Heart Disease Detection Using Machine Learning Algorithm

Pham Duc Toan
Faculty of Computer Science
University of Information Technology, VNU-HCM
Email: 21521550@gm.uit.edu.vn

Tran Nhat Anh
Faculty of Computer Science
University of Information Technology, VNU-HCM
Email: 21521841@gm.uit.edu.vn

Nguyen Duong Quoc Anh
Faculty of Computer Science
University of Information Technology, VNU-HCM
Email: 21521829@gm.uit.edu.vn

Le Quoc Trung
Faculty of Computer Science
University of Information Technology, VNU-HCM
Email: 21522723@gm.uit.edu.vn

July 2, 2023

Abstract

Health care [1] is an inevitable task to be done in human life. Health concern business has become a notable field in the wide spread area of medical science. Health care industry contains large amount of data and hidden information. Effective decisions are made with this hidden information by applying data mining techniques [2]. Several tests are done in the detection of cardiovascular diseases in the patient; however with data mining these tests could be reduced. But there is a lack of analysing tool to provide effective test results with the hidden information, so a system is developed using data mining algorithms for classifying the data and to detect the heart diseases.This paper we will analyse four algorithms: Decision tree, Support Vector Machine , K-nearest neighbors, Random Forest which are applied for detecting heart disease.

The results show that Random Forest model achieved the highest metric outperforming the SVM , KNN , and Decision Tree models in terms of overall performance. While the Random Forest model demonstrated superior overall performance, it is important to consider additional factors when evaluating these models. The paper highlights that SVM has the highest recall value, indicating its ability to accurately identify positive cases of heart disease. Furthermore, the accuracy of SVM is reported to be comparable to that of the Random Forest model.

I. Introduction

The heart is one of the main organs of the human body. It pumps blood trough the blood vessels of the circulatory system. The circulatory system is extremely important because it transports blood, oxygen and other materials to the different organs of the body. Heart plays the most crucial role in circulatory system. If the heart does not function properly then it will lead to serious health conditions including death. Today, Cardiovascular diseases are the number 1 cause of death worldwide, claiming an estimated 17.9 million lives each year, which represents 31% of all deaths worldwide. Four out of every 5 deaths from cardiovascular disease are due to heart attacks and strokes, and a third of these deaths occur prematurely in people under 70 years of age.

Data mining has been played an important role in the intelligent medical health care systems. Medical data mining in health care is regarded as an important yet complicated task that needs to be executed accurately and efficiently. Health care data mining attempts to solve real world health problems in diagnosis and treatment of disease. The relationship of disorders and real cause of disorders and the effects of symptoms that are spontaneously seen in patients can be evaluated by using the heart disease prediction system, is a computerized method for diagnosing heart diseases based on prior data and information.

II. Related work

Heart disease is a term that assigns to a large number of medical conditions related to heart. These medical conditions describe the abnormal health conditions that directly influence the heart and all its parts. Heart disease is a major health problem in today's time. Table 1 shows different data mining techniques used in the diagnosis of Heart disease over different Heart disease datasets. In some papers this is given that they use only one technique for diagnosis of heart disease as given in Shadab et al ,Carlos et al etc. but in case of other research work more than one data mining techniques are used for the diagnosis of heart disease as given in Ms. Ishtake et al., .JABBAR, et al, Shantakumar et al etc.

The related work section will focus on exploring previous research and studies in this domain. We will examine various approaches and techniques employed by researchers in utilizing machine learning algorithms to identify and diagnose heart disease. By analyzing the existing literature, we aim to gain insights into the strengths, limitations, and advancements in this field. This comprehensive review of related work will provide a solid foundation for our research and contribute to the existing knowledge in the area of heart disease detection. Including four methods that used to predict the ouput, we will find out which is the most suitable with the dataset.

Author	Year	Technique used	Attributes
Dr. K. Usharani	2011	Classification/Neural Networks	13
Jesminahar, et al	2013	Apriori/Predictive Apriori/Tertius	14
Latha, et al	2008	Genetic Algorithm/CANFIS	14
Majabber, et al	2011	Clustering/Association Rule Mining/Sequence number	14
Ms.Lshtake et al.	2013	DecisionTree/Neural Network/Naive bayes	15

Table 1: Table shows different data mining techniques used in the diagnosis of Heart disease over different Heart disease datasets.

- **Decision Tree**: deployed by **Pham Duc Toan**.
- **SVM**: deployed by **Le Quoc Trung**.
- **KNN**: deployed by **Nguyen Duong Quoc Anh**.
- **Random Forest** : deployed by **Tran Nhat Anh**.

III. Our Method

1. Decision Tree

A **decision tree** is a hierarchical model [3] used in decision support that depicts decisions and their potential outcomes, incorporating chance events,utilizes a tree-like model to make decisions based on input features, resource expenses, and utility. This algorithmic model utilizes conditional control statements and is non-parametric, supervised learning, useful for both classification and regression tasks. The tree structure is comprised of a root node, branches, internal nodes, and leaf nodes, forming a hierarchical, tree-like structure. In the context of heart disease detection, decision trees can be trained on a dataset of patient information, including various risk factors and symptoms, to predict the presence or absence of heart disease

A **decision tree** includes:

- Root Nodes: It is the node present at the beginning of a decision tree from this node the population starts dividing according to various features.
- Decision Nodes: the nodes we get after splitting the root nodes are called Decision Node
- Leaf Nodes: the nodes where further splitting is not possible are called leaf nodes or terminal nodes
- Sub-tree: just like a small portion of a graph is called subgraph similarly a sub-section of this decision tree is called sub-tree.
- Pruning: is nothing but cutting down some nodes to stop overfitting.

Entropy and Information Gain:

Entropy is employed to measure a dataset's impurity or randomness [4], [5]. The value of entropy always lies between 0 and 1. Its value is better when it is equal to 0 while it is worse when it is equal to 0, i.e. the closer its value to 0 the better. As shown in "Fig. 3". If the target is G with different attribute values, the entropy of the classification of set S with respect to c states [6], [7]. As shown in "equation (1)".

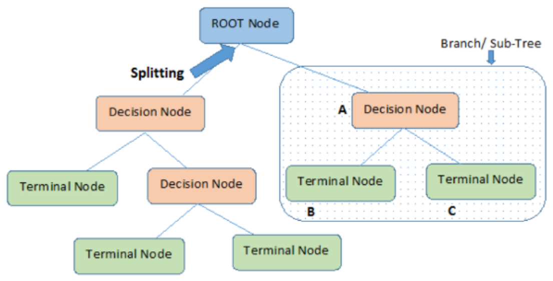


Figure 1: Decision tree

$$\text{Entropy}(S) = \sum_{i=1}^{c} P_i \log_2(P_i) \tag{1}$$

Where P_i is the ratio of the sample number of the subset and the i-th attribute value.

