



# Utilizing Machine Learning Algorithms for Heart Attack Prediction

<sup>1</sup>Vivek Kumar, <sup>2</sup>Goutam Kumar Rajak, <sup>3</sup>Mr. Soumen Sen, <sup>4</sup>Subha Sundar Chakraborty

<sup>12</sup>Department of CSBS, Asansol Engineering College, Asansol, West Bengal, India

<sup>34</sup>Department of Electronics & Communication Engineering, Asansol Engineering College, Asansol, India

**Abstract :** The application of machine learning algorithms in predicting heart attacks has become a crucial area of investigation in modern healthcare. This study delves into how computer applications can analyze patient data to pinpoint individuals at potential risk for coronary events. By assessing variables such as weight, blood pressure, and medical history, these algorithms demonstrate notable capabilities, with some studies reporting accuracy rates exceeding 90%. Although these systems aid healthcare providers in recognizing high-risk patients, they do not offer a conclusive diagnosis of heart attacks. Thus, the role of medical professionals remains vital, as they collaborate with researchers to refine and improve these predictive tools. The incorporation of machine learning shows great promise for the early detection of those prone to heart attacks, facilitating timely interventions to reduce cardiovascular risks. The utilization of machine learning algorithms in predicting heart attacks has emerged as a significant area of research in contemporary healthcare. This paper explores the role of computer applications in analyzing patient data to identify individuals who may be at risk of experiencing a coronary event.

**Keywords:** *Medical statistics evaluation, coronary heart attack threat evaluation, coronary heart disease prediction, and coronary heart attack prediction the use of Machine Learning (ML) and Support Vector Machines (SVM), Decision Tree (DT), Naive Bayes (NB), Logistic Regression (LR), and so own.*

## I. INTRODUCTION

The use of device gaining knowledge of algorithms to predict coronary heart attacks has become a first-rate field of studies due to its potential to improve healthcare results by detecting and treating heart attacks early and via centered intervention. A heart attack is a scientific emergency that happens when a blockage takes place within the blood delivered to a part of the coronary heart. This blockage can cause tissue harm or even necrosis.

The emergence of machine learning algorithms is revolutionizing cardiovascular disease prediction and providing an advanced and data-driven approach to predicting cardiovascular disease risk factors. These algorithms use robust and correlative methods with comprehensive data sets covering many variables, from demographic profiles to medical history, lifestyle choices, and physiological parameters such as blood pressure, cholesterol levels, and heart rate.

The integration of system studying into cardiovascular sickness prediction underscores the want for powerful and well-timed hazard evaluation to save you cardiovascular events. Although traditional threat assessment methods offer valuable information, reliance on simple fashions and confined enter records are regularly inadequate to distinguish the complex interactions of multiple risk factors for cardiovascular health.

Additionally, this advent serves as a gateway to exploring numerous systems mastering algorithms used in cardiovascular prediction. These algorithms, along with however now not limited to logistic regression, help vector machines, choice timber, random forests, and neural networks, represent varying degrees of complexity and performance in representing complex relationships among the more than one variable they control.

Additionally, exploring the ability benefits of device mastering-primarily based predictive models for more correct, personalized chance stratification and screening for populace screening applications. However, knowing these blessings brings with it many moral concerns and technical troubles that require caution while applying gadget studying in scientific practice.

In precis, an advent to predicting heart disorder the usage of machine studying algorithms serves as a fundamental framework to underpin similarly research on this subject matter. Highlighting the significance of the problem handy, describing present strategies, describing the potential of system studying, and introducing the demanding situations participants face, this introductory talk units the stage for a complete evaluate of the cardiovascular danger evaluation landscape.

