

**Project Report**  
**On**

**HEART DISEASE PREDICTION USING  
K-NEAREST NEIGHBORS AND  
DECISION TREE ALGORITHMS**

*Submitted*

*In partial fulfillment of the requirements for the degree of*

**BACHELOR OF SCIENCE**

*in*

**MATHEMATICS**

*By*

**SWETLANA T J**

**(Register No. AB22BMAT005)**

*Under the supervision of*

**DR. SWATHI V V**



**DEPARTMENT OF MATHEMATICS**

**ST. TERESA'S COLLEGE (AUTONOMOUS) ERNAKULAM,  
KOCHI – 682011**

**APRIL 2025**

**ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM**



**CERTIFICATE**

This is to certify that the dissertation entitled, **HEART DISEASE PREDICTION USING K-NEAREST NEIGHBORS AND DECISION TREE ALGORITHMS** is a bonafide record of the work done by **SWETLANA T J** under my guidance as partial fulfillment of the award of the degree of **Bachelor of Science in Mathematics** at St. Teresa's College (Autonomous), Ernakulam affiliated to Mahatma Gandhi University, Kottayam. No part of this work has been submitted for any other degree elsewhere.

Place: Ernakulam

Date: 17-02-2025

*for* *Swathi*  
**Dr. Swathi V V**

Assistant professor,  
Department of Mathematics,  
St. Teresa's College (Autonomous),  
Ernakulam

*Elizabeth*

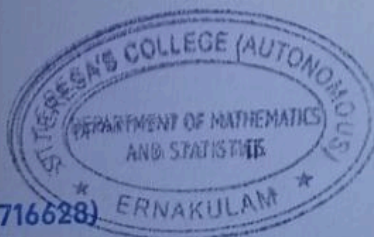
**Dr. Elizabeth Reshma M T**

Assistant Professor and Head,  
Department of Mathematics,  
St. Teresa's College (Autonomous),  
Ernakulam

**External examiners**

1: *Sreeja*  
*30/4/25*

2: .....



**Dr. SREEJA K.U. (PEN:716628)**  
Assistant Professor  
Department of Mathematics  
Maharaja's College  
Ernakulam - 682 011



## DECLARATION

I hereby declare that the work presented in this project is based on the original work done by me under the guidance of Dr. SWATHI V V, Assistant Professor, Department of Mathematics , St. Teresa's College (Autonomous), Ernakulam and have not been included in any other project submitted previously for the award of any degree.

Ernakulam

Date:17-02-2025



SWETLANA T J(AB22BMAT005)

## ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude towards Dr. SWATHI V V of the Department of Mathematics of St. Teresa's College who encouraged me to carry this work. Her continuous invaluable knowledgeable guidance throughout this study helped me complete the work upto this stage. I will always be thankful to you in this regard. I also express my thanks to all those who have indirectly guided and helped me in the completion of this project.

Ernakulam

Date:17-02-2025

SWETLANA T J(AB22BMAT005)

# CONTENTS

<i>CERTIFICATE</i> .....	ii
<i>DECLARATION</i> .....	iii
<i>ACKNOWLEDGEMENT</i> .....	iv
<i>CONTENT</i> .....	v
<b>Chapter 1- Introduction to Machine learning</b>	<b>1</b>
1.1 Features of Machine learning. ....	1
1.2 Importance of Machine learning. ....	2
1.3 Classification of Machine Learning. ....	2
Supervised Machine Learning	
Unsupervised Machine Learning	
Reinforcement Machine Learning	
<b>Chapter 2- K-Nearest Neighbor algorithm (KNN)</b>	<b>5</b>
2.1 Why do we need a KNN algorithm?. ....	5
2.2 How does KNN works?. ....	6
2.3 Distance metrics used in KNN algorithm ....	7
2.4 How to select the value of K in the KNN algorithm? ....	8
2.5 Advantages of KNN algorithm ....	8
2.6 Disadvantages of KNN algorithm ....	9
<b>Chapter 3- Decision tree algorithm</b>	<b>10</b>
3.1 Why we use Decision Trees? ....	10
3.2 Decision Tree terminologies ....	11

3.3	How does the Decision Tree works? . . . . .	12
3.4	Advantages of the Decision Tree . . . . .	15
3.5	Disadvantages of the Decision Tree . . . . .	15

## **Chapter 4- Performance evaluation of classification algorithms** **16**

4.1	Confusion matrix. . . . .	16
4.2	Metrics based on Confusion matrix data. . . . .	17
4.2.1	Accuracy	
4.2.2	Precision	
4.2.3	Recall	
4.2.5	F1-Score	
4.2.6	Specificity	
4.3	Type 1 and Type 2 error. . . . .	18
4.3.1	Type 1 error	
4.3.2	Type 2 error	
4.4	Classification report? . . . . .	19

## **Chapter 5- Heart disease prediction using the algorithms** **20**

5.1	Heart disease prediction using K-Nearest Neighbor algorithm. . . . .	21
5.2	Heart disease prediction using Decision Tree algorithm . . . . .	25

<i>CONCLUSION</i> .....	30
-------------------------	----

<i>REFERENCES</i> .....	31
-------------------------	----

## **Chapter 1**

### **INTRODUCTION TO MACHINE LEARNING**

Machine learning is a developing technology which enables computers to authorize automatically from past data. It uses several algorithms for building mathematical models and making predictions using historical data or information. Machine learning is said as a subset of artificial intelligence that is mainly concerned with the growing of algorithms which allow a computer to learn from data and past experiences on their own. The term machine learning was first introduced by Arthur Samuel in 1959.

Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed. With the aid of sample historical data, which is called as training data, machine learning algorithms construct a mathematical model that helps in making predictions or decisions without being explicitly programmed. Machine learning brings computer science and statistics together for creating predictive models. The more we will provide the information, the higher will be the performance.

#### **1.1 Features of Machine learning**

Machine learning uses data to detect various patterns in a given dataset to improve accuracy and efficiency, and to make better decisions.

It can learn from past data and improve automatically.

It is a data-driven technology because it uses data to make predictions and decisions.

Machine learning is much similar to data mining as it also deals with the huge amount of data.

## **1.2 Importance of Machine learning**

Machine learning can analyze vast amount of data to make predictions and automate decisions, saving time and resources in various application like fraud detection, customer services and product recommendation.

By identifying patterns and trends hidden in large datasets, Machine learning can provide valuable insights that human might miss, enabling better understanding of complex systems.

Machine learning models can continuously improve their performance over time by learning from new data, allowing them to adapt to changing conditions.

It enables the development of new technologies and applications across diverse Industries, including healthcare, finance, manufacturing and transportation.

## **1.3 Classification of Machine learning**

At a broad level, machine learning can be classified into three types:

- 1)Supervised Machine learning
- 2)Unsupervised machine learning
- 3)Reinforcement learning

### **Supervised Machine learning**

Supervised Machine learning is a fundamental approach for Machine learning and artificial intelligence. It is a kind of Machine learning method in which we give sample labeled data to the Machine learning system in order to teach it, and on that basis, it predicts the output. The system generates a model using labeled data to understand the datasets and learn about each data, once the training and processing are done then we test the model by providing a sample data to check whether it is predicting the exact output or not. The goal of supervised learning is to map input data with the output data. The supervised learning is based on supervision, and it is the same as when a student learns things in the supervision of the teacher.