Information gain is one metric used for segmentation and is often called mutual information. This intuitively informs how much knowledge of a random variable's value [8, 9] . It's the opposite of entropy, the higher its value is the better. The data gain *Gain(S,A)* is defined as the following: of entropy [10,11], as shown in "equation (2)"

$$\text{Gain}(S, A) = \sum_{v \in V(A)} \frac{|S_v|}{|S|} \cdot \text{Entropy}(S_v) \tag{2}$$

Where the range of attribute A is *V(A)*, and S_v is a subset of set S equal to the attribute value of attribute v[10] .

2. SVM

The **Support Vector Machines (SVM)** is a powerful machine learning algorithm used for classification and regression tasks. It finds the best hyperplane to separate classes and can handle both linear and non-linear datasets using the kernel trick. **SVM** is effective in high-dimensional spaces, robust against overfitting, and offers versatility through different kernel functions. However, it requires careful parameter tuning and can be computationally expensive for large datasets. Overall, **SVM** is a popular and effective algorithm in machine learning.

SVM is used in computer vision, natural language processing, bioinformatics, finance, medical diagnosis, spam detection, handwriting recognition, face detection, social network analysis, recommendation systems, and quality control.

The optimization problem in SVM:

The optimization problem in SVM (Support Vector Machine) is a crucial part of building an SVM model. The objective of the optimization problem is to find an optimal decision boundary for classifying data points belonging to different classes in a high-dimensional space.

In SVM, the optimization problem involves searching for optimal hyperparameters and necessary constraints to determine

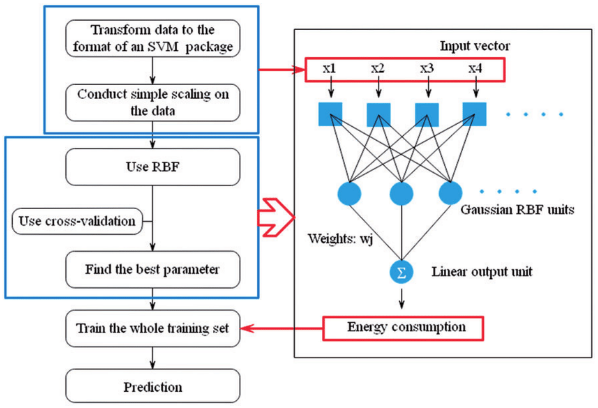


Figure 2: SVM Diagram

the best decision boundary. Specifically, the SVM optimization problem is typically formulated as follows:

$$min_{w,b,e}(\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i) \tag{1}$$

subject to:

$$y_i(w \cdot x_i - b) \geq 1 - \xi_i, \quad \forall i \; \xi_i \geq 0, \quad \forall i \tag{2}$$

Where:

w is the weight vector of the decision boundary.b is the bias term.xi is the i-th data point.yi is the label of data point x_i (1 or -1).xii is a slack variable that measures the deviation of a data point from the boundary and constraints.

C is the regularization parameter to control the trade-off between maximizing the margin between the closest data points and minimizing the classification error. This parameter needs to be tuned to achieve a balance between model accuracy and complexity.

1.Linear Kernel:

Equation:

$$\mathrm{K}(x, x') = x \cdot x' \tag{3}$$

The linear kernel is used when the data can be well separated by a straight line or hyperplane.

The decision boundary is defined by a straight line or hyperplane in the original feature space. Suitable for linearly separable classification problems

2.Polynomial Kernel:

Equation:

$$\mathrm{K}(x, x') = (x \cdot x' + r)^d \tag{4}$$

The polynomial kernel is used when the data exhibits non-linear separability.

The parameter d represents the degree of the polynomial, and r is the bias term.The polynomial kernel allows for complex curves to separate the data.

3.Gaussian (RBF) Kernel:

Equation:

$$\mathrm{K}(x, x') = \exp\left(-\frac{|x - x'|^2}{2\sigma^2}\right) \tag{5}$$

The Gaussian kernel is used when the data exhibits non-linear separability and the underlying feature shape is unknown.

The parameter sigma determines the width of the Gaussian bell curve.

The Gaussian kernel creates complex non-linear curves, allowing for flexible data separation in high-dimensional spaces.

4.Sigmoid Kernel:

Equation:

$$\mathrm{K}(x, x') = \tanh(\alpha x \cdot x' + c) \tag{6}$$

The sigmoid kernel is used to handle non-linear and non-homogeneous data.The parameters alpha and c control the shape of the sigmoid curve.The sigmoid kernel creates S-shaped curves that can separate non-linear data.

3. KNN

The **K-nearest Neighbors Algorithm**, also known as **KNN** or **k-NN**, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data. For classification problems, a class label is assigned on the basis of a majority vote—i.e. the label that is most frequently represented around a given data point is used.

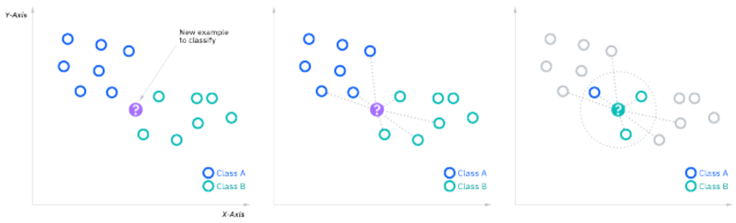


Figure 3: KNN Diagram

Distance Metrics Used in KNN Algorithm

As we know that the **KNN Algorithm** helps us identify the nearest points or the groups for a query point. But to determine the closest groups or the nearest points for a query point we need some metric. For this purpose, we use below distance metrics:

- **Euclidean Distance**: This is nothing but the cartesian distance between the two points which are in the plane/hyperplane. Euclidean distance can also be visualized as the length of the straight line that joins the two points which are into consideration. This metric helps us calculate the net displacement done between the two states of an object.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{7}$$

- **Manhattan Distance**: This distance metric is generally used when we are interested in the total distance traveled by the object instead of the displacement. This metric is calculated by summing the absolute difference between the coordinates of the points in n-dimensions.

$$d(x,y) = \sum_{i=1}^{n}|x_i - y_i| \tag{8}$$

- **Minkowski Distance**: This distance measure is the generalized form of Euclidean and Manhattan distance metrics. The parameter, p, in the formula below, allows for the creation of other distance metrics. Euclidean distance is represented by this formula when p is equal to two, and Manhattan distance is denoted with p equal to one.

$$d(x,y,p) = \left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{\frac{1}{p}} \tag{9}$$