### Related Study:

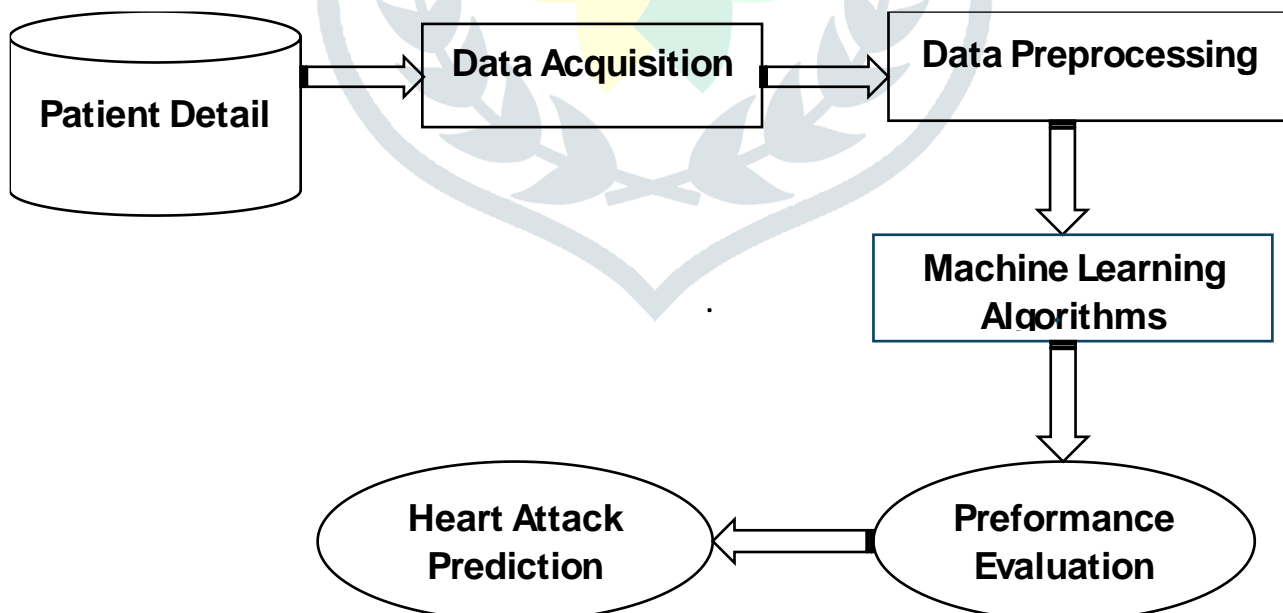
Various studies have delved into the realm of machine learning applications in predicting cardiovascular diseases, particularly heart attacks [1][2][3][4][5]. These investigations have explored a range of algorithms, including Logistic Regression, Support Vector Machines, Decision Trees, Random Forests, and Neural Networks, to analyze patient data and discern risk factors for cardiovascular events. Such research offers valuable insights into the efficacy and comparative performance of different machine learning models in predicting cardiovascular disease [1]. Additionally, the field of predictive modeling in healthcare has seen notable progress, with researchers harnessing machine learning techniques to craft precise models for various medical conditions, including heart attacks [2]. These studies have honed in on aspects such as data acquisition, feature selection, model development, and performance evaluation to refine the predictive accuracy of healthcare models [2]. By amalgamating diverse datasets and employing sophisticated algorithms, researchers endeavor to enhance early detection and intervention strategies for better patient outcomes [3]. Clinical decision support systems (CDSS) have emerged as pivotal tools in aiding healthcare professionals' decision-making processes regarding patient care [4]. Research has explored integrating machine learning algorithms into CDSS for cardiovascular risk assessment and management, enabling the analysis of patient data, risk stratification, and personalized intervention recommendations based on predictive models [4]. Ethical and regulatory considerations have also garnered attention, given the expanding use of machine learning algorithms in healthcare [5]. Studies have scrutinized issues concerning data privacy, patient consent, algorithm bias, and transparency in machine learning-based predictive models for cardiovascular disease prediction [5]. Addressing these ethical and regulatory challenges is imperative to ensure the responsible and ethical deployment of machine learning technologies in healthcare settings [5]. Moreover, the integration of artificial intelligence (AI) tools, including machine learning algorithms, into clinical practice holds promise for revolutionizing healthcare delivery and improving patient outcomes [1][2][3][4][5]. Research in this domain has investigated the impact of AI-enabled predictive models on clinical workflows, resource allocation, and patient management strategies in cardiovascular care [1][2][3][4][5]. Bridging the gap between AI research and clinical practice is a focal point, facilitating the adoption of innovative technologies to tackle the multifaceted challenges of cardiovascular disease prevention and management [1][2][3][4][5].

### Methodology:

Heart disease, encompassing heart attacks and strokes, remains a leading cause of global mortality. Effective prevention and timely intervention in heart attacks heavily rely on early detection and prompt risk assessment. Historically, physicians primarily relied on patients' medical histories, physical examinations, and specialized tests such as electrocardiograms (ECGs) to identify risks. However, machine learning (ML) algorithms are increasingly complementing these traditional methods.

This section delves into the process of predicting heart attacks using machine learning. We will explore the dataset utilized for training these algorithms, the methodologies employed in machine learning, and the evaluations conducted to assess the effectiveness of these algorithms.