Supervised Machine learning can be grouped into two categories of algorithms: Classification and Regression. Classification algorithms are only used for data that is



discrete. It helps in mapping the input value with the output variable. These algorithms are useful for problems like identifying spam emails, recognition of speech etc. In Regression algorithms, the data is continuous. It helps to map the input value and the continuous output variable. Regression algorithms help in solving the problems like predicting the weather, estimating housing cost, predicting population growth, predicting stock price etc.

Supervised Machine learning has many real world applications such as natural language processing, finance, health detection, cyber security, predictive analysis, sentimental analysis etc.

### **Unsupervised Machine learning**

Unsupervised Machine learning is a learning method in which a machine learns without any supervision. The training is provided to the machine with the set of data that has not been labeled, classified or categorized and the algorithm needs to act on that data without any supervision. The goal of Unsupervised Machine learning is to restructure the input data into new features or a group of objects with similar patterns. In unsupervised learning, we don't have a predetermined result. The machine tries to find useful insights from the huge amount of data.

Unsupervised Machine learning can be classified into two categories of algorithms: Clustering and Association. Clustering focuses on similar data points into clusters based on their features, while Association aims to discover relationships and patterns between different variables within a dataset.

In real world, we do not always have input data with the corresponding output. So to solve such cases we need unsupervised Machine learning. It has many applications such as anomaly detection, customer segmentation, genetic research, clinical cancer studies etc.

**Reinforcement learning**

Reinforcement learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each right action, the agent gets a reward and for each wrong action, the agent gets penalty. The agent learns automatically using feedbacks without any labeled data. It is a core part of Artificial Intelligence and all AI agent works on the concept of Reinforcement learning. Here we do not need to pre-program the agent, as it learns from its own experience without any human intervention.

Reinforcement learning is the most common algorithm used to solve games. It is used to optimize the trajectory, motion and paths of robots and used to predict how people speak by studying language patterns. Reinforcement learning is also used for healthcare, to control traffic light, autonomous driving, education, finance, marketing etc.

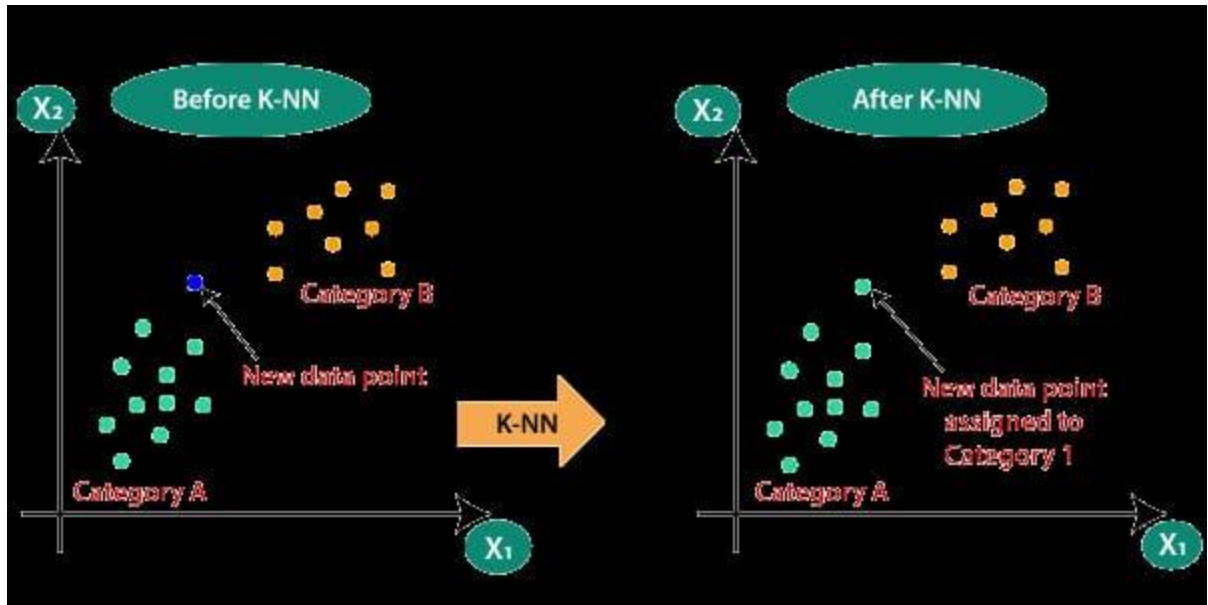
## **Chapter 2**

### **K-NEAREST NEIGHBORS ALGORITHM (KNN)**

K-Nearest Neighbors is one of the simplest Machine learning algorithms based on Supervised Learning technique. KNN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. This means when new data appears then it can be easily classified into a well suite category by using KNN algorithm. KNN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. KNN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

#### **2.1 Why do we need a KNN algorithm?**

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point  $x_1$ , so this data point will lie in which of these categories. To solve this type of problem, we need a KNN algorithm. With the help of KNN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



## 2.2 How does KNN works?

The KNN working can be explained on the basis of the below algorithm:

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of K number of neighbors

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: Our model is ready.

### 2.3 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 between  $A_1$  and  $B_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

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

$$\text{Distance}(x, X_i) = \sum_{j=1}^d (x_j - X_{ij})^2$$

By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:





As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

#### 2.4 How to select the value of K in the KNN algorithm?

There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5. A very low value for K such as  $K=1$  or  $K=2$ , can be noisy and lead to the effects of outliers in the model. Large values for K are good, but it may find some difficulties.

#### 2.5 Advantages of KNN algorithm

Easy to implement- The KNN algorithm is easy to implement because its complexity is relatively low as compared to other Machine learning algorithms.

Easily adaptable- KNN stores all data in memory, so when new data points are added, it automatically adjusts and uses the new data for future predictions.

Few hyperparameters- The only parameters which are required in the training of a KNN algorithm are the value of K and the choice of the distance metric which we would like to choose from our evaluation metric.

## **2.6 Disadvantages of KNN algorithm**

Doesn't scale well- KNN considered a “lazy” algorithm, meaning it requires a lot of computing power and memory. This makes it slow, especially with large datasets.

Always needs to determine the value of K which may be complex some time.

