Heart Disease Prediction Using Machine Learning Techniques

Rajasekhar Golanakonda
Data Science
Coventry University
Coventry, England
golanakonr@coventry.ac.uk

Abstract*— This study investigates the application of machine learning algorithms in predicting heart disease based on patient behaviour data. We employ Logistic Regression, Decision Trees, Support Vector Machines, K-Nearest Neighbors, Naive Bayes, Random Forest, and Gradient Boosting Classifiers on a heart disease dataset obtained from the Kaggle repository. The Python programming language is utilized along with established machine learning libraries to implement these algorithms. The performance of each algorithm is evaluated using relevant metrics, and the results are visualized for comparative analysis. This analysis aims to identify the most suitable machine-learning technique for predicting heart disease within the context of this specific dataset.

I. Introduction

Heart disease and stroke are examples of cardiovascular diseases (CVDs), which constitute a global health crisis. The World Health Organisation (WHO) estimates that these illnesses cause an astounding 12 million fatalities per year. This concerning figure highlights how urgent it is to create efficient plans for managing and preventing heart disease, which is the world's leading cause of death.

Fortunately, there are promising technologies available to address this difficulty because to breakthroughs in machine learning (ML) and artificial intelligence (AI). ML algorithms are able to discover intricate patterns and correlations between different parameters and the risk of heart disease by utilising large datasets of patient data. Using this knowledge, one can then:

Predict: Determine who is most likely to get heart disease so that preventative actions that could save lives can be implemented early on.

Diagnose: Improve treatment outcomes by helping medical practitioners diagnose cardiac disease more precisely and quickly.

To enhance patient care and quality of life, treatment programmes can be personalised by analysing patient data and making recommendations.

This investigation will look into the potential applications of machine learning techniques in the fight against heart disease, a worldwide health issue that calls for creative solutions. We'll look at different machine learning techniques, their possible advantages, and the difficulties in putting them into practice in the fight against this common and deadly illness.

II. Problem and Dataset(s)

A. Problem

The purpose of this study is to look into the main risk factors that lead to the development of heart disease. It also aims to create machine learning models that are capable of accurately predicting a patient's vulnerability to this common and potentially fatal illness.

B. Dataset

FEATURES:
The dataset comprises numerical features, with some converted into categorical features. Further details regarding feature types can be discerned during exploratory data analysis (EDA).

Feature Name	**Description**
Age	The number of years a person has lived.
Sex	Gender of the patient, where Male is represented by 1 and Female by 0.
Chest Pain Type (cp)	Categorized into four values.
Resting Blood Pressure (trestbps)	Resting blood pressure in mm Hg.
Serum Cholesterol Level (chol)	Serum cholesterol level in mg/dl.
Fasting Blood Sugar (fbs)	Indicates whether fasting blood sugar is above 120 mg/dl.
Resting Electrocardiographic (ECG) Results (restecg)	Resting electrocardiographic results with values 0, 1, or 2.
Maximum Heart Rate Achieved (thalach)	Maximum heart rate achieved during exercise test.
Exercise Induced Angina (exang)	Indicates whether exercise induced angina is present (yes or no).
ST Depression Induced by Exercise Relative to Rest (oldpeak)	ST depression induced by exercise relative to rest (mm)
Slope	Represents the slope of the peak exercise ST segment.
Number of Major Vessels Colored by Fluoroscopy (ca)	Number of major vessels (0-3) colored by fluoroscopy.
Thalium Stress Test Results (thal)	Thalium stress test results: 3 = normal, 6 = fixed defect, 7 = reversible defect.
condition	0 = no disease, 1 = disease

TARGET VARIABLE:
The target variable, derived from the original dataset, signifies the diagnosis of heart disease, specifically angiographic disease status.

0:Less than 50% diameter narrowing (negative for heart disease).
1: Greater than 50% diameter narrowing (positive for heart disease

III. Machine Learning Methods

A. *Logistic Regression*

- Linear model predicting the probability of an event belonging to a specific class (often binary: 0 or 1).
- Widely used for its interpretability (coefficients represent feature importance) and efficiency.
- Works well for linearly separable data.

B. *K-Nearest Neighbor(KNN)*

- Classifies data points based on the majority vote of their k nearest neighbors in the feature space.
- Simple to implement but requires careful selection of the distance metric and number of neighbors (k).