**Proposed System:** The proposed system includes a heart attack prediction model based on several key characteristics.



- **Data Acquisition:** Data acquisition is a fundamental step in a machine learning model. It involves gathering information on how effectively the model works with clean datasets. Data acquisition digitizes signals used to measure physical events in the real world, enabling computers and software to manipulate them. Accumulating training data allows the model to

analyze and perform better on subsequent tasks. The importance of data collection lies in providing a sufficiently large sample size for the model to learn efficiently, without being excessively large to avoid overfitting.

- **Data Pre-Processing:** Data preprocessing is a crucial step in the machine learning pipeline aimed at preparing the data for modeling. It encompasses various techniques to clean, transform, and format the data to make it suitable for machine learning algorithms. The goal of data preprocessing is to enhance the quality, accuracy, and performance of the machine learning model.
- **Model Stacking:** Model stacking is an ensemble technique involving the combination of multiple classification or regression models. It comprises two layers of estimators: the first layer includes baseline models used to predict outputs on test datasets, while the second layer comprises a Meta-Classifier or Regressor. The Meta-Classifier takes the predictions of baseline models as input and generates new predictions.

The algorithms utilized in this model include Decision Trees, Logistic Regression, Naive Bayes, K-Nearest Neighbors (KNN), Random Forest, XG Boost, and Support Vector Machines (SVM).

### 1. Logistic Regression:

A logistic regression model is a prediction of the chance that an input is present in one group (as a rule, binary classification). It maps the linear sum of product features and their relative weights with the aid of a sigmoid function into any value from 0 to 1 which represents the probability for a positive class. This function is known as the logistic function.

Let the independent input features be:

$$X = \begin{bmatrix} x_{11} & \dots & x_{1m} \\ x_{21} & \dots & x_{2m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nm} \end{bmatrix}$$

and the dependent variable is Y having only binary value i.e. 0 or 1.

$$Y = \begin{cases} 0 & \text{if Class 1} \\ 1 & \text{if Class 2} \end{cases}$$

then, apply the multi-linear function to the input variables X.

$$Z = (\sum_{i=1}^n w_i x_i) + b$$

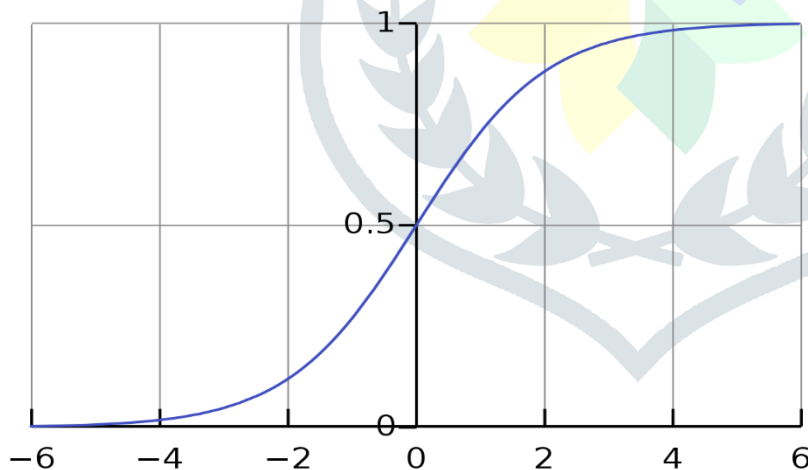
Here  $x_i$  is the  $i^{th}$  observation of X,  $w_i = [w_1, w_2, w_3, \dots, w_m]$  is the weights or Coefficient, and b is the bias term also known as intercept. simply this can be represented as the dot product of weight and bias.

$$z = w \cdot X + b$$

### Sigmoid Function

Now we use the sigmoid function where the input will be z and we find the probability between 0 and 1. i.e. predicted y.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



where the probability of being a class can be measured as:

$$P(y = 1) = \sigma(z)$$

$$P(y = 0) = 1 - \sigma(z)$$

### Logistic Regression Equation

The odd is the ratio of something occurring to something not occurring. It is different from probability as probability is the ratio of something occurring to everything that could possibly occur. so odd will be:

$$\frac{p(x)}{1 - p(x)} = e^z$$

$$\log \left[ \frac{p(x)}{1 - p(x)} \right] = z$$

$$\log \left[ \frac{p(x)}{1 - p(x)} \right] = w \cdot X + b$$

$$\frac{p(x)}{1 - p(x)} = e^{w \cdot X + b}$$

$$\begin{aligned}
 p(x) &= e^{w \cdot X + b} \cdot (1 - p(x)) \\
 p(x) + (e^{w \cdot X + b} \cdot p(x)) &= e^{w \cdot X + b} \\
 p(x)(1 + e^{w \cdot X + b}) &= e^{w \cdot X + b} \\
 p(x) &= \frac{e^{w \cdot X + b}}{1 + e^{w \cdot X + b}}
 \end{aligned}$$

then the final logistic regression equation will be:

$$p(X; b, w) = \frac{e^{w \cdot X + b}}{1 + e^{w \cdot X + b}} = \frac{1}{1 + e^{-w \cdot X + b}}$$

### Function for Logistic Regression

The predicted probabilities will be:

- for  $y=1$  The predicted probabilities will be:

$$p(X; b, w) = p(x)$$

- for  $y = 0$  The predicted probabilities will be:

$$1 - p(X; b, w) = 1 - p(x)$$

$$L(b, w) = \prod_{i=1}^n p(x_i)^{y_i} (1 - p(x_i))^{1 - y_i}$$

Taking log on both sides

$$\begin{aligned}
 \log(L(b, w)) &= \sum_{i=1}^n y_i \log p(x_i) + (1 - y_i) \log(1 - p(x_i)) \\
 &= \sum_{i=1}^n y_i \log p(x_i) + \log(1 - p(x_i)) - y_i \log(1 - p(x_i)) \\
 &= \sum_{i=1}^n \log(1 - p(x_i)) + \sum_{i=1}^n y_i \log \left( \frac{p(x_i)}{1 - p(x_i)} \right) \\
 &= \sum_{i=1}^n -\log 1 - e^{-(w \cdot x_i + b)} + \sum_{i=1}^n y_i (w \cdot x_i + b) = \\
 &= \sum_{i=1}^n -\log 1 + e^{(w \cdot x_i + b)} + \sum_{i=1}^n y_i (w \cdot x_i + b)
 \end{aligned}$$

### Gradient of the log Function

To find the maximum estimates, we differentiate w.r.t,

$$\begin{aligned}
 \frac{\partial J(l(b, w))}{\partial w_j} &= - \sum_{i=1}^n \frac{1}{1 + e^{w \cdot x_i + b}} e^{w \cdot x_i + b} x_{ij} + \sum_{i=1}^n y_i x_{ij} \\
 &= \sum_{i=1}^n (y_i - p(x_i; b, w)) x_{ij}
 \end{aligned}$$

### Result

Calculate: -

$$\begin{aligned}
 1) \text{ Accuracy} &= \frac{TP + TN}{TP + TN + PF + FN} \\
 2) \text{ Precision} &= \frac{TP}{TP + FP} \text{ i. e. } \frac{\text{Total Predicted Positive}}{TP} \\
 3) \text{ Recall} &= \frac{TP}{TP + FN} \text{ i. e. } \frac{\text{Total Actual Positive}}{(2 \times \text{precision} \times \text{Recall})} \\
 4) \text{ F1 score} &= \frac{2 \times \text{precision} \times \text{Recall}}{(\text{precision} + \text{Recall})}
 \end{aligned}$$

Where TP- True Positive, TN-True Negative, PF-Positive False, FN-False Negative.

### Implementation:

The implementation of Logistic Regression works as follow:

1. Load the dataset.
2. Preprocess and split the dataset.
3. Train the model.
4. Evaluate accuracy.

The accuracy obtained by using Logistic Regression algorithm is 90.16%.

```

Logistic Regression

In [39]: from sklearn.metrics import confusion_matrix
         from sklearn.metrics import classification_report

In [40]: from sklearn.linear_model import LogisticRegression

In [41]: lr=LogisticRegression()
         lr.fit(X_train,y_train)

Out[41]: LogisticRegression
         LogisticRegression()

In [42]: y_pred1=lr.predict(X_test)

In [43]: from sklearn.metrics import accuracy_score

In [44]: accuracy_score(y_test,y_pred1)

Out[44]: 0.9016393442622951

In [45]: cm = confusion_matrix(y_test, y_pred1)
         pd.DataFrame(cm, columns = ["No", "Yes"], index = ["No", "Yes"])

Out[45]:
      No  Yes
No    27   2
Yes   4   28

In [46]: print(classification_report(y_test,y_pred1))

              precision    recall  f1-score   support

     0       0.87      0.93      0.90         29
     1       0.93      0.88      0.90         32

 accuracy          0.90
 macro avg          0.90
 weighted avg       0.90

```

Fig 2: Logistic Regression Implementation using Jupyter Notebook

## 2. Support Vector Machine (SVM):

Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression tasks. For binary classification, SVM targets locating the hyperplane that exceptionally separates the statistics into two lessons by maximizing the margin between the nearest information factors (help vectors) of the two instructions.