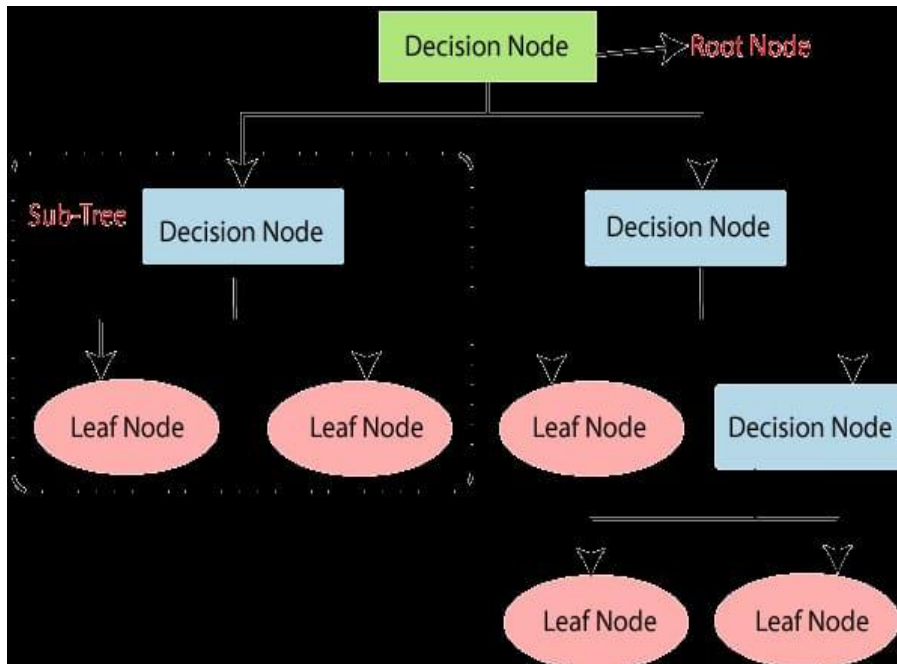
Curse of dimensionality- When the number of features increases, KNN struggles to classify data accurately, a problem known as curse of dimensionality, which implies the algorithm faces a hard time classifying the data points properly when the dimensionality is too high.

Prone to overfitting- As the algorithm is affected due to the curse of dimensionality it is prone to the problem of overfitting as well.

## Chapter 3

### DECISION TREE ALGORITHM

Decision Tree is a Supervised Machine learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset. **It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.** It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees. Below diagram explains the general structure of a decision tree:



Note: A decision tree can contain categorical data (YES/NO) as well as numeric data.

### 3.1 Why we use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.

The logic behind the decision tree can be easily understood because it shows a tree-like structure.

### 3.2 Decision Tree terminologies

**Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.

**Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

**Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

**Branch/Sub Tree:** A tree formed by splitting the tree.

**Pruning:** Pruning is the process of removing the unwanted branches from the tree.

**Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

### 3.3 How does the Decision Tree works?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step-3: Divide the S into subsets that contains possible values for the best attributes.

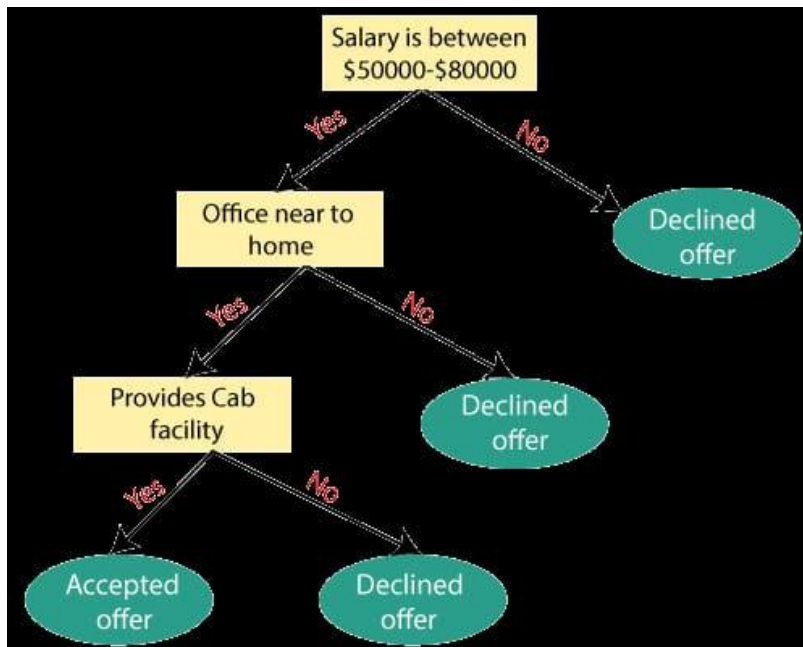
Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

**Example:** Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and



one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



### Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as Attribute selection measure or ASM. By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- Information gain
- Gini index

#### 1. Information gain:

Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute. It calculates how much information a feature provides us about a class. According to the value of information gain, we split the node and build the decision tree. A decision tree algorithm always tries to maximize the value of information

gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

$$\text{Information gain} = \text{Entropy}(S) - [(\text{Weighted avg}) * \text{Entropy (each feature)}]$$

**Entropy:** Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

$$\text{Entropy}(s) = -P(\text{yes}) \log_2 P(\text{yes}) - P(\text{no}) \log_2 P(\text{no})$$

Where,

S = Total number of samples

P(yes) = probability of yes

P(no) = probability of no

## 2. Gini Index:

Gini index is a measure of impurity or purity used while creating a decision tree in the CART (Classification and Regression Tree) algorithm. An attribute with the low Gini index should be preferred as compared to the high Gini index. It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.

Gini index can be calculated using the below formula:

$$\text{Gini index} = 1 - \sum_j P_j^2$$

## Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree. A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning.

There are mainly two types of tree **pruning** technology used:

- Cost Complexity Pruning
- Reduced Error Pruning.

### 3.4 Advantages of the Decision Tree

Simplicity and interpretability- It is easy to understand as it follows the same process which a human follow while making any decision in real-life. Decision Trees are straightforward and you can visualize them like a flowchart which makes it simple to see how decisions are made.

It can be very useful for solving decision-related problems.

It helps to think about all the possible outcomes for a problem.

There is less requirement of data cleaning compared to other algorithms.

No need for feature scaling- They don't require you to normalize or scale your data.

### 3.5 Disadvantages of the Decision Tree

The decision tree contains lots of layers, which makes it complex.

Overfitting issue- Overfitting occurs when a Decision Tree captures noise and details in the training data and it performs poorly on new data. It which can be resolved using the **Random Forest algorithm**.

For more class labels, the computational complexity of the decision tree may increase.

## Chapter 4

# PERFORMANCE EVALUATION OF CLASSIFICATION ALGORITHMS

A performance evaluation of classification algorithms on a specific dataset involves evaluating how well different classification algorithms perform when applied to a particular dataset, typically measured by metrics like accuracy, precision, recall, F1-score and specificity. By comparing different classification algorithms using performance metrics, we can choose the most suitable model for our specific dataset and problem. Analyzing performance metrics highlights areas where a model can be improved and provide a clear picture of their relative strengths and weakness. Accurate predictions from a well evaluated model enable better decision making in various domain.

### 4.1 Confusion matrix

A **confusion matrix** is a matrix that summarizes the performance of a machine learning model on a set of test data. It is a means of displaying the number of accurate and inaccurate instances based on the model's predictions. It is often used to measure the performance of classification models, which aim to predict a categorical label for each input instance.

The matrix displays the number of instances produced by the model on the test data.

**True Positive (TP):** The model correctly predicted a positive outcome (the actual outcome was positive).

**True Negative (TN):** The model correctly predicted a negative outcome (the actual outcome was negative).

**False Positive (FP):** The model incorrectly predicted a positive outcome (the actual outcome was negative). Also known as a Type I error.

**False Negative (FN):** The model incorrectly predicted a negative outcome (the actual outcome was positive). Also known as a Type II error.