C. *SVM*

- Finds a hyperplane separating data points belonging to different classes with the maximum margin.
- Powerful for high-dimensional data and works well with small datasets.
- Can be less interpretable than other algorithms.

D. *Decision Tree*

- The tree-like structure where splits are made based on feature values to classify data points.
- Offers interpretability through the decision path.
- Prone to overfitting if not regularized (e.g., limiting tree depth).

E. *Random Forest*

- Ensemble method combining multiple decision trees, improving accuracy and reducing overfitting.
- Offers robustness and handles mixed-type features well.
- Interpretability can be challenging due to the ensemble nature.

IV. EXPERIMENT RESULTS

An Initial logistical regression was performed to check the accuracy of the classification dataset.

. *Table I – Logistic Regression accuracy scores*

Train Accuracy	Test Accuracy
85.5%	81.1%

The results indicate that the trained Logistic Regression model achieved a training accuracy of approximately **85.5%** and a test accuracy of approximately **81.1%.** The confusion matrix provides additional insights into the model's performance, showing the distribution of correct and incorrect predictions across different classes.

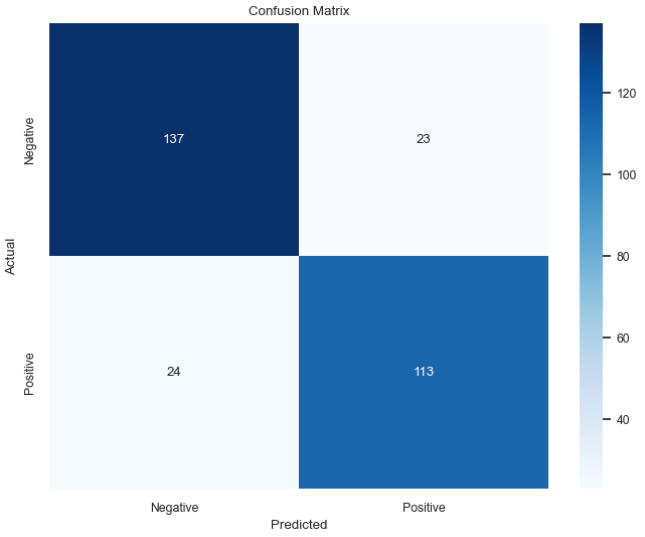

Fig. 1 Confusion Matrix for Logistic Regression

The confusion matrix summarizes the performance of the classification model by presenting the counts of true positive, true negative, false positive, and false negative predictions. It helps in understanding the model's ability to correctly classify instances and identify any potential errors.
In the provided confusion matrix:

- The top-left cell (**137**) represents the count of true negatives (**TN**), indicating the number of instances correctly classified as negative.
- The top-right cell (**23**) represents the count of false positives (**FP**), indicating the number of instances incorrectly classified as positive.
- The bottom-left cell (**24**) represents the count of false negatives (**FN**), indicating the number of instances incorrectly classified as negative.
- The bottom-right cell (**113**) represents the count of true positives (**TP**), indicating the number of instances correctly classified as positive.

Table II – K-value, accuracy scores

K-Value	Train Accuracy	Test Accuracy
14	82.1%	80%

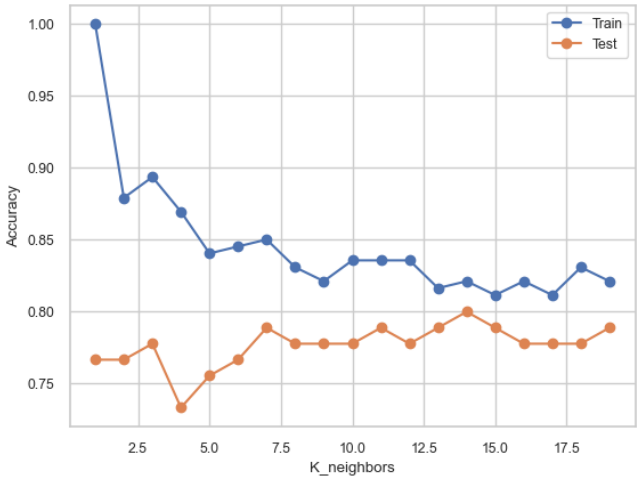

Fig. 2 Knn values VS Accuracies

The K-value and the accompanying accuracy ratings that resulted from using the K-Nearest Neighbors (KNN) algorithm are displayed in Table II. Applied to pattern recognition applications, KNN is a non-parametric classification technique. The KNN algorithm produced 82.1% train accuracy and 80% test accuracy for a K-value of 14. The percentage of correctly categorized instances in the training and testing datasets is shown by these accuracy scores.