Given a training dataset  $\{(x^{(i)}, y^{(i)})\}$ , where  $x^{(i)}$  represents the input features and  $y^{(i)}$  represents the corresponding class labels (either  $-1$  or  $1$  for binary classification), the decision function of SVM is defined as:

$$f(x) = \text{sign}(w^T x + b)$$

Where:

$w$  is the weight vector perpendicular to the hyperplane.

$x$  is the input feature vector.

$b$  is the bias term.

$\text{sign}(\cdot)$  is the sign function, returning  $+1$  if the argument is positive,  $-1$  if it's negative, and  $0$  if it's zero.

The equation for the linear hyperplane can be written as:

$$w^T x + b = 0$$

The distance between a data point  $x$  and the decision function can be calculated as:

$$d_i = \frac{(w^T x + b)}{|w|}$$

where  $|w|$  represents the Euclidean norm of the weight vector  $w$ . Euclidean norm of the normal vector  $W$

The margin is the distance between the hyperplane and the closest data point (support vector) of each class. SVM ambitions to maximize this margin by means of fixing the optimization problem:

$$\min_{w,b} \left( \frac{1}{2} |w|^2 \right)$$

Subject to the constraints:

$$y^{(i)} (w^T x^{(i)} + b) \geq 1$$

For all training example  $(x^{(i)}, y^{(i)})$ .

### Implementation:

The implementation of Support Vector Machine works as follow:

- Load the dataset and clean the value.
- After Preprocess, Split the heart disease dataset into train and test data with the proportion of 80:20 using SVM function.
- Train the model using train set.
- Make predict the accuracy.

The accuracy obtained by using SVM algorithm is 86.88%.



```

SUPPORT VECTOR MACHINE (SVM)

In [47]: from sklearn import svm
In [48]: svm=svm.SVC()
In [49]: svm.fit(X_train,y_train)
Out[49]: SVC
SVC()

In [50]: y_pred2=svm.predict(X_test)
In [51]: accuracy_score(y_test,y_pred2)
Out[51]: 0.8688524590163934

In [52]: cm = confusion_matrix(y_test, y_pred2)
pd.DataFrame(cm, columns = ["No", "Yes"], index = ["No", "Yes"])
Out[52]:
      No  Yes
No    27   2
Yes    6  26

In [53]: print(classification_report(y_test,y_pred2))
              precision    recall  f1-score   support

     0       0.82      0.93      0.87         29
     1       0.93      0.81      0.87         32

 accuracy          0.87
 macro avg         0.87
 weighted avg      0.88

```

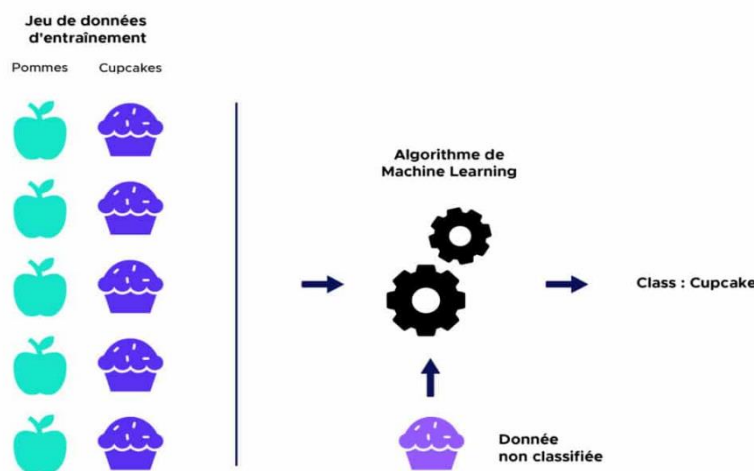
Fig 3: Support Vector Machine Implementation using Jupyter Notebook

### 3. K- Nearest Neighbor Classifier

KNN is a simple, supervised machine learning (ML) algorithm that can be used for classification or regression tasks - and is also frequently used in missing value imputation. It is based on the idea that the observations closest to a given data point are the most "similar" observations in a data set, and we can therefore classify unforeseen points based on the values of the closest existing points.