4. Random Forest

Random forest is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.[13][14]

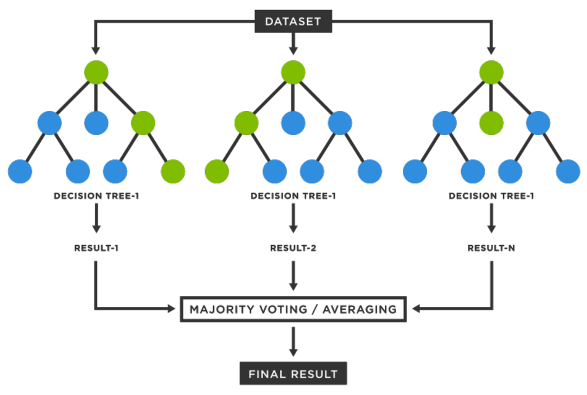


Figure 4: Random forest diagram

Algorithm:

1. Bagging [15]:

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $X = x_1, ..., x_n$ with responses $Y = y_1, ..., y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For b = 1, ..., B:

1. Sample, with replacement, n training examples from X, Y; call these Xb, Yb.
2. Train a classification or regression tree fb on Xb, Yb.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

or by taking the majority vote in the case of classification trees.

This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

Additionally, an estimate of the uncertainty of the prediction can be made as the standard deviation of the predictions from all the individual regression trees on x':

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}}.$$

The number of samples/trees, B, is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and nature of the training set. An optimal number of trees B can be found using cross-validation, or by observing the out-of-bag error: the mean prediction error on each training sample xi, using only the trees that did not have xi in their bootstrap sample.[15] The training and test error tend to level off after some number of trees have been fit.

2. From bagging to random forests [16]:

The above procedure describes the original bagging algorithm for trees. Random forests also include another type of bagging scheme: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called "feature bagging". The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated.

3. Relationship to nearest neighbors:

A relationship between random forests and the k-nearest neighbor algorithm (k-NN) was pointed out by Lin and Jeon in 2002. It turns out that both can be viewed as so-called weighted neighborhoods schemes. These are models built from a training set $\{(x_i, y_i)\}_{i=1}^{n}$ that make predictions $\hat{y}$ for new points x' by looking at the "neighborhood" of the point, formalized by a weight function W:

$$\hat{y} = \sum_{i=1}^{n} W(x_i, x')\, y_i$$

Here, $W(x_i, x')$ is the non-negative weight of the i'th training point relative to the new point x' in the same tree. For any particular x', the weights for points x_i must sum to one. Weight functions are given as follows:

- In k-NN, the weights are $W(x_i, x') = \frac{1}{k}$ if xi is one of the k points closest to x', and zero otherwise.
- In a tree, $W(x_i, x') = \frac{1}{k'}$ if xi is one of the k' points in the same leaf as x', and zero otherwise.

Since a forest averages the predictions of a set of m trees with individual weight functions W_j, its predictions are

$$\hat{y} = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{n} W_j(x_i, x')\, y_i = \sum_{i=1}^{n}\left(\frac{1}{m}\sum_{j=1}^{m} W_j(x_i, x')\right)\, y_i$$

This shows that the whole forest is again a weighted neighborhood scheme, with weights that average those of the individual trees. The neighbors of x' in this interpretation are the points x_i sharing the same leaf in any tree j. In this way, the neighborhood of x' depends in a complex way on the structure of the trees, and thus on the structure of the training set. Lin and Jeon show that the shape of the neighborhood used by a random forest adapts to the local importance of each feature. [17]

IV. Experiment

1. Pipeline of Buiding Models

A machine learning pipeline is a series of defined steps taken to develop, deploy and monitor a machine learning model. The approach is used to map the end-to-end process of developing, training, deploying and monitoring a machine learning model. It's often used to automate the process. Every stage of the machine learning process makes up a distinct module in the overall pipeline. Each component can then be optimised or automated. When building the machine learning pipeline, the orchestration of these different components is a major consideration.

The purpose of a machine learning pipeline is to outline the machine learning model process, a series of steps which take a model from initial development to deployment and beyond. The machine learning process is a complex one which spans different teams with different skills. Manually taking a machine learning model from development to deployment is a time consuming task. Outlining the machine learning pipeline means the approach can be refined and understood at a top-down level. Once outlined in a pipeline, elements can be optimised and automated to improve efficiency of the whole process. The entire flow of the machine learning pipeline can be automated in this way, freeing up human resources to focus on other considerations.

It's useful to understand the common machine learning pipeline architecture before starting the build. Overall, the components of the machine learning pipeline will be the series of steps taken to train, deploy and continuously optimise the model. Each individual section is a module that is outlined and explored in detail. The machine learning pipeline architecture also includes static sections like the data storage or archives for version control.Common sections of the machine learning pipeline include:

- Data ingestion: Data ingestion in a machine learning pipeline refers to the process of collecting and preparing raw data from various sources to be used as input for training or inference in machine learning models. It involves extracting data from diverse sources, transforming it into a suitable format, and loading it into a target system or storage for further processing.

  The data ingestion step is crucial because the quality and reliability of the data have a significant impact on the performance and accuracy of the machine learning models. The goal is to ensure that the data is comprehensive, consistent, and properly structured before it is fed into the models.

  Overall, data ingestion in a machine learning pipeline is a critical step that involves collecting, transforming, integrating, and loading data from various sources to enable effective training and inference of machine learning models. Proper data ingestion practices help ensure the accuracy and reliability of the models and the insights derived from them.

- Data preparation: also known as data preprocessing or data cleaning, is an essential step in machine learning that involves transforming raw data into a format suitable for training and evaluating machine learning models. Data preparation addresses issues such as missing values, outliers, noise, inconsistencies, and other irregularities in the data that could hinder the performance and accuracy of the models. The goal is to enhance the quality and reliability of the data before it is fed into the machine learning algorithms.

- Training of the model: Training of a model in machine learning refers to the process of using a dataset to build or train a machine learning model. It involves presenting the model with input data and allowing it to learn from that data to make accurate predictions or decisions.

- Tune Hyperparameters: Hyperparameter tuning is the process of finding the optimal values for the hyperparameters of a machine learning model. Hyperparameters are parameters that are set before training the model and cannot be learned from the data. They control aspects such as model complexity, regularization, learning rate, number of hidden layers, and more. Tuning hyperparameters is crucial for optimizing the performance of a model and improving its ability to generalize well to unseen data.