### **Why do we need a Confusion matrix?**

When assessing a classification model's performance, a confusion matrix is essential. It offers a thorough analysis of true positive, true negative, false positive, and false negative predictions, facilitating a more profound comprehension of a model's **recall**, **accuracy**, **precision**, and overall effectiveness in class distinction. When there is an uneven class distribution in a dataset, this matrix is especially helpful in evaluating a model's performance beyond basic accuracy metrics.

## **4.2 Metrics based on Confusion matrix data**

### **4.2.1 Accuracy**

Accuracy is used to measure the performance of the model. It is the ratio of Total correct instances to the total instances.

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

### **4.2.2 Precision**

Precision is a measure of how accurate a model's positive predictions are. It is defined as the ratio of true positive predictions to the total number of positive predictions made by the model.

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

### **4.2.3 Recall**

Recall measures the effectiveness of a classification model in identifying all relevant instances from a dataset. It is the ratio of the number of true positive (TP) instances to the sum of true positive and false negative (FN) instances.

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



#### 4.2.4 F1-Score

F1-score is used to evaluate the overall performance of a classification model. It is the harmonic mean of precision and recall,

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

#### 4.2.5 Specificity

Specificity is another important metric in the evaluation of classification models, particularly in binary classification. It measures the ability of a model to correctly identify negative instances. Specificity is also known as the True Negative Rate. Formula is given by:

$$\text{Specificity} = \frac{TN}{TN + FP}$$

### 4.3 Type 1 and Type 2 error

#### 4.3.1 Type 1 error

Type 1 error occurs when the model predicts a positive instance, but it is actually negative. Precision is affected by false positives, as it is the ratio of true positives to the sum of true positives and false positives.

$$\text{Type 1 Error} = \frac{FP}{TN + FP}$$

For example, in a courtroom scenario, a Type 1 Error, often referred to as a false positive, occurs when the court mistakenly convicts an individual as guilty when, in truth, they are innocent of the alleged crime. This grave error can have profound consequences, leading to the wrongful punishment of an innocent person who did not commit the offense in question. Preventing Type 1 Errors in legal proceedings is paramount to ensuring that justice is accurately served and innocent individuals are protected from unwarranted harm and punishment.

#### 4.3.2 Type 2 error

Type 2 error occurs when the model fails to predict a positive instance. Recall is directly affected by false negatives, as it is the ratio of true positives to the sum of true positives and false negatives.

In the context of medical testing, a Type 2 Error, often known as a false negative, occurs when a diagnostic test fails to detect the presence of a disease in a patient who genuinely has it. The consequences of such an error are significant, as it may result in a delayed diagnosis and subsequent treatment.

$$\text{Type 2 Error} = \frac{FN}{TP + FN}$$

Precision emphasizes minimizing false positives, while recall focuses on minimizing false negatives .

#### **4.4 Classification Report**

As the name suggests, it is the report which explains everything about the classification. This is the summary of the quality of classification made by the constructed ML model. It comprises mainly 5 columns and (N+3) rows. The first column is the class label's name and followed by Precision, Recall, F1-score, and Support. N rows are for N class labels and other three rows are for accuracy, macro average, and weighted average.

## Chapter 5

### HEART DISEASE PREDICTION USING ALGORITHMS

In this chapter, we apply K-Nearest Neighbors and Decision Tree algorithm to a dataset to find out which algorithm is better. It gives the potential of Machine learning techniques in predictive analytics for healthcare, offering valuable insights for early diagnosis and effective management of heart disease.

The dataset consist details of 1025 rows and 14 columns. Here's a brief overview of the important columns:

age: Age of the patient.

sex: Gender (1=male, 0=female)

cp: Chest pain type (4 values)

trestbps: Resting blood pressure

chol: Serum cholesterol in mg/dl

fbs: Fasting blood sugar > 120 mg/dl (1=true, 0=false)

restecg: Resting electrocardiographic results

thalach: Maximum heart rate achieved

exang: Exercise-induced angina(1=yes, 0=no)

oldpeak: ST depression induced by exercise relative to rest.

slope: Slope of the peak exercise ST segment

ca: Number of major vessels (0-3) colored by fluoroscopy

thal: A blood disorder called thalassemia (1=normal, 2=fixed defect, 3=reversible defect)

target: Diagnosis of heart disease (1= presence, 0=absence)

## 5.1 Heart disease prediction using K-Nearest Neighbors algorithm

We are taking a clear dataset of patients and we are using the KNN algorithm to predict if they have the chance to get heart disease or not.

Steps to implement the KNN algorithm:

- Data Pre-processing step

- Fitting the KNN algorithm to the Training set

- Predicting the test result

### Data Pre-Processing Step:

```
# importing libraries
```

```
import numpy as nm
```

```
import pandas as pd
```

```
#importing datasets
```

```
data_set= pd.read_csv('/content/heart.csv')
```

```
[5] x=df.iloc[:, :-1].values
x
array([[52., 1., 0., ..., 2., 2., 3.],
       [53., 1., 0., ..., 0., 0., 3.],
       [70., 1., 0., ..., 0., 0., 3.],
       ...,
       [47., 1., 0., ..., 1., 1., 2.],
       [50., 0., 0., ..., 2., 0., 2.],
       [54., 1., 0., ..., 1., 1., 3.]])

[6] y=df.iloc[:, -1]
y
target
0      0
1      0
2      0
3      0
4      0
```

