Detecting Heart Attacks from Patient Data

Nathaniel Netznik, Huy Ngo, Zheyu Zhang

CMPSC 445 - Applied Machine Learning in Data Science

Abstract

In recent years, machine learning models have led to breakthroughs in the medical field. These developments have allowed for the detection and prediction of health conditions with notable accuracy. We deployed three well-known machine learning models - k-nearest neighbors, naive Bayes, and decision trees - to detect heart attacks in patients. We found that the naive Bayes model was most effective at the task, followed by k-nearest neighbors and decision trees respectively.

1 Introduction

Illness detection is critical for providing effective care to patients reporting discomfort. Traditionally, this task has been addressed by empirical human judgment. With the emergence of computing over the past few decades, medical professionals have access to a new arsenal of systematic, automated, and data-driven methods for solving such problems. In this study we are particularly interested in detecting heart attacks in patients given biomedical data such as their age, gender, type of chest pain, and various other potentially relevant factors. We will apply three basic machine learning algorithms to address this problem. This study could assist medical practitioners in diagnosing and treating patients appropriately.

There have been previous studies that use machine learning to detect heart attacks. Karthikeyan et al. (2020) used XGBoost, an extreme gradient boosting method, obtaining an accuracy of 90.46%. The model has many parameters, requiring a large amount of tuning. Chitra and Seenivasagam (2013) consider the use of neural networks for detecting heart attacks, finding that using the Hybrid Intelligent Algorithm improves its accuracy. However, they comment that a neural networks approach can be time consuming; one must balance dimensionality reduction with accuracy. Ranga and Rohila (2018) deployed a variety of classifiers including k-nearest neighbors, artificial neural network, decision tree, random forest, and support vector machine (SVM), with each resulting in an accuracy of approximately 80%. They found that the random forest method was most effective, but believe that their results might have been more accurate if they had more instances. Ware et al. (2020) conducted a similar

study by testing six classifiers; they found that SVMs were the most effective. Like Ranga and Rohila, they noted that a larger dataset might result in higher accuracy, as well as more features. Gawale and Gawande (2020) compared the results between decision trees and SVMs, confirming that SVMs were more effective. However, they did not compare any other models.

This previous work has explored the use of a variety of machine learning models in heart attack detection. In our work, we will particularly bring together a few of the simplest, most well-known machine learning algorithms to demonstrate and compare their effectiveness with respect to each other at the task. We will also note the effects of tweaking hyperparameters in the models we use.

1.1 Our Contribution

Our contributions in this project include:

- Detecting heart attacks given patient data
- Exploring how simple machine learning algorithms can be applied in addressing this task
- Identifying which of these algorithms are most effective at solving this task

2 Material and Methods

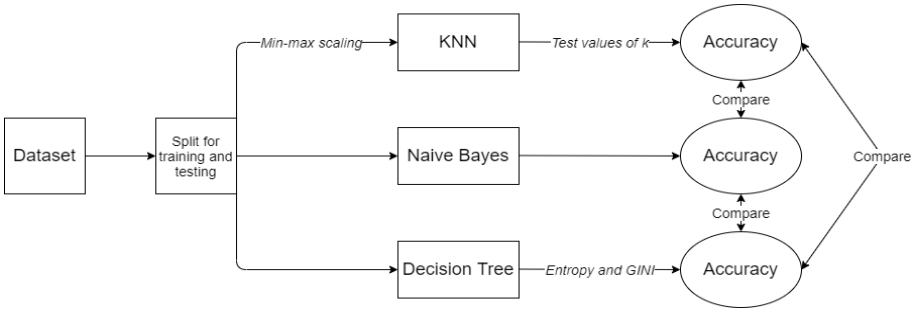

Figure 1: A diagram of our methods

Beginning with a dataset containing patient heart attack data, we will apply three simple machine learning methods - k-nearest neighbors, naive Bayes, and decision trees - testing relevant hyperparameters and model variations as appropriate. Upon obtaining the accuracy of each of these models, we will compare them to determine which models are the most effective.