The test accuracy indicates the model's capacity for generalization on untested data, whereas the training accuracy shows how well the model performed on the training dataset.

The KNN algorithm's ability to learn from the training data and produce precise predictions on new, unknown data is demonstrated by these accuracy scores. The comparatively high test accuracy indicates that the model can generalize well to new cases and has successfully learned patterns from the training data. To maximize the K-value and maybe enhance the model's performance, more research and testing might be required.

Table III – SVM Kernels and accuracy scores

Kernel Name	Train Accuracy	Test Accuracy
poly	92.7%	84.4%
linear	85.9%	81.1%
sigmoid	50.2%	48.8%
rbf	86.9%	78.8%

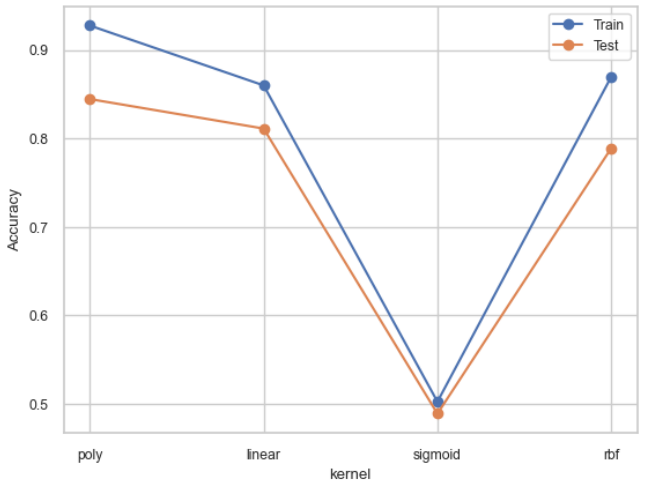

Fig. 3 SVM Kernels VS Accuracies

In Table III, we present the different kernel functions used in the Support Vector Machines (SVM) algorithm along with their corresponding accuracy scores. SVM is a powerful supervised learning algorithm commonly used for classification tasks.

For each kernel function, the table displays both the train accuracy and the test accuracy achieved by the SVM algorithm.

Poly Kernel:

The SVM model utilizing the polynomial kernel achieved a training accuracy of 92.7% and a test accuracy of 84.4%.

These accuracy scores indicate the model's ability to correctly classify instances in both the training and testing datasets when using the polynomial kernel.

Linear Kernel:

With the linear kernel, the SVM model attained a train accuracy of 85.9% and a test accuracy of 81.1%.

These accuracy scores demonstrate the performance of the SVM model when using a linear decision boundary to separate the classes.

Sigmoid Kernel:

The sigmoid kernel yielded a train accuracy of 50.2% and a test accuracy of 48.8%.

These accuracy scores suggest poor performance of the SVM model with the sigmoid kernel, as the accuracy is close to random guessing.

RBF Kernel:

Using the radial basis function (RBF) kernel, the SVM model achieved a training accuracy of 86.9% and a test accuracy of 78.8%.

These accuracy scores indicate the SVM model's performance when employing a non-linear decision boundary generated by the RBF kernel.

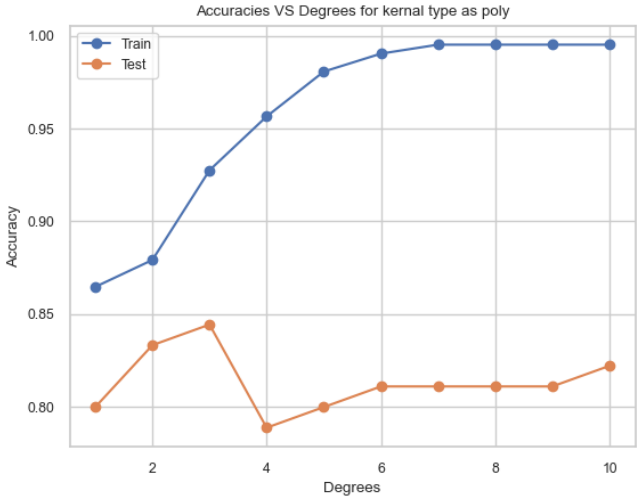

Fig. 4 Degree VS Accuracies

The Poly Kernel achieves the highest test accuracy of 84.4% when the degree value is 3.