*#Extracting Independent and dependent Variable*

*x= data\_set.iloc[:, :-1].values*

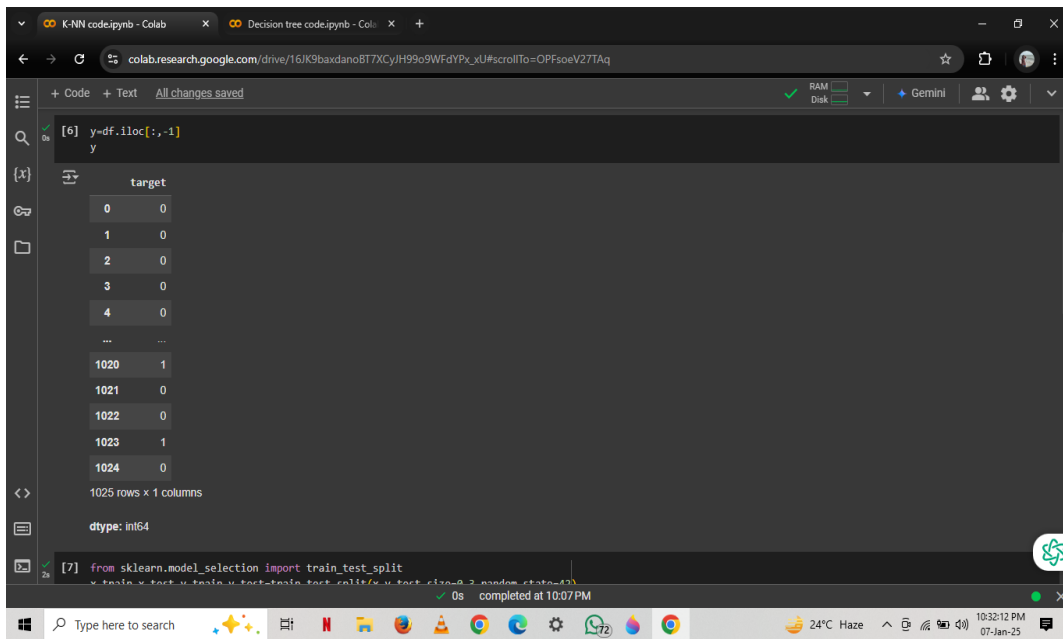
*x*

*y= data\_set.iloc[:, -1]*

*y*

```
[5] x=df.iloc[:, :-1].values
x
array([[52., 1., 0., ..., 2., 2., 3.],
       [53., 1., 0., ..., 0., 0., 3.],
       [70., 1., 0., ..., 0., 0., 3.],
       ...,
       [47., 1., 0., ..., 1., 1., 2.],
       [50., 0., 0., ..., 2., 0., 2.],
       [54., 1., 0., ..., 1., 1., 3.]])

[6] y=df.iloc[:, -1]
y
target
0      0
1      0
2      0
3      0
4      0
```



```
[6] y=df.iloc[:,1]
y

target
0      0
1      0
2      0
3      0
4      0
...    ...
1020    1
1021    0
1022    0
1023    1
1024    0
...    ...
1025 rows x 1 columns

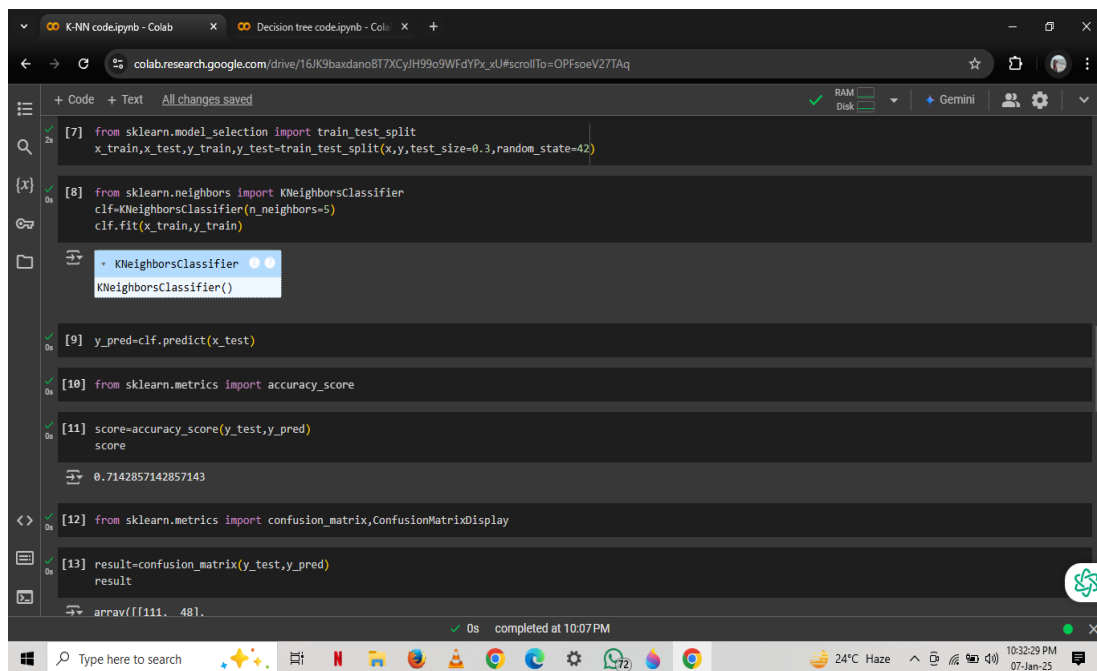
dtype: int64

[7] from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=42)
```

*# Splitting the dataset into training and test set.*

*from sklearn.model\_selection import train\_test\_split*

*x\_train, x\_test, y\_train, y\_test= train\_test\_split(x, y, test\_size= 0.3, random\_state=42)*



```
[7] from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=42)

[8] from sklearn.neighbors import KNeighborsClassifier
clf=KNeighborsClassifier(n_neighbors=5)
clf.fit(x_train,y_train)

KNeighborsClassifier()

[9] y_pred=clf.predict(x_test)

[10] from sklearn.metrics import accuracy_score

[11] score=accuracy_score(y_test,y_pred)
score
0.7142857142857143

[12] from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay

[13] result=confusion_matrix(y_test,y_pred)
result
array([[111, 48],
```

### Fitting KNN classifier to the Training data:

To do this we will import the KNeighborsClassifier class of Sklearn Neighbors library. After importing the class, we will create the Classifier object of the class. The Parameter of this class will be n\_neighbors: To define the required neighbors of the algorithm. Usually, it takes 5. And then we will fit the classifier to the training data. Below is the code for it:

```
#Fitting KNN classifier to the training set
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
classifier= KNeighborsClassifier(n_neighbors=5 )
```

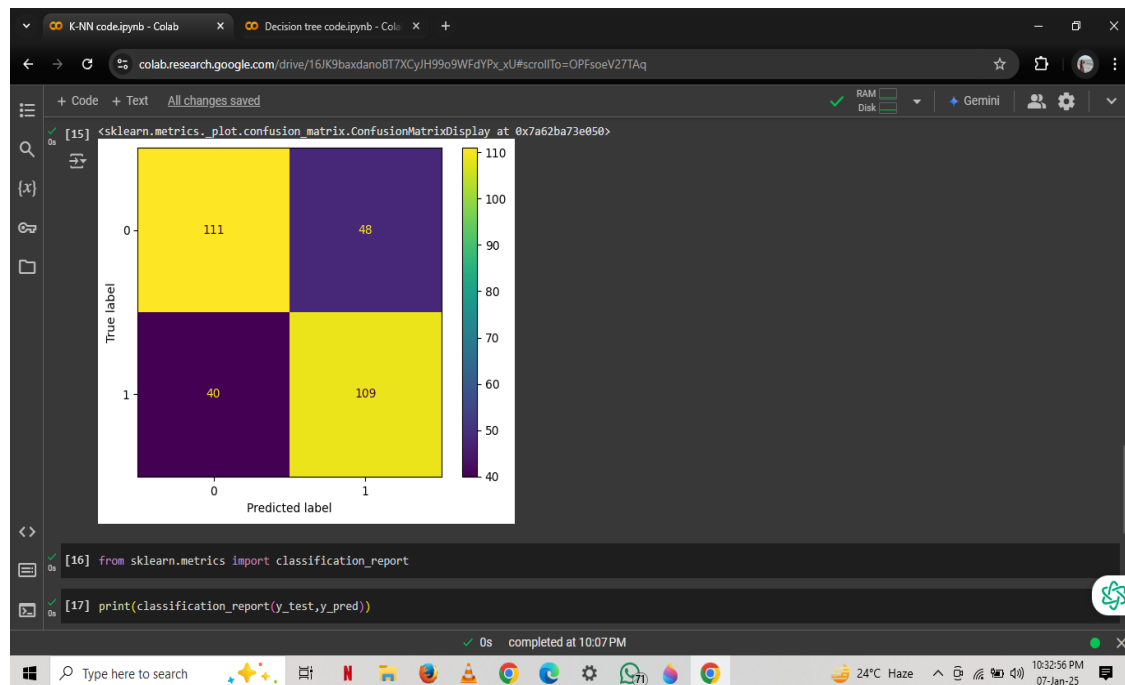
```
classifier.fit(x_train, y_train)
```