2.1 Dataset Description

Our dataset is "Heart Attack Analysis & Prediction Dataset", uploaded to Kaggle by Rashik Rahman (https://www.kaggle.com/rashikrahmanpritom/heart-attack-analysis-prediction-dataset). Its attributes include:

- *age*: age of the patient
- *sex*: gender of the patient
- *cp*: type of chest pain that the patient is experiencing (0, 1, 2, or 3)
- *trtbps*: the patient's resting blood pressure in mm Hg
- *chol*: the patient's cholesterol in mg/dl as obtained by BMI sensor
- *fbs*: indicates whether the patient's fasting blood sugar is greater than 120 (1 = true, 0 = false)
- *restecg*: the patient's resting electrocardographic results
- *thalachh*: the patient's maximum heart rate measured
- *exng*: exercised induced angina 1 = yes, 0 = no
- *oldpeak*: the patient's previous peak
- *slp*: slope
- *caa*: the patient's number of major blood vessels
- *thall*: the patient's Thal rate

The class variable is *output* - whether or not the patient experienced a heart attack.

2.2 Methodology

2.2.1 K-nearest Neighbors

K-nearest neighbors (KNN) is a method that classifies an instance by comparing it to its k closest neighbors. To classify an instance, the algorithm begins by computing the distance between it and each instance in the training dataset. For this study we will use the Euclidian distance metric. It then returns the majority class of the k instances closest to the instance in question. We selected this algorithm because it is simple to implement and works well on numerical data.

Before applying the KNN algorithm, we will normalize our dataset using min-max scaling. We will then test the algorithm with all possible values of k, ranging from 1 to the size of our training dataset.

2.2.2 Naive Bayes Classifier

The **naive Bayes classifier** is a probabalistic method for predicting the class of an instance. Given features $F = \{f_1, f_2, ..., f_n\}$ and classes $C = \{c_1, c_2, ..., c_m\}$, the naive Bayes classifier c_{NB} is defined as:

$$c_{NB} = \underset{c \epsilon C}{\operatorname{argmax}} P(c) \prod_{f \epsilon F} P(f|c) \tag{1}$$

(Jurafsky and Martin, 2020). Given an instance $(x_1, x_2, ..., x_n)$, this classifier yields its most likely class based on the training data. If an attribute is discrete, its conditional probability will be calculated by relative frequency; if continuous, the conditional probability will be calculated according to the Gaussian probability density function. We chose this model due to its flexibility; it can be used properly on both categorical and continuous data.

2.2.3 Decision Trees

A **decision tree** applies a sequence of tests to an instance's attribute values to determine the instance's class. Each node of the tree corresponds to a test on an attribute; its branches correspond to the attribute's possible values. Each leaf corresponds to a class (Russel and Norvig, 2020). See the associated figure for an example. We selected this model because it is a simple yet powerful way

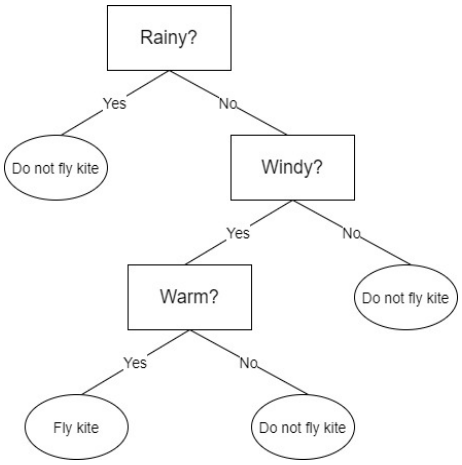

Figure 2: An example decision tree for deciding whether to fly a kite

to classify instances with both categorical and continuous attributes.

There are two methods for selecting which attributes to add next while constructing a decision tree - entropy and the GINI index. We will build two decision trees, applying one of these methods to each, then compare their accuracies.

2.3 Performance Evaluation

To partition the dataset into training and testing data, we will first shuffle its instances. We will then select the last 60 of these instances (approximately 20 percent of the data) to use as our testing data. The remaining 243 instances will be used as training data. Upon fitting each model to the training data and applying each model to the testing data, we will report the model accuracy for each. We will also generate a confusion matrix for each model, by which we will obtain the numbers of true positives, false positives, true negatives, and false negatives.

3 Experimental Analysis

3.1 K-nearest neighbors

We found that $k = 7$ yielded the highest accuracy (83.33%) for the KNN algorithm with min-max scaling.