Table IV – Decision Tree(without hyper tune) accuracy scores

Train Accuracy	Test Accuracy
100%	82%

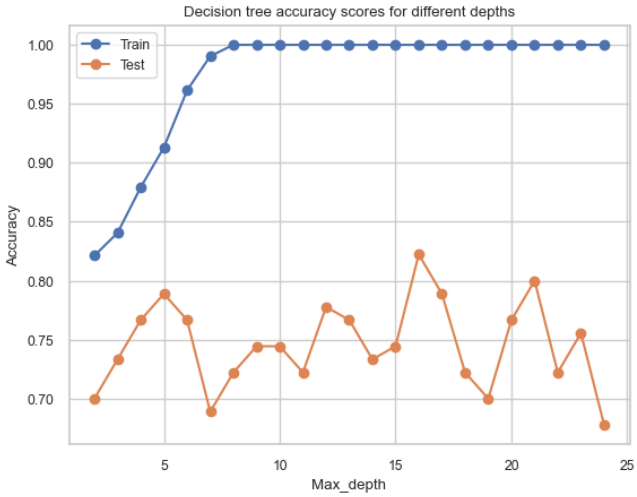

Fig. 5 Decision Tree plot (Accuracies VS depths)

Fig.5 represents the classification accuracy of a decision tree model on a given dataset. It showcases two key performance metrics: training accuracy and test accuracy.

- **Train Accuracy (100%)**: This indicates that the decision tree perfectly classified all instances in the training data.
- **Test Accuracy (82%)**: This reflects the model's ability to generalize to unseen data. The model correctly classified 82% of the instances in the test set.

Observations:

The significant difference between training accuracy (100%) and test accuracy (82%) suggests potential **overfitting** in the decision tree model. Overfitting occurs when the model memorizes the specific patterns in the training data and fails to capture the underlying relationships that generalize well to unseen examples.

Table V – Decision Tree hyper tuning parameters and accuracy scores

criterion	Max depth	Max Features	Min Samples Leaf	Min Samples Split	Train accuracy	Test Accuracy
entropy	5	Log2	1	2	88.8%	74.4%

The below figure represents a decision tree plot for sample of 5 depths.

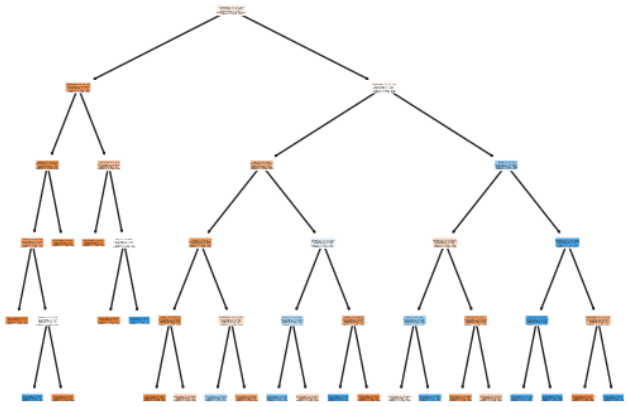

Fig.6 Decision Tree plot for max_depths=5 after hypertuning

The table provides the results of applying Decision Tree classifiers with different hyperparameters, along with their corresponding accuracy scores on the training and testing datasets. Let's interpret these results:

- **MaxDepth**: This parameter controls the maximum depth of the decision tree. A higher max depth allows the tree to capture more complex relationships in the data but increases the risk of overfitting.
- **MaxFeatures**: It determines the maximum number of features considered for splitting at each node. Using fewer features can reduce the model's complexity and improve generalization.
- **MinSamplesLeaf**: This parameter sets the minimum number of samples required to be at a leaf node. A higher value can prevent the model from creating nodes with very few samples, reducing overfitting.
- **MinSamplesSplit**: It specifies the minimum number of samples required to split an internal node. Similar to min_samples_leaf, it helps control the tree's growth and prevent overfitting.

