Comparison of Classifiers for Predicting Heart Attack in Patients*

Oliwia Cimała[1,*,†], Maria Bocheńska[1,†]

[1]Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44100 Gliwice, POLAND

Abstract

Heart attack predictions play a pivotal role in patients health. While having two options of fast responding to health issue, making many tests on patients to see whats wrong or compare information about the patients with others to classify a patient and narrow down the search to the right field.

This study presents a comprehensive comparison of three classification algorithms — Soft Set Classifier, Naive Bayes, and K-Nearest Neighbors (KNN) — for predicting heart attack in patients. Through experimentation with different variations of these algorithms, including custom implementations, the project evaluates their effectiveness in recognizing high or low chance of heart attack. Methodologically, the project explores the nuances of each algorithm, discussing their underlying principles and implementation details. Experimental results reveal insights into the performance of each algorithm, providing valuable considerations for practical applications. Additionally, the project discusses the significance of precision, recall, F1-score, and accuracy metrics in assessing algorithm performance. Overall this study contributes to advancing heart attack prediction technology, offering valuable insights into algorithmic approaches.

Keywords

Soft Set Classifier, Naive Bayes, K-Nearest Neighbors, Heart Attack Prediction, Machine Learning

1. Introduction

The heart is vital to the body's function, acting as a powerful pump that circulates blood, oxygen, and essential nutrients throughout the body. This cardiovascular system ensures that all bodily tissues receive the resources they need to operate effectively. Consequently, any issues with the heart can disrupt the normal functioning of other organs and systems, leading to widespread health problems [1]. Heart disease are the main responsible for one-third of all human deaths in the world [2], making accurate and timely diagnosis critical for effective treatment. Traditional diagnostic methods often rely on various tests and clinical evaluations, which can be time-consuming and costly. With the advancement of machine learning, there is an increasing interest in developing automated systems for predicting heart disease using patient data [3, 4, 5].

*IVUS2024: Information Society and University Studies 2024, May 17, Kaunas, Lithuania

[1,*] Corresponding author

[†] These author contributed equally.

oc307854@student.polsl.pl (O. Cimała); mb307847@student.polsl.pl (M. Bocheńska)

0009-0002-1923-0781 (O. Cimała); 0009-0001-3285-9229 (M. Bocheńska)

Existing solutions leverage different algorithms to achieve this goal, including logistic regression, decision tree, random forest, voting and neural networks [6]. However, our study focuses on comparing three distinct classifiers: the Soft Set Classifier [7], Naive Bayes [8], and K-Nearest Neighbors (KNN) [9]. Each of these algorithms offers unique advantages and challenges, which we explore in the context of heart disease prediction.

To get a closer look into the applied classifiers, the following paragraphs will briefly describe them to illustrate the differences between these calculation methods.

The Soft Set classifier is a flexible and general mathematical tool used for handling uncertainty in data. It does not rely on predefined probabilities or distances, making it particularly useful in situations where traditional probabilistic or distance-based models like Naive Bayes or K-Nearest Neighbors (KNN) may not perform well. The classifier iteratively adjusts the membership values based on the training data, thus enabling it to handle imprecise and vague information effectively. The model's adaptability to various forms of uncertainty makes it a valuable tool in fields where data ambiguity is prevalent.

The Naive Bayes classifier is a probabilistic machine learning model based on Bayes' theorem, which calculates the probability of a certain class given a set of features. It assumes that the features are conditionally independent, hence "naive."

K-Nearest Neighbors (KNN) is a non-parametric supervised learning algorithm used for classification and regression tasks. In KNN, the class of a new data point is determined by the majority class among its k nearest neighbors in the feature space. It's simple to implement and understand but can be computationally expensive for large datasets, as it requires storing all training data and computing distances for each prediction.

All three algorithms have varying time consumption, with K-Nearest Neighbors (KNN) being more computationally expensive due to its need to calculate distances for each prediction. While making the algorithms we follow the same build of the specific class. The class contains two functions the fit and predict, if needed also other functions like: distance or score of the given sample. Now, let's delve into a brief explanation of each of the applied algorithms and the underlying thought process behind their selection. The first classifier is the Soft Set classifier that is independently create. Next, the Naive Bayes classifier is from the library, change a little to be built like a rest (it also have a fit, predict functions in Bayes class). The third classifier is a K-Nearest Neighbours algorithm but in this instance written by us. It was created following open-access models with an interest to achieve as high accuracy as possible. After performing the calculations, each algorithm displays a matrix and a table with the results of the effectiveness in defining of low or high probability of heart attack.