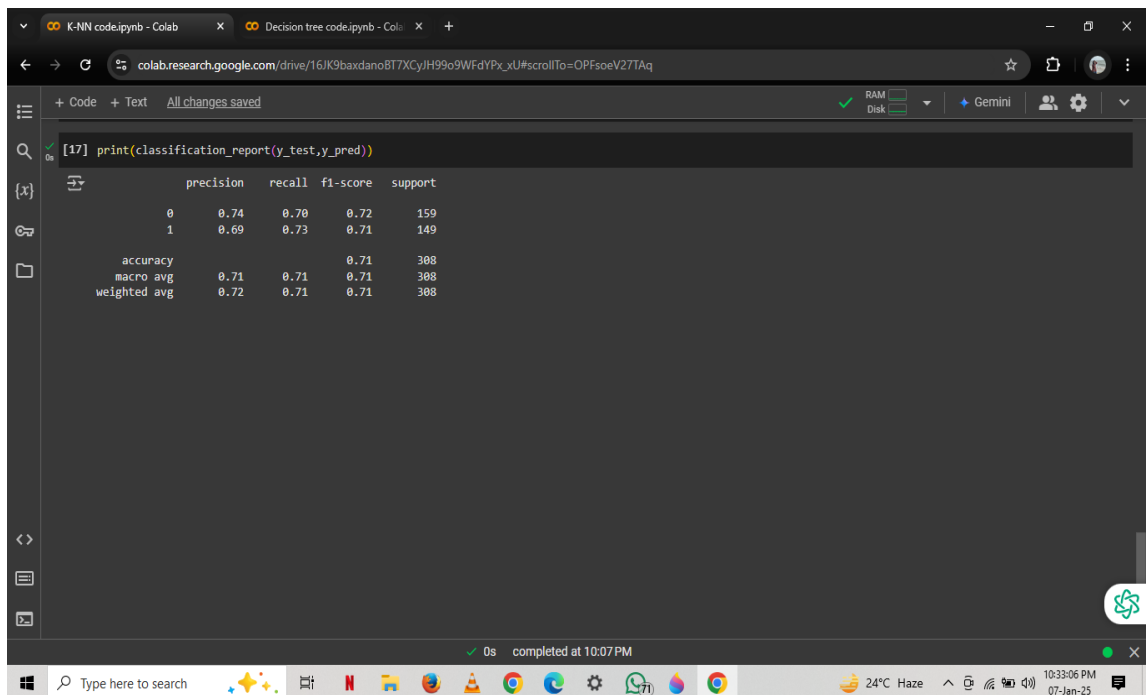
### Predicting the Test Result:

To predict the test set result, we will create a y\_pred vector, Below is the code for it:

```
#Predicting the test set result
```

```
y_pred= classifier.predict(x_test)
```





The screenshot shows a Google Colab notebook interface. The code cell contains the command `print(classification_report(y_test, y_pred))`. The output is a classification report for a binary classifier. The report shows precision, recall, f1-score, and support for both classes (0 and 1). The overall accuracy is 0.71, with a macro average and weighted average both at 0.71. The support for each class is 159.

	precision	recall	f1-score	support
0	0.74	0.70	0.72	159
1	0.69	0.73	0.71	149
accuracy			0.71	308
macro avg	0.71	0.71	0.71	308
weighted avg	0.72	0.71	0.71	308

## 5.2 Heart disease prediction using Decision Tree algorithm

Similarly, we can implement **Decision Tree Algorithm** in this data;

Steps will also remain the same, which are given below:

Data Pre-processing step

Fitting a Decision-Tree algorithm to the Training set

Predicting the test result

### Data Pre-Processing Step:

Below is the code for the pre-processing step:

*# importing libraries*

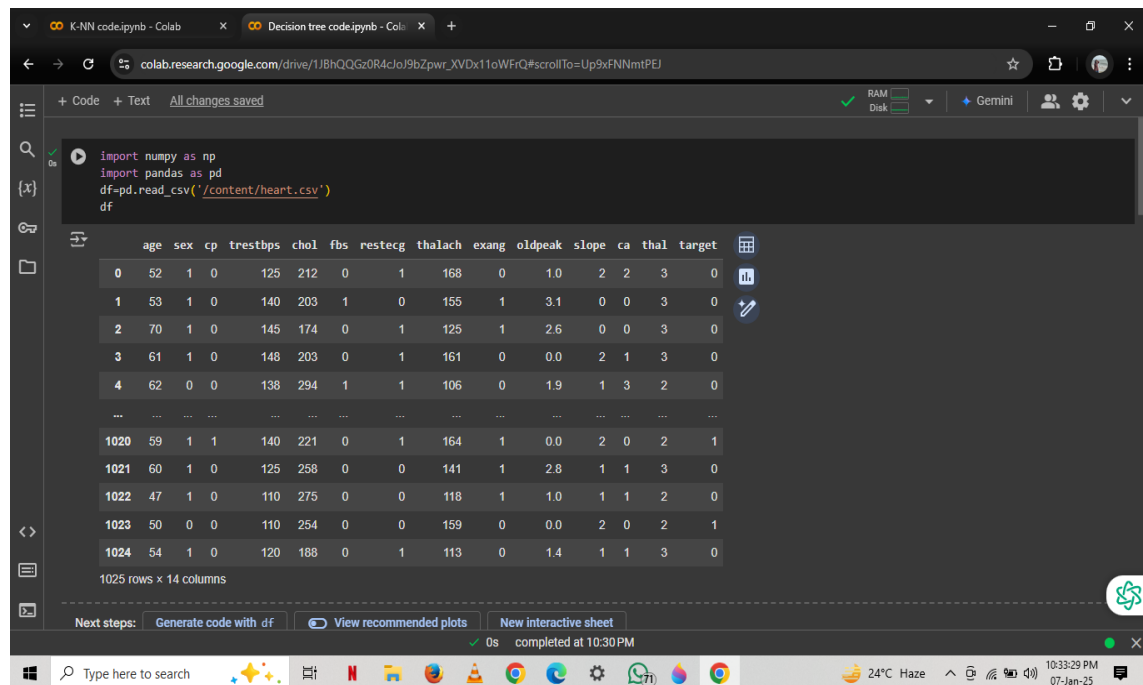
*import numpy as nm*

*import pandas as pd*



*#importing datasets*

*data\_set= pd.read\_csv('/content/heart.csv')*



The screenshot shows a Google Colab notebook with the following code executed:

```
import numpy as np
import pandas as pd
df=pd.read_csv('/content/heart.csv')
df
```

The output displays a preview of the dataset with columns: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal, target. The first few rows are:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

... (rows omitted) ...