  - Grid Search: involves defining a grid of hyperparameter values and exhaustively evaluating the model's performance for each combination of hyperparameters. This technique performs an exhaustive search over all possible hyperparameter combinations and selects the one that yields the best performance. Grid search is simple to implement but can be computationally expensive when dealing with a large number of hyperparameters or a wide range of values.

  - Random Search: Unlike grid search, random search selects random combinations of hyperparameter values from predefined ranges. This approach explores the hyperparameter space more efficiently compared to grid search when the search space is large. By randomly sampling different combinations, it may quickly identify promising regions of the hyperparameter space.
  - Bayesian Optimization: uses probabilistic models to estimate the performance of a model for different hyperparameter configurations. It iteratively selects hyperparameters to evaluate based on their expected performance, updating the model's belief as more evaluations are performed. Bayesian optimization is more efficient compared to grid search and random search for high-dimensional hyperparameter spaces but may require more computational resources.

- Model Evaluation: Model evaluation is the process of assessing the performance and quality of a trained machine learning model. It involves measuring how well the model generalizes to unseen data and how effectively it solves the intended problem. Proper model evaluation is crucial to determine the model's reliability, compare different models, and make informed decisions about deploying the model in real-world scenarios.
  - Splitting the Data: The available dataset is typically divided into training, validation, and testing sets. The training set is used to train the model, the validation set is used for hyperparameter tuning and model selection, and the testing set is used for final evaluation. The testing set should represent unseen data that the model has not been exposed to during training and validation.
  - Confusion Matrix: In classification tasks, a confusion matrix provides a detailed breakdown of the model's predictions compared to the ground truth labels. It allows you to calculate metrics like accuracy, precision, recall, and F1 score for different classes, providing insights into the model's performance on individual classes.
  - Cross-Validation: Cross-validation is a technique used to estimate the model's performance by repeatedly splitting the data into different training and evaluation sets. Common cross-validation methods include k-fold cross-validation, stratified k-fold cross-validation, and leave-one-out cross-validation. Cross-validation provides a more robust estimate of the model's performance, particularly when the dataset is limited or imbalanced.

It's important to evaluate the model using appropriate evaluation techniques and metrics that align with the problem domain and business objectives. Model evaluation should be performed on independent test data that the model has not seen during training or hyperparameter tuning. Evaluating the model's performance on multiple evaluation metrics and considering various aspects of model behavior helps gain a comprehensive understanding of its strengths and weaknesses.

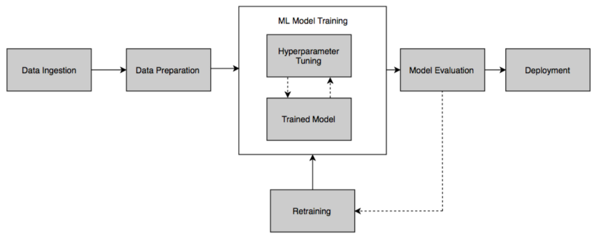


Figure 5: Pipeline for Machine Learning

2. Datasets

For datasets, we use the data heart disease dataset from kaggle which is included:

- Number of Instances: 918 , *random_state* =42
- Number of Attributes: 11

  1 - Age: patient's age (years)

  2 - Sex: patient's sex (M: Male, F: Female)

  3 - ChestPainType: chest pain type (TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic)

  4 - RestingBP: resting blood pressure (mm Hg)

  5 - Cholesterol: serum cholesterol (mm/dl)

  6 - FastingBS: fasting blood glucose (1: if FastingBS $>$ 120 mg/dl, 0: otherwise)

  7 - RestingECG: Resting ECG results (Normal: normal, ST: with ST-T wave abnormality, LVH: showing probable or definite left ventricular hypertrophy by Estes criteria)

  8 - MaxHR: maximum heart rate reached (Numeric value between 60 and 202)

  9 - ExerciseAngina: exercise-induced angina (Y: Yes, N: No)

  10 - Oldpeak: old peak = ST (Numerical value measured in depression)

  11 - ST_Slope: the slope of the peak exercise ST segment (Up upsloping, Flat: flat, Down downsloping)

- Missing Attributes Value: None
- Class Output Distribution:

Class	N	N [%]
1: heart disease	508	55.337691%
0: normal	410	44.662309%

- We divide the dataset into train and test in ratio 7:3 by using train test split

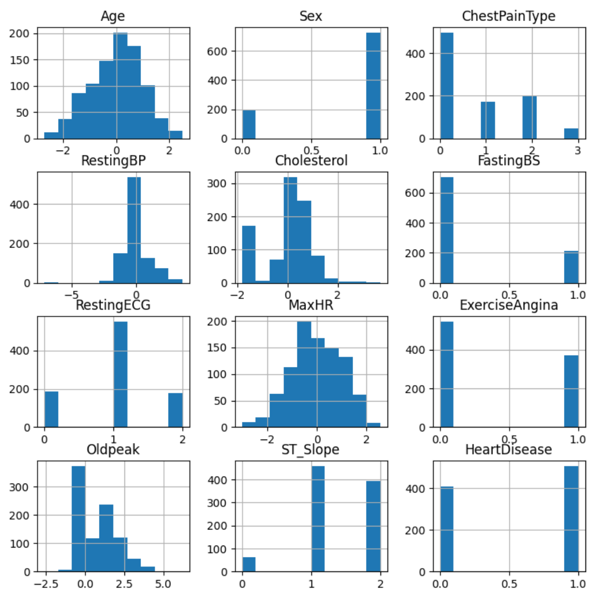
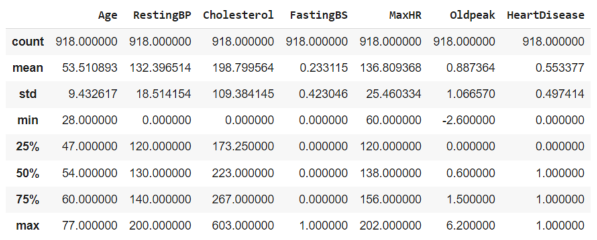


Figure 6: Visualize data

3. Exploratory Data Analysis

To understand the data in detail, EDA will help us to gain that. **Exploratory Data Analysis (EDA)** is an approach that is used to analyze the data and discover trends, patterns, or check assumptions in data with the help of statistical summaries and graphical representations.

Python supports us with many libraries to do the EDA such as: Autoviz, Sweetviz, Pandas,...Let's get a quick summary of the dataset using the Pandas **describe()** method. The **describe()** function applies basic statistical computations on the dataset like extreme values, count of data points standard deviation, etc. Any missing value or NaN value is automatically skipped. **describe()** function gives a good picture of the distribution of data.