Now, let's analyse the accuracy scores:

Best Accuracy:

The combination with the highest test accuracy is:

- Criterion: entropy
- MaxDepth: 5
- MaxFeatures: Log2
- MinSamplesLeaf: 1
- MinSamplesSplit: 2
- Test Accuracy: 74.4%

This configuration achieves the highest accuracy on the testing dataset, making it the best-performing model among the options provided.

In summary, the decision tree with a Gini criterion, max depth of 5, log2 max features, minimum samples per leaf of 4, and minimum samples per split of 5 demonstrates the best performance in terms of accuracy on the testing dataset.

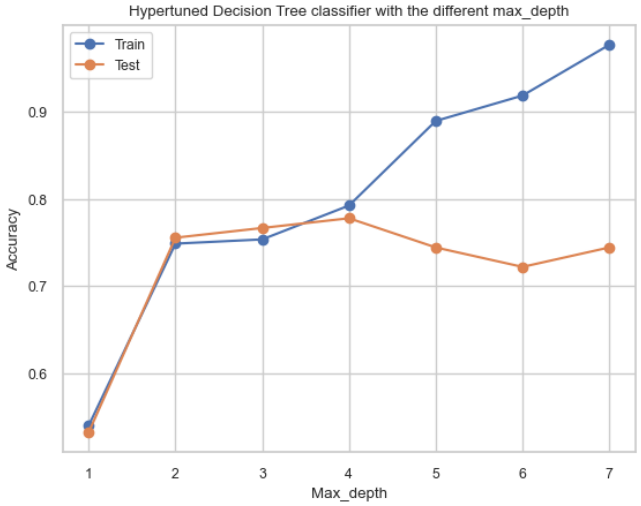

Fig.7 Decision Tree plot for different max_depths after hypertuning

Table VI – Navie Bayes accuracy scores

Train Accuracy	Test Accuracy
85.5%	81.1%

Table VII – Random Forest hyper tuning parameters and accuracy scores

N_estimators	Max depth	Max Features	Min Samples Leaf	Min Samples Split	Train accuracy	Test Accuracy
50	5	sqrt	1	2	96.1%	80%

V. CONCLUSION

ML Technique	Train Accuracy	Test Accuracy
Logistic Regression	85.5%	81.1%
KNN	82.1%	80%
SVM	92.7%	84.4%
Decision Tree	88.8%	74.4%
Navie Bayes	85.5%	81.1%
Random Forest	96.1%	80%

Based on the table, the best model appears to be **SVM (Support Vector Machine)**. Here's the reasoning:

- **Highest Test Accuracy:** SVM achieves the highest test accuracy (84.4%) among the listed models. This metric is most important, as it reflects how well the model generalizes to unseen data.
- **Balance Between Train and Test Accuracy:** While SVM has a high training accuracy (92.7%), the gap between training and test accuracy is lower compared to some other models (e.g., Logistic Regression, Decision Tree). This suggests a good balance between fitting the training data and generalizing to unseen examples.

VI. APPENDIX

The data is source is taken from kaggle and below is the link https://www.kaggle.com/code/shtrausslearning/heart-disease-gaussian-process-models/input?select=heart_cleveland_upload.csv

Python code for model evaluation is available in below git repository

https://github.com/Rajsekhar12-tech/Machine_learning

REFERENCES

[1] Quinlan, J. R. (1986). Induction of decision trees. Machine learning, 1(1), 81-106.

[2] Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician, 46*(3), 175-185.

[3] Cortes, C., & Vapnik, V. (1995). *Support-vector networks. Machine learning, 20*(3), 273-297.

[4] McCallum, A., & Nigam, K. (1998). A comparison of event models for Naive Bayes text classification. In AAAI-98 workshop on learning for text categorization (Vol. 752, No. 1, pp. 41-48)

[5] Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. R news, 2(3), 18-22.

[6] Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1), 21-27

[7] Rish, I. (2001). An empirical study of the naive Bayes classifier. In IJCAI 2001 workshop on empirical methods in artificial intelligence (Vol. 3, pp. 41-46).

[8] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

[9] Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). Applied logistic regression (Vol. 398). John Wiley & Sons.