1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	0
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	1
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	0

1025 rows x 14 columns

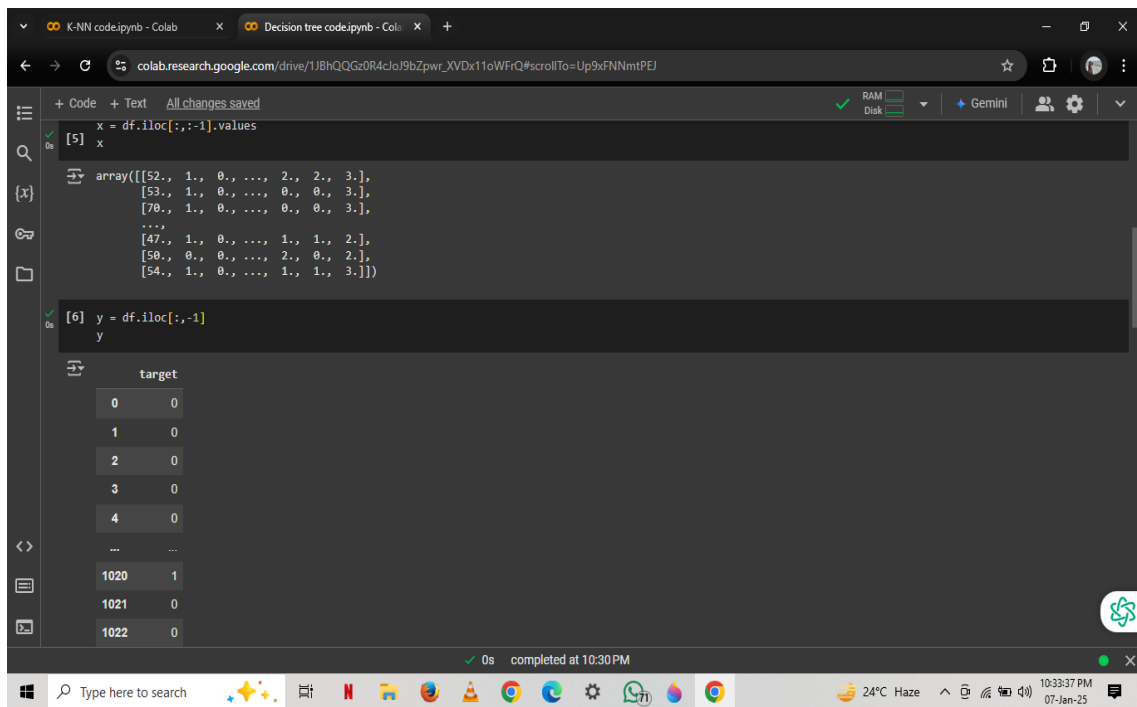
*#Extracting Independent and dependent Variable*

*x= data\_set.iloc[:, :-1].values*

*x*

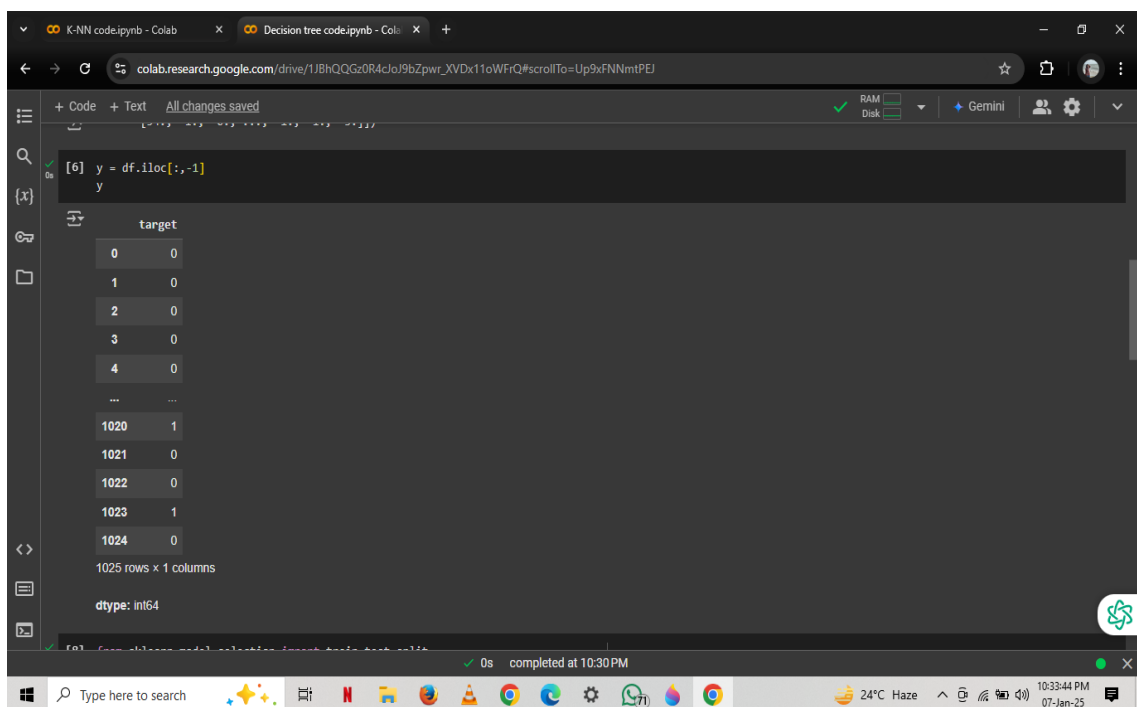
*y= data\_set.iloc[:, -1]*

*y*



```
x = df.iloc[:, :-1].values
x
array([[52., 1., 0., ..., 2., 2., 3.],
       [53., 1., 0., ..., 0., 0., 3.],
       [70., 1., 0., ..., 0., 0., 3.],
       ...,
       [47., 1., 0., ..., 1., 1., 2.],
       [50., 0., 0., ..., 2., 0., 2.],
       [54., 1., 0., ..., 1., 1., 3.]])

[6] y = df.iloc[:, -1]
y
target
0    0
1    0
2    0
3    0
4    0
...  ...
1020  1
1021  0
1022  0
```

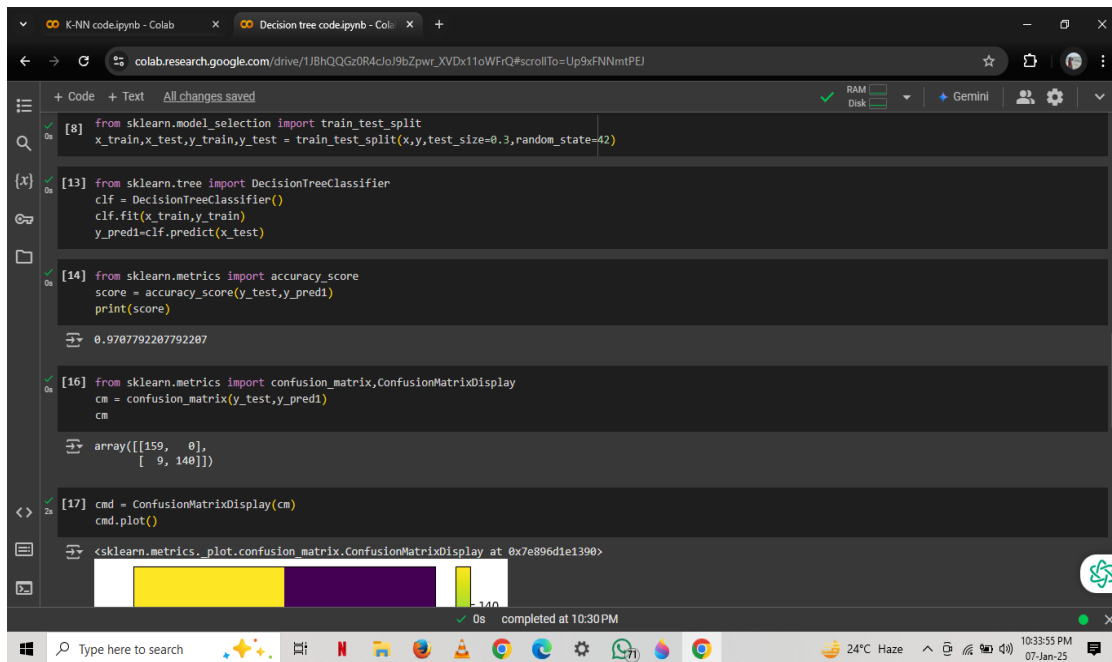


```
[6] y = df.iloc[:, -1]
y
target
0    0
1    0
2    0
3    0
4    0
...  ...
1020  1
1021  0
1022  0
1023  1
1024  0
1025 rows x 1 columns
dtype: int64
```