2. Methodology

This section details the methodologies used for each classifier, including their mathematical foundations and implementation specifics.

2.1. SoP Set Classifier

The Soft Set Classifier, from a mathematical perspective, assigns to each element of the set X a value from the interval $<-1, 1>$, representing the degree of membership of that element to the

set X. A membership value of 1 indicates assignment to the negative class, while a membership value of -1 indicates assignment to the positive class.

Algorithm 1: Soft Set Classifier

Input: Training set X_{train} , Training labels y_{train} , Number of iterations n_{iters} , Regularization parameter λ_{param}

Output: Fitted model Y

Initialize weight vector Y to zeros of length equal to the number of features;
for *iteration in range* n_{iters} **do**
  for *each sample* x_i, y_i *in* X_{train}, y_{train} **do**
    if y_i * *classify*(x_i) ≤ *1* **then**
      Update Y by $Y \leftarrow Y + y_i * x_i - 2 * \lambda_{\text{param}} * Y$
Return Fitted weight vector Y

Algorithm 2: Soft Set Prediction

Input: Test set X_{test}, Fitted weight vector Y

Output: Predicted labels y_{pred}

for *each sample* x_i *in* X_{test} **do**
  Compute classification score classification ← classify(x_i));
  Assign label $y_{\text{pred}} \leftarrow$ sign (classification);
return *Predicted labels* y_{pred}

2.2. Naive Bayes Classifier

The Naive Bayes classifier is based on Bayes' theorem and assumes that the features are conditionally independent given the class label. The implementation follows these steps:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \tag{1}$$

where $P(y|X)$ is the posterior probability of class y given feature vector X

Algorithm 3: Naive Bayes

Input: Training set X_{train}, Training labels y_{train}, Test set X_{test}

Output: Predicted labels y_{pred}

Step 1: Initialize the Gaussian Naive Bayes model;
Step 2: Fit the model with the training data X_{train} and y_{train};
Step 3: Predict the labels for X_{test} using the trained model;

2.3. K-Nearest Neighbors (KNN) Classifier

The KNN classifier classifies a sample based on the majority label among its k-nearest neighbors in the training set. The distance metric used is typically the Euclidean distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (2)$$

Algorithm 4: KNN Algorithm

Input: Training set X_{train}, Training labels y_{train}, Test set X_{test}, Number of neighbors k
Output: Predicted labels y_{pred}

```
for each sample x in X_test do
    Compute distances between x and all samples in X_train;
    Identify the k-nearest neighbors;
    Assign the label based on the majority vote of the neighbors;
```

3. Experiments

3.1. Dataset Description

The dataset includes records of patients along with their medical attributes and the presence or absence of heart disease. The dataset contains 13 columns with different attributes: age, sex, number of major vessels, chest pain type, resting blood pleasure, cholesterol, maximum heart rate achieved, fasting blood sugar, resting electrocardiograph results, exercises, slope, thal rate and the last column that we compare to (target variable).
All records were first normalized and then subjected to further tests. The normalization function operated on the basic min-max algorithm [10].

3.2. Data Splitting and Testing

To evaluate the performance of our classifiers, we split the dataset into a training set and a test set. This is a crucial step to ensure that the model can generalize well to unseen data. We used the 'train-test-split' function from the 'sklearn.model-selection' library for this purpose.

```
X_train, X_test, y_train, y_test = train_test_split(X
    , y,test_size=0.35, random_state=42)
```

This function performs the following tasks:

- Input Parameters:
  - X: the feature matrix containing the input data for all samples.
  - y: the target vector containing the labels for all samples.
  - test_size=0.35: specifies the proportion of the dataset to include in the test split. (Here, 35% of the data is allocated for testing, and the remaining 65% is used for training.)
  - random_state=42: this parameter ensures reproducibility of the results. By setting a specific random state, we ensure that the same split is generated every time the code is run.

- Outputs:
  - `X_train`: the feature matrix for the training set.
  - `X_test`: the feature matrix for the test set.
  - `y_train`: the target vector for the training set.
  - `y_test`: the target vector for the test set.

By splitting the data into training and testing sets, we can train the model on one subset of the data and evaluate its performance on another, independent subset. This approach helps in assessing how well the model can generalize to new, unseen data and is an essential part of model validation in machine learning.