	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

Figure 7: Describe Method in Pandas

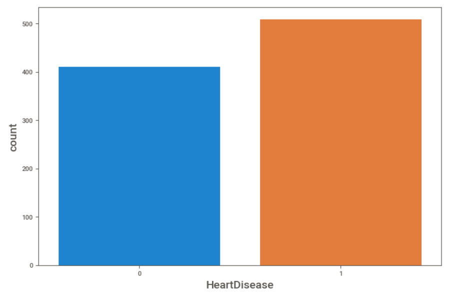


Figure 8: Allocation of the main feature - Heart Disease

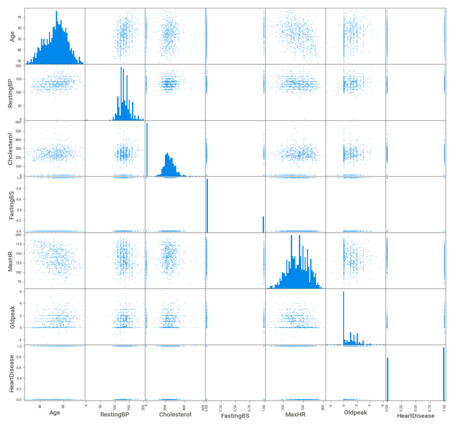
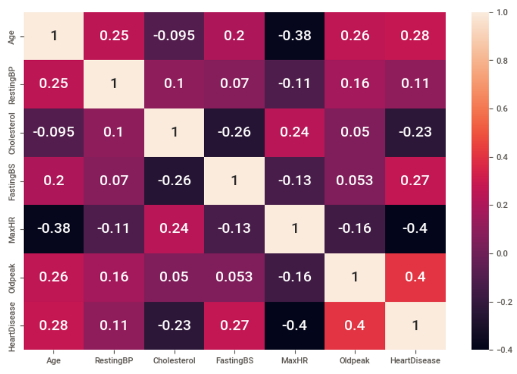


Figure 9: Scatter Matrix - Pair Plot

Next, we will visualize the allocation of all the features one by one with **matplotlib** - a popular data visualization library in Python. It provides a wide range of functions and tools for creating various types of plots, charts, and graphs. Matplotlib allows you to create high-quality visualizations to explore and present data in a clear and effective manner. Here is the example:

Another way, using **the scatter matrix** also known as a scatter plot matrix or pair plot, is a grid of scatter plots that displays the pairwise relationships between multiple variables. It allows for the examination of the correlations and distributions between pairs of variables.

The final step is using **the correlation matrix** to displays the correlation coefficients between multiple variables in our dataset. Each cell in the matrix represents the correlation between two variables, indicating the strength and direction of their linear relationship.



	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
Age	1	0.25	-0.095	0.2	-0.38	0.26	0.28
RestingBP	0.25	1	0.1	0.07	-0.11	0.16	0.11
Cholesterol	-0.095	0.1	1	-0.26	0.24	0.05	-0.23
FastingBS	0.2	0.07	-0.26	1	-0.13	0.053	0.27
MaxHR	-0.38	-0.11	0.24	-0.13	1	-0.16	-0.4
Oldpeak	0.26	0.16	0.05	0.053	-0.16	1	0.4
HeartDisease	0.28	0.11	-0.23	0.27	-0.4	0.4	1

Figure 10: Correlation Matrix

Bonus: we use **Sweetviz** - another way to visualize the data by converting all the features's charts in HTML file. This file will be located in our Colab's code.

To sum up from those methods, we can see that:

- It is observed from the histogram that the range from 55-65 people is highly getting heart disease.
- It shows the male gender has more records than the female and the male has a greater chance of heart attack when compared to the total numbers of people
- The results indicate chest pain ASY; patients are higher as compared to other possible types.

- People with high resting blood pressure and fasting blood glucose have more percentage of getting heart disease.People with low cholesterol at risk of having heart disease.
- From statistic, we can see that normal people can easily get heart disease than other with Resting ECG.
- People with low Maximum Heart Rate can get Heart Disease more frequency than people with high MaxHR.
- People with exercise-induced and flat can get Heart Disease while people from 0-2 St Depression can avoid getting heart Disease.

4. Evaluation Metric

If the values are close to each other, the set can be said to be precise. If their average is close to the true value of the quantity being measured, the set can be said to be accurate. Only if given a set of data points from repeated measurements of the same quantity then one can measure above two terms [13].

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (10)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (11)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (12)$$

$$\text{F1-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (13)$$

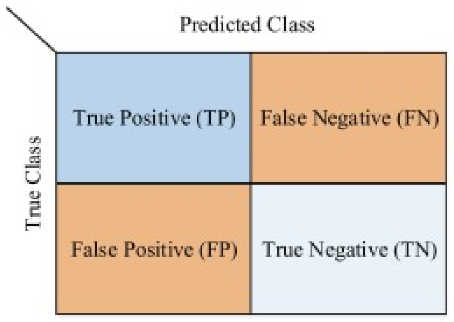


Figure 11: : Confusion Matrix sample

TP = True positive, TN = True Negative
FP = False Positive, FN = False Negative

5. Hyperparameters tuning

a) Decision tree: We can see that increase max-depth number can capture more intricate relationships but it may lead to overfitting.So we choose range from 0-20 for max depth.

In decision tree, we use two common criterion are *gini* for the Gini impurity and *entropy* for information gain. The dataset is almost balance so we do not need class weight. From the dataset, we tune main parameters max depth, criterion, min samples split, max leaf nodes.From my experiences of tunning, the range of min samples split fit the dataset well is 2-50.

From figure 12, we can choose range for tunning of max leaf nodes is 2-10 Then We choose optimal parameters by using

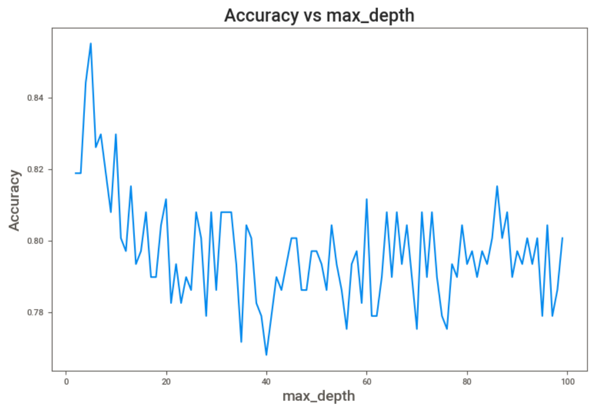


Figure 12: Max-depth Tunning

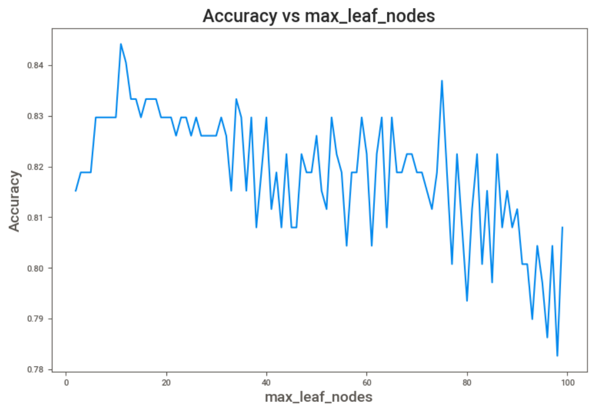


Figure 13: Max leaf nodes Tunning

method *GridSearchCV* , *RandomSearchCV* and *Bayes Optimization* [12]

b) SVM:

