could lead to classification error as well.

6.0.2. Potential Improvements

As for the logistic regression model, one way I could think of to improve the model's accuracy is to normalize the input features to the same scale, so as to prevent some features having overly-dominating effects on the output than others. Another way to increase the accuracy is to try different type of algorithm, such as 'newton-cholesky' or 'saga'. As for the k-nearest neighbors model, one way I could think of to improve the classification accuracy is to prune features that have minimum effect on the classification outcomes. This will make neighbors being close to each other. As for decision tree, a potential improvement is to add further restriction to the size of the tree, so over-fitting would be less likely to happen. For naive bayes, a potential improvement is to do some data pre-processing before hand. That is, to identify which features are more likely to be related to each other, and remove some of them accordingly.

7. Conclusion

In general, after the process of training and tuning hyper-parameters on the training data set, the 4 models have the majority of predictions of heart failures correct on the test data set. It turns out that in order to further improve the classification accuracy, different models have different requirements on the data set. For example, K-nearest neighbors expect the dimension of input features to be low, and naive bayes assume the input features are (conditionally) independent from each other. As for the implementation process, it turns out that the process of determining which hyper-parameters to tune for each model is challenging, and oftentimes there are more than 1 hyper-parameter to tune. I figured out that grid search is a helpful algorithm that enables multiple hyper-parameters to be tested simultaneously. However, the limitation of grid search is that if there are a wide range of values for each hyper-parameter to be chose from, it is time consuming. Something I observe unexpected is that for all the 4 models I built, the false negative rate is always higher than the false positive rate, which suggests that all models have tendencies to classify an example as 'having heart failure' than not. Also, logistic regression has the best classification accuracy on test data set, and k-nearest neighbors has the worst. In terms of runtime, logistic regression, naive bayes and k-nearest neighbors can have training and testing be done in time that is linear in size of the training data set. The runtime of decision tree is dependent on whether the tree to be constructed is balanced or not.

References

Dissanayake, K. and Johar, M. G. M. *Comparative study of heart disease classification*. PhD thesis, School of Graduate Studies, Management Science University, School of Computing, Pioneer Institute of Business & Technology, 2021.

Kaggle. Heart failure prediction dataset, 2021.