3.3. Results Analysis

To compare the different performance parameters of the used algorithms, we utilized the metrics module from the 'sklearn' library. The dataset containing numerical values in 13 different types of attributes (medical data of the patient) with a total length of 303 records was divided into training and testing sets in a 65:35 ratio. For each algorithm, we compared parameters such as:

- precision - it is a measure that determines the ratio of correctly predicted class elements to all those marked as the given class

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

- recall - a measure that informs us how many elements from given class were correctly recognized

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4}$$

- f1-score - it is the harmonic mean between precision and recall

$$\text{F1-score} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5}$$

- support - a measure of the occurrences of each class in dataset
- accuracy - it is the ratio of correctly classified samples to all cases in the test set

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP} \tag{6}$$

Meaning of labels:

- TP - true positive - cases that were correctly classified as positive by the classifier
- TN - true negative - cases that were correctly classified as negative by the classifier
- FP - false positive - an error where the test result incorrectly indicates the presence of a condition when it is not present
- FN - false negative - an error where the test result incorrectly indicates the absence of a condition when it is actually present

3.4. Results

As we can see in the results above in matrix we have 0 and 1 (Fig. 1) as the output were 0 is a low chance of heart attack and 1 is a higher chance of heart attack. And in the classification-report, that is from 'sklearn' library, the 0 value is change to -1 (Tab.: 1, 2, 3).
Analyzing the results shown in above matrix and table, we can observe that all three algorithm have lower precision in qualify the low chance of heart attack.

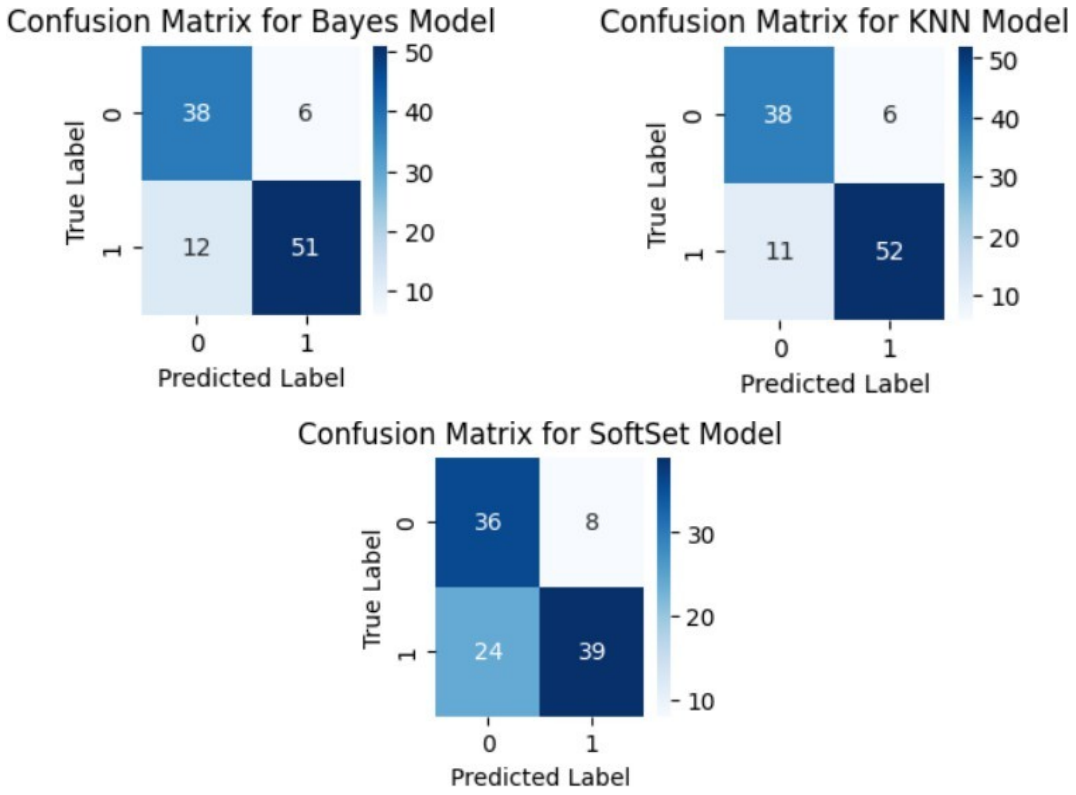

Figure 1: Comparison of Different Classifiers

As observed, the Soft Set algorithm struggles the most (have the lowest accuracy 70% (see Tab. 3)). With only 1% advantage at accuracy the K-Nearest Neighbors performs better then the Naive Bayes algorithm whose accuracy is at 83% (see Tab. 2).