We can search and select the optimal values the optimal values for the hyperparameters of the SVM model. There are some important hyperparameters in SVM, such as the C parameter and the kernel parameter. The C parameter determines how ”soft” the decision boundary is in the SVM. After adjusting parameter C, we have the following results:

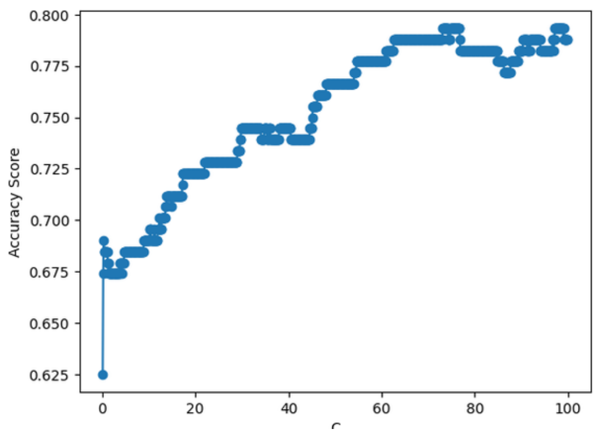


Figure 14: Accuracy with C

c) KNN: Similar to Decision Tree's Method, we will tune main parameters of KNN like n_neighbors, weights, algorithm, p, metric. Once the tuning has done, we will get the best group of these parameters.

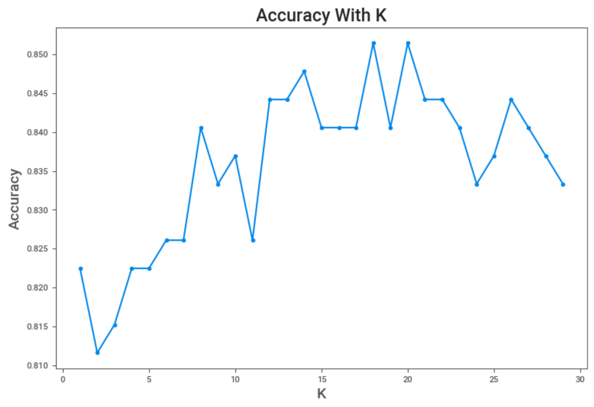


Figure 15: Accuracy with K

d) Random Forest: Hyperparameter tuning is about finding a set of optimal hyperparameter values which maximizes the model's performance, minimizes loss and produces better outputs.

Below is the list of the most important parameters of Random Forest:

- max-depth: The maximum depth of the tree - meaning the longest path between the root node and the leaf node.

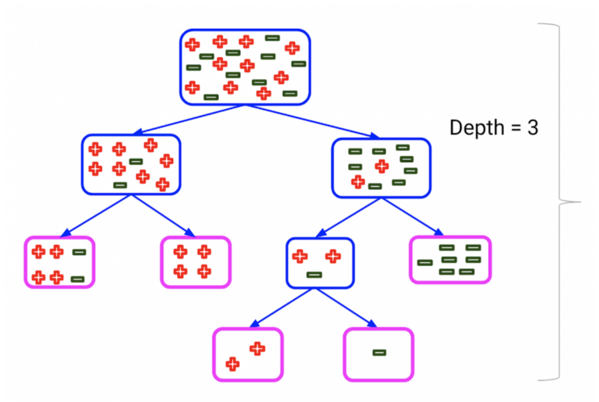


Figure 16: Max-depth

Using the max-depth parameter, we can limit up to what depth we want every tree in random forest to grow.

In this graph, we can clearly see that as the max depth of the decision tree increases, the performance of the model over the training set increases continuously. On the other hand as the max-depth value increases, the performance over the test set increases initially but after a certain point, it starts to decrease rapidly.

The reason is the tree starts to overfit the training set and therefore is not able to generalize over the unseen points in the test set.

So, we should set value of max-depth in moderate range to avoid overfitting.

- min-samples-split: The minimum number of samples required to split an internal node where the default = 2. By increasing the value of the min-sample-split, we can reduce the number of splits that happen in the decision tree and therefore prevent the model from overfitting.

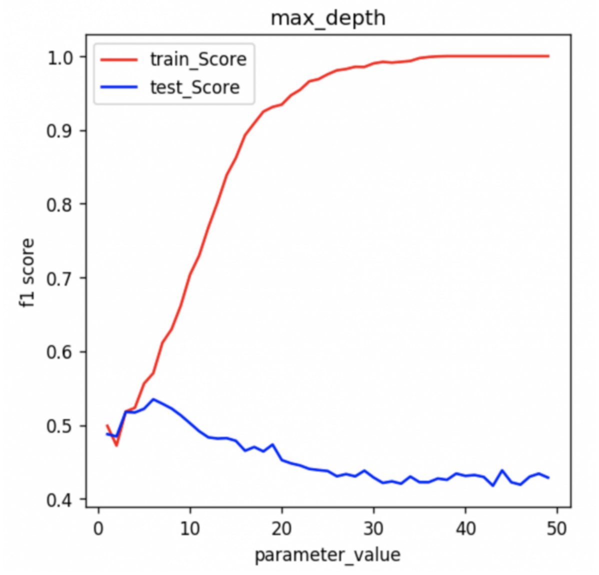


Figure 17: Max-depth vs f1-score

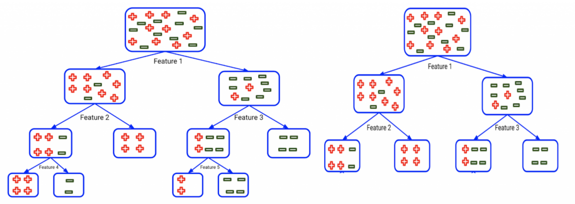


Figure 18: Min-samples-split

On increasing the value of the min-sample-split hyperparameter, we can clearly see that for the small value of parameters, there is a significant difference between the training score and the test scores. But as the value of the parameter increases, the difference between the train score and the test score decreases.