*# Splitting the dataset into training and test set.*

*from sklearn.model\_selection import train\_test\_split*

*x\_train, x\_test, y\_train, y\_test= train\_test\_split(x, y, test\_size= 0.3, random\_state=42)*



```
[8] from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=42)

[13] from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf.fit(x_train,y_train)
y_pred1=clf.predict(x_test)

[14] from sklearn.metrics import accuracy_score
score = accuracy_score(y_test,y_pred1)
print(score)

0.9787792287792287

[16] from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay
cm = confusion_matrix(y_test,y_pred1)
cm

array([[159,  0],
       [ 9, 140]])

[17] cmd = ConfusionMatrixDisplay(cm)
cmd.plot()

<sklearn.metrics.plot.confusion_matrix.ConfusionMatrixDisplay at 0x7e896d1e1398>
```

## Fitting a Decision Tree algorithm to the Training set

Now we will fit the model to the training set. For this, we will import the DecisionTreeClassifier class from sklearn.tree library. Below is the code for it:

*#Fitting Decision Tree classifier to the training set*

*From sklearn.tree import DecisionTreeClassifier*

*classifier= DecisionTreeClassifier()*

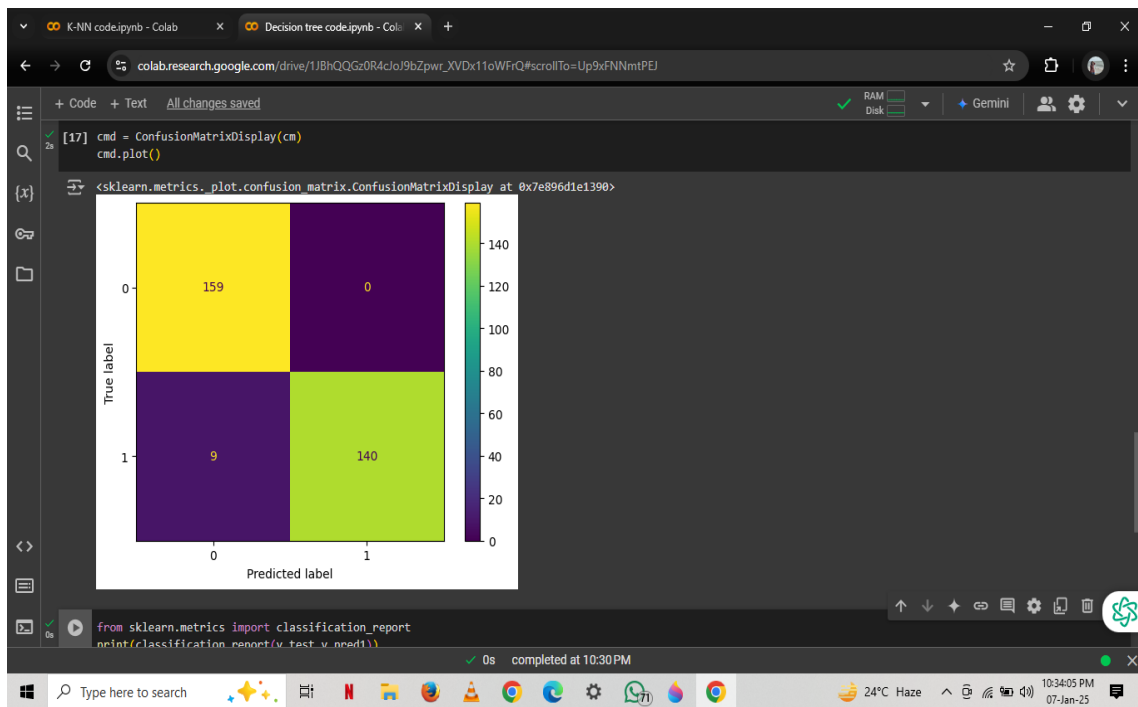
*classifier.fit(x\_train, y\_train)*

## Predicting the test result:

Now we will predict the test set result. We will create a new prediction vector y\_pred. Below is the code for it:

*#Predicting the test set result*

*y\_pred= classifier.predict(x\_test)*



```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred1))
```

	precision	recall	f1-score	support
0	0.95	1.00	0.97	159
1	1.00	0.94	0.97	149
accuracy			0.97	388
macro avg	0.97	0.97	0.97	388
weighted avg	0.97	0.97	0.97	388

In this output, the predicted output and real test output are given. We can clearly see that there are some values in the prediction vector, which are different from the real vector values. These are prediction errors.

## **CONCLUSION**

In this project, I successfully implemented two machine learning algorithms, K-Nearest Neighbors (KNN) and Decision Tree, for predicting heart disease. Through a systematic approach involving data preprocessing, model training, and performance evaluation, I demonstrated how these algorithms could be applied to a medical dataset. The results highlighted the effectiveness of each algorithm in terms of accuracy and other performance metrics. While KNN's simplicity and robustness make it a reliable choice for classification tasks, its performance is highly dependent on the value of  $k$  and computationally intensive for large datasets. Decision Trees, on the other hand, provided an intuitive and easily interpretable model structure but are prone to overfitting without appropriate pruning techniques.

By comparing the models, I observed their respective strengths and weaknesses, emphasizing the importance of algorithm selection based on the nature of the dataset and the problem at hand. This study underscores the potential of machine learning techniques in predictive analytics for healthcare, offering valuable insights for early diagnosis and effective management of heart disease. And I get 97% accuracy score in Decision Tree algorithm while as 71% in K- Nearest Neighbors algorithm. So I came to the conclusion that Decision Tree algorithm provides the accurate and precise prediction result.

## REFERENCES

- Mathematics for machine learning Marc Peter Deisenroth, A. Aldo Faisal, Published by Cambridge University Press (2020).
- <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>
- <https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm>
- <https://www.kaggle.com/code/prthmgoyl/neuralnetwork-heart-disease-dataset/notebook>
- <https://www.geeksforgeeks.org/confusion-matrix-machine-learning/>