Table 1
Accuracy when model is trained with KNN: 84.11214953271028

Class	Precision	Recall	F1-score	Support
-1.0	0.78	0.86	0.82	44
1.0	0.90	0.83	0.86	63
Accuracy			0.84	107
Macro avg	0.84	0.84	0.84	107
Weighted avg	0.85	0.84	0.84	107

Table 2
Accuracy when model is trained with Bayes: 83.17757009345794

Class	Precision	Recall	F1-score	Support
-1.0	0.76	0.86	0.81	44
1.0	0.89	0.81	0.85	63
Accuracy			0.83	107
Macro avg	0.83	0.84	0.83	107
Weighted avg	0.84	0.83	0.83	107

Table 3
Accuracy when model is trained with Soft Set: 70.09345794392523

Class	Precision	Recall	F1-score	Support
-1.0	0.60	0.82	0.69	44
1.0	0.83	0.62	0.71	63
Accuracy			0.70	107
Macro avg	0.71	0.72	0.70	107
Weighted avg	0.74	0.70	0.70	107

4. Conclusion

This study presented a comparative analysis of three different classifiers for heart disease prediction. The Soft Set Classifier, while effective in handling uncertainty, showed moderate accuracy which equals 70%. The Naive Bayes classifier demonstrated high accuracy 83%, making it a strong candidate for medical diagnostics. The K-Nearest Neighbors classifier also performed well, with an accuracy of 84%. These results provide valuable insights into the strengths and limitations of each classifier, guiding future research and application in medical diagnostics. In all this pondering we need to remember that the Naive Bayes classifier wasn't written by us. We can only assume what kind of results can give independently written the Naive Bayes algorithm and what results can bring us the K-Nearest Neighbors and Soft Set classifier written from the library.
Improvements that we can make in the future are to write the Naive Bayes algorithm and check its accuracy then, remake the Soft Set algorithm so it reaches higher accuracy. In addition to

boost the accuracy we can compare all of the three algorithms to the ones from library and eliminate the weak points because of which the accuracy isn't as high as needed.

References

[1] H. Arghandabi, P. Shams, A comparative study of machine learning algorithms for the prediction of heart disease, International Journal for Research in Applied Science and Engineering Technology 8 (2020) 677–683. doi:10.22214/ijraset.2020.32591.

[2] K. Uyar, A. Ilhan, Diagnosis of heart disease using genetic algorithm based trained recurrent fuzzy neural networks, Procedia Computer Science 120 (2017) 588–593. doi:10.1016/j.procs.2017.11.283.

[3] I. Rojek, P. Kotlarz, M. Kozielski, M. Jagodziński, Z. Królikowski, Development of ai-based prediction of heart attack risk as an element of preventive medicine, Electronics 13 (2024). doi:10.3390/electronics13020272.

[4] R. J. A. Laxamana, J. M. M. Vale, Heart attack prediction using machine learning algorithms, Journal of Electrical Systems 20 (2024) 1428–1436. doi:10.52783/jes.2474, license CC BY-ND 4.0.

[5] S. K. Gupta, A. Shrivastava, S. P. Upadhyay, P. Chaurasia, A machine learning approach for heart attack prediction, International Journal of Engineering and Advanced Technology 10 (2021) 124–134. doi:10.35940/ijeat.F3043.0810621, mahatma Gandhi Central University Bihar, Babasaheb Bhimrao Ambedkar Central University Lucknow.

[6] K. Oliullah, A. Barros, M. Whaiduzzaman, Analyzing the Effectiveness of Several Machine Learning Methods for Heart Attack Prediction, 2023, pp. 225–236. doi:10.1007/978-981-19-9483-8_19.

[7] P. Majeed, H. A. Shareef, H. M. Darwesh, Three classes of soft functions via soft-open sets and soft-closed sets, Wasit Journal of Pure Sciences 3 (2024) 1–17. doi:10.31185/wjps.288.

[8] P. Langley, W. Iba, K. Thompson, et al., An analysis of bayesian classifiers 90 (1992) 223–228.

[9] K. Prokop, Grey wolf optimizer combined with k-nn algorithm for clustering problem, in: IVUS 2022: 27th International Conference on Information Technology, 2022.

[10] M. Shantal, Z. Othman, A novel approach for data feature weighting using correla- tion coefficients and min–max normalization, Symmetry 15 (2023) 2185. doi:10.3390/sym15122185.