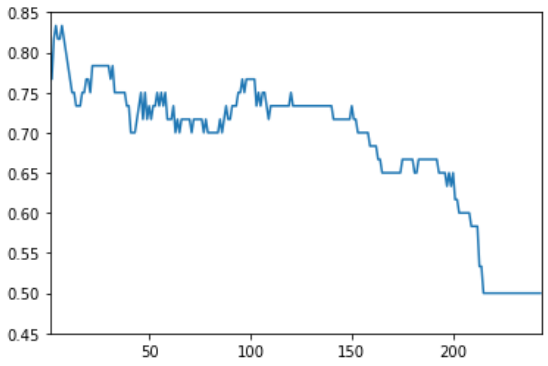

Figure 3: Plot of k vs. accuracy

KNN ($k = 7$)	**Predicted 1**	**Predicted 0**
Actual 1	28	2
Actual 0	8	22

Table 1: Confusion matrix for KNN algorithm

3.2 Naive Bayes Classifier

The naive Bayes classifier yielded an accuracy of 85%.

Naive Bayes	**Predicted 1**	**Predicted 0**
Actual 1	27	3
Actual 0	6	24

Table 2: Confusion matrix for naive Bayes classifier

3.3 Decision trees

The decision tree constructed using the GINI index yielded an accuracy of 82%; using entropy yielded an accuracy of 80%.

DT (GINI)	**Predicted 1**	**Predicted 0**
Actual 1	27	3
Actual 0	7	23

Table 3: Confusion matrix for decision tree with GINI

DT (Entropy)	**Predicted 1**	**Predicted 0**
Actual 1	27	3
Actual 0	9	21

Table 4: Confusion matrix for decision tree with entropy

3.4 Model Comparison

Upon comparing the accuracy of each model, we found that the naive Bayes classifier was the most accurate. It was followed by KNN, decision trees with GINI, and decision trees with entropy, respectively.

KNN ($k = 7$)	**Naive Bayes**	**DT (GINI)**	**DT (entropy)**
83.33%	85%	82%	80%

Table 5: Accuracy obtained from each model

3.5 Files

Our data, code, and all other relevant files can be obtained from `https://github.com/nhNetz/CMPSC_445_Heart`.

4 Conclusion

We used a set of patient medical data to train and test three simple machine learning models with the objective of detecting heart attacks. For each model,

we tested different values for relevant hyper-parameters and model variations. Based on our results, the naive Bayes classifier is the most accurate at detecting heart attacks. It is followed by the KNN algorithm (with $k = 7$ and min-max scaling), decision trees (with GINI), and decision trees (with entropy) respectively. All models yielded an accuracy of approximately 80% or higher. Future work could include experimenting with other models, applying different performance evaluation techniques, or testing models on different combinations of attributes.

4.1 Contributions

Each author's contribution to this project is listed below.

- Nathaniel Netznik implemented the KNN algorithm and wrote the report.
- Huy Ngo implemented the decision trees.
- Zheyu Zhang implented the naive Bayes classifier.
- The authors collaborated to review each other's work and to address all other aspects of the study.

References

Chitra, R. and Seenivasagam, V. (2013). Review of heart disease prediction system using data mining and hybrid intelligent techniques. *ICTACT Journal On Soft Computing*, 3(4):605–609. Retrieved from `https://core.ac.uk/download/pdf/25558318.pdf`.

Gawale, R. and Gawande, R. M. (2020). A machine learning approach to heart attack prediction. *International Journal of Advance Research and Innovative Ideas in Education*, 6(4). Retrieved from `https://www.ijeat.org/wp-content/uploads/papers/v8i4/D6354048419.pdf`.

Jurafsky, J. and Martin, J. H. (2020). Speech and language processing. `https://web.stanford.edu/~jurafsky/slp3/4.pdf`.

Karthikeyan, M., Myakala, C. R., and Chappidi, S. C. (2020). Heart attack prediction using xgboost. *International Journal of Advanced Science and Technology*, 29(6):2392–2399. Retrieved from `http://sersc.org/journals/index.php/IJAST/article/view/13543`.

Ranga, V. and Rohila, D. (2018). Parametric analysis of heart attack prediction using machine learning techniques. *International Journal of Grid and Distributed Computing*, 11(4):37–48. Retrieved from `http://article.nadiapub.com/IJGDC/vol11_no4/4.pdf`.

Russel, S. J. and Norvig, P. (2020). *Artificial intelligence: A modern approach*. Pearson.

Ware, S., Rakesh, S. K., and Choudhary, B. (2020). Heart attack prediction by using machine learning techniques. *International Journal of Recent Technology and Engineering*, 8(5):1577–1580. Retrieved from `https://www.ijrte.org/wp-content/uploads/papers/v8i5/D9439118419.pdf`.