But, when the parameter value increases too much, there is an overall dip in both the training score and test scores. This is due to the fact that the minimum requirement of splitting a node is so high that there are no significant splits observed. As a result, the random forest starts to underfit.

- min-samples-leaf: This is the minimum number of samples required to be at a leaf node where the default = 1.

Similar to the two hyperparameters mentioned above, this hyperparameter also helps prevent overfitting as the parameter value increases.

We can clearly see that the Random Forest model is overfitting when the parameter value is very low (when parameter value < 100), but the model performance quickly rises up and rectifies the issue of overfitting ($100 <$ parameter value < 400). But when we keep on increasing the value of the parameter (> 500), the model slowly drifts towards the realm of underfitting.

So far, we have looked at the hyperparameters that are also covered in Decision Trees. Let's now look at the hy-

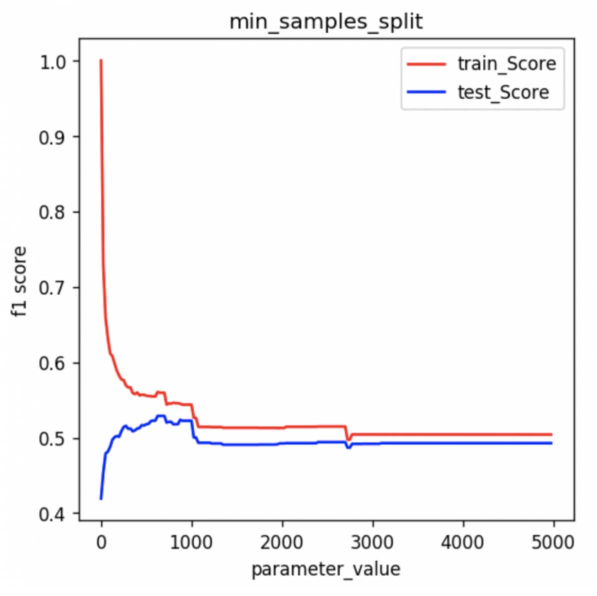


Figure 19: Min-samples-split vs f1-score

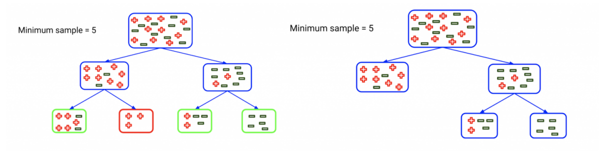


Figure 20: Min-samples-leaf

perparameters that are exclusive to Random Forest. Since Random Forest is a collection of decision trees, let's begin with the number of estimators.

- n-estimators: This is the number of trees in the forest. We might say that more trees should be able to produce a more generalized result, right? But by choosing more number of trees, the time complexity of the Random Forest model also increases. In graph below, we can clearly see that the performance of the model sharply increases and then stagnates at a certain level:

  This means that choosing a large number of estimators in a random forest model is not the best idea. Although it will not degrade the model, it can save you the computational complexity and CPU.

- max-features: This is the number of features to consider when looking for the best split. Finally, we will observe the effect of the max-features hyperparameter. This resembles the number of maximum features provided to each tree in a random forest.

  We can see that the performance of the model initially increases as the number of max-feature increases. However, after a certain point, the train-score keeps on increasing. But the test-score saturates and even starts decreasing towards the end, which clearly means that the model starts to overfit. So, we should choose moderate max-features value to get the highest test-score.

After analyzing several hyperparameters, we determine that using Grid Search CV is the best approach to find the optimal

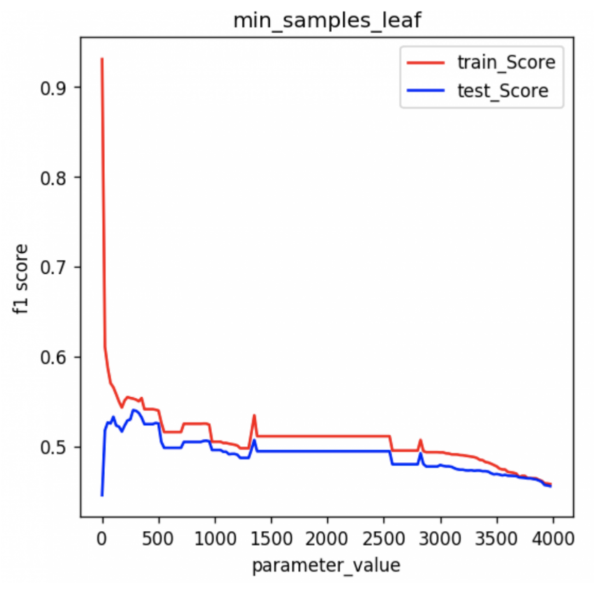


Figure 21: Min-samples-leaf vs f1-score

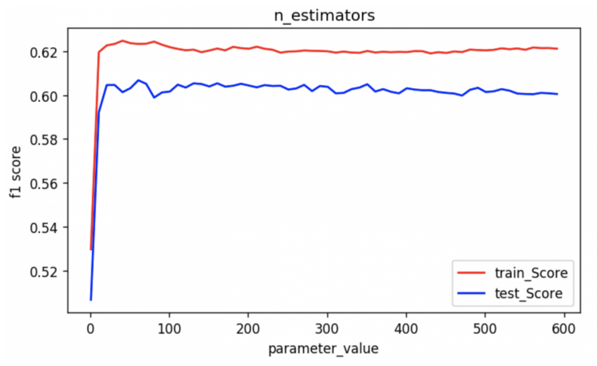


Figure 22: n-estimators vs f1-score

estimator for the model. The table below shows the results before tuning (Model 1) and after tuning (Model 2).

Model	max-depth	min-samples-split	min-samples-leaf	n-estimators	max-features	metric
1	None	2	1	100	sqrt	88%
2	10	2	1	100	log2	88%

Table 2: The result of Random Forest tuning

The results indicate that after tuning the hyperparameters, the model performs better (approximately 4% improvement).So, Grid Search CV has found a more suitable and optimized set of parameters for this dataset.

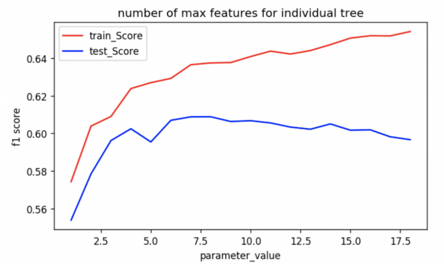


Figure 23: Max-features vs f1-score

6. Comparision

Decision tree:

Model	criterion	max depth	max features	max leaf nodes	min samples split	metric
1	Gini	None	None	None	2	80%
2	Gini	5	Sqrt	10	39	82% ↑
3	Gini	56	None	11	3	84% ↑

Table 3: Table shows the result in Decision Tree

SVM:

Model	C	kernel	Metric
1	1.0	linear	87%
2	1.0	rbf	86% ↓
3	1.0	sigmoid	77% ↓
4	1.0	poly	85% ↓
5	0.1	rbf	81% ↓

Table 4: Table shows the result in SVM

KNN:

Model	algorithm	metric	n neighbors	p	weights	metric
1	auto	minkowski	6	2	uniform	82%
2	auto	manhattan	12	1	distance	85% ↑

Table 5: Table shows the result in KNN

Random Forest:

Model	max-depth	min-samples-split	min-samples-leaf	n-estimators	max-features	metric
1	None	2	1	100	sqrt	88.04%
2	10	2	1	100	log2	88.4% ↑

Table 6: Table shows the result in Random Forest

Comparision:

Model	Student	metric
Decision Tree	Pham Duc Toan	84.04%
SVM	Le Quoc Trung	87%
KNN	Nguyen Duong Quoc Anh	85%
Random Forest	Tran Nhat Anh	88.5%

Table 7: Table shows the best result

V. Conclusion

Based on the survey conducted in the paper, four machine learning models were analyzed for heart disease detection: Decision Tree, K-Nearest Neighbor (K-NN), Support Vector Machine (SVM), and Random Forest. Among these models, the Random Forest achieved the highest metric with 88%, surpassing the metrics of SVM (87%), KNN (85%), and Decision Tree (84%).

While the Random Forest model performed the best in terms of overall metric, it is important to consider additional factors when evaluating the models. In this case, the paper highlights that SVM has the highest value with recall, indicating its ability to correctly identify positive cases for heart disease. Furthermore, the accuracy of SVM is reported to be closely comparable to that of the Random Forest model.

Considering the stability of the models, the paper concludes that SVM appears to be more stable compared to the other models. This suggests that SVM consistently performs well in terms of recall and maintains a high accuracy rate, making it a reliable choice for heart disease detection.

It is worth noting that the conclusion is based on the metrics and observations presented in the paper. Further analysis and experimentation may be required to validate and corroborate these findings in real-world scenarios.

Acknowledgement

We would like to express our gratitude to Ph.D. Tiep Nguyen Vinh of the University of Information Technology for instructing and guiding the project implementation team

References

[1] https://en.wikipedia.org/wiki/Health_care

[2] Lee, S. J., & Siau, K. (2001). "A review of data mining techniques". *Industrial Management & Data Systems*, *101*(1), 41–46.

[3] Gelfand, A. E., & Professor1. "Hierarchical Modeling for Spatial Data Problems". *Journal Name*, *Volume*(Issue), Page numbers.

[4] RekhaMolala. "Entropy, Information gain and Gini Index; the crux of a Decision Tree." *Medium*, Mar. 23, 2020.

[5] V. Cheushev, D. A. Simovici, V. Shmerko, and S. Yanushkevich. "Functional entropy and decision trees." In *Proceedings. 1998 28th IEEE International Symposium on Multiple-Valued Logic (Cat. No. 98CB36138)*, 1998, pp. 257–262.

[6] X. Chen, Z. Yang, and W. Lou. "Fault Diagnosis of Rolling Bearing Based on the Permutation Entropy of VMD and Decision Tree." In *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*, Xiamen, China, Oct. 2019, pp. 1911–1915. doi: 10.1109/EITCE47263.2019.9095187.

[7] C. Shang, M. Li, S. Feng, Q. Jiang, and J. Fan. "Feature selection via maximizing global information gain for text classification." *Knowledge-Based Systems*, vol. 54, pp. 298–309, Dec. 2013. doi: 10.1016/j.knosys.2013.09.019.

[8] L. E. Raileanu and K. Stoffel. "Theoretical Comparison between the Gini Index and Information Gain

Criteria." *Annals of Mathematics and Artificial Intelligence*, vol. 41, no. 1, pp. 77–93, May 2004. doi: 10.1023/B:AMAI.0000018580.96245.c6.

[9] Y. Liu, L. Hu, F. Yan, and B. Zhang. "Information Gain with Weight Based Decision Tree for the Employment Forecasting of Undergraduates." In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, Beijing, China, Aug. 2013, pp. 2210–2213. doi: 10.1109/GreenCom-iThingsCPSCom.2013.417.

[10] R. L. De Mántaras. "A distance-based attribute selection measure for decision tree induction." *Machine Learning*, vol. 6, no. 1, pp. 81–92, 1991.

[11] S. Taneja, C. Gupta, K. Goyal, and D. Gureja. "An enhanced k-nearest neighbor algorithm using information gain and clustering." In *2014 Fourth International Conference on Advanced Computing & Communication Technologies*, 2014, pp. 325–329.

[12] Zahedi, L., Mohammadi, F. G., Rezapour, S., Ohland, M. W., & Amini, M. H. (Year). Title of the Paper. *Journal Name*, Volume(Issue), Page Range.

[13] Piryonesi S. Madeh; El-Diraby Tamer E. (2020-06-01). "Role of Data Analytics in Infrastructure Asset Management: Overcoming Data Size and Quality Problems". Journal of Transportation Engineering, Part B: Pavements.

[14] Piryonesi, S. Madeh; El-Diraby, Tamer E. (2021-02-01). "Using Machine Learning to Examine Impact of Type of Performance Indicator on Flexible Pavement Deterioration Modeling". Journal of Infrastructure Systems.

[15] https://en.wikipedia.org/wiki/Bootstrap_aggregating

[16] https://en.wikipedia.org/wiki/Random_subspace_method

[17] Lin, Yi; Jeon, Yongho (2002). Random forests and adaptive nearest neighbors (Technical report). Technical Report No. 1055. University of Wisconsin.

[18] https://scikitlearn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[19] https://towardsdatascience.com/knn-algorithm-what-when-why-how-41405c16c36f

[20] Antony Christopher. "K-Nearest Neighbor". https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4

[21] : K-Nearest Neighbors (KNN) Algorithm. https://www.geeksforgeeks.org/k-nearest-neighbours/

[22] AlindGupta. "ML—Implementation of KNN classifier using Sklearn". https://www.geeksforgeeks.org/ml-implementation-of-knn-classifier-using-sklearn/

[23] Nick McCullum. "How to Build and Train K-Nearest Neighbors and K-Means Clustering ML Models in Python". https://www.freecodecamp.org/news/how-to-build-and-train-k-nearest-neighbors-ml-models-in-python/