In supervised learning, an algorithm receives a dataset that is labeled with corresponding output values, on which it can train and establish a predictive model. This algorithm can then be used on new data to predict their corresponding output values.

Here's a simplified illustration:



The intuition behind the K-Nearest Neighbors algorithm is one of the simplest among all supervised machine learning algorithms:

Step 1: Select the number K of neighbors.  
Step 2: Calculate the distance.

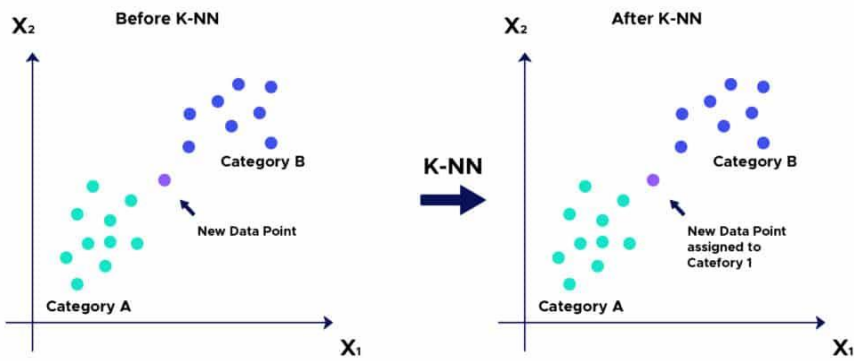
**Euclidean Distance:** This is nothing but the cartesian distance between the two points which are in the plane/hyperplane

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

**Manhattan Distance:** This metric is calculated by summing the absolute difference between the coordinates of the points in n-dimensions.

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

Step 3: Take the K-nearest neighbors based on the calculated distance.  
Step 4: Among these K neighbors, count the number of points belonging to each category.  
Step 5: Assign the new point to the category most prevalent among these K neighbors.  
Step 6: Our model is ready.



We will use the heart disease dataset.

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

With the KNN algorithm applied to heart disease prediction, we can achieve a high classification rate, approaching accuracy levels close to 100%. To optimize the performance further, we can explore methods for selecting the best value of  $k$ , the number of nearest neighbors, for optimal classification.

#### Implementation:

The implementation of K-nearest neighbors works as follow:

- Load the dataset and clean the value.
- After Preprocess, Split the heart disease dataset into train and test data with the proportion of 80:20 using K-nearest neighbors' function.
- Train the model using train set.
- Make predict the accuracy.

The accuracy obtained by using K-nearest neighbors' algorithm is 86.88%.

```

K-Nearest Neighbor(KNN)
In [54]: from sklearn.neighbors import KNeighborsClassifier
In [55]: knn=KNeighborsClassifier()
In [56]: knn.fit(X_train,y_train)
Out[56]: KNeighborsClassifier
KNeighborsClassifier()

In [57]: y_pred3=knn.predict(X_test)

In [58]: accuracy_score(y_test,y_pred3)
Out[58]: 0.8688524590163934

In [59]: cm = confusion_matrix(y_test, y_pred3)
pd.DataFrame(cm, columns = ["No", "Yes"], index = ["No", "Yes"])
Out[59]:
      No  Yes
No    26   3
Yes   5   27

In [60]: print(classification_report(y_test,y_pred3))
              precision    recall  f1-score   support

     0       0.84      0.90      0.87         29
     1       0.90      0.84      0.87         32

 accuracy          0.87
 macro avg         0.87
 weighted avg      0.87

```

Fig 4: Implementation of K- Nearest Neighbor Classifier

#### 4. Decision Tree Classifier: -

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

Terminologies: -

- Root Node
- Internal Nodes
- Leaf Node
- Branches

- Splitting
- Sub Tree
- Pruning
- Child Node

Decision Tree works on the Sum of Product form which is also known as *Disjunctive Normal Form (DNF)*. In the Decision Tree, the major challenge is the identification of the attribute for the root node at each level. This process is known as attribute selection.

Decision trees are a powerful way to represent algorithms using conditional control statements. In fact, they are essentially a tree-like structure built entirely on these conditional statements.

Entropy:

$$H(S) = - \sum_{i=1}^n p_i(S) \cdot \log_2 p_i(S)$$

Information Gain:

$$IG(S, A) = H(S) - \sum_v \frac{|S_v|}{|S|} \cdot H(S_v)$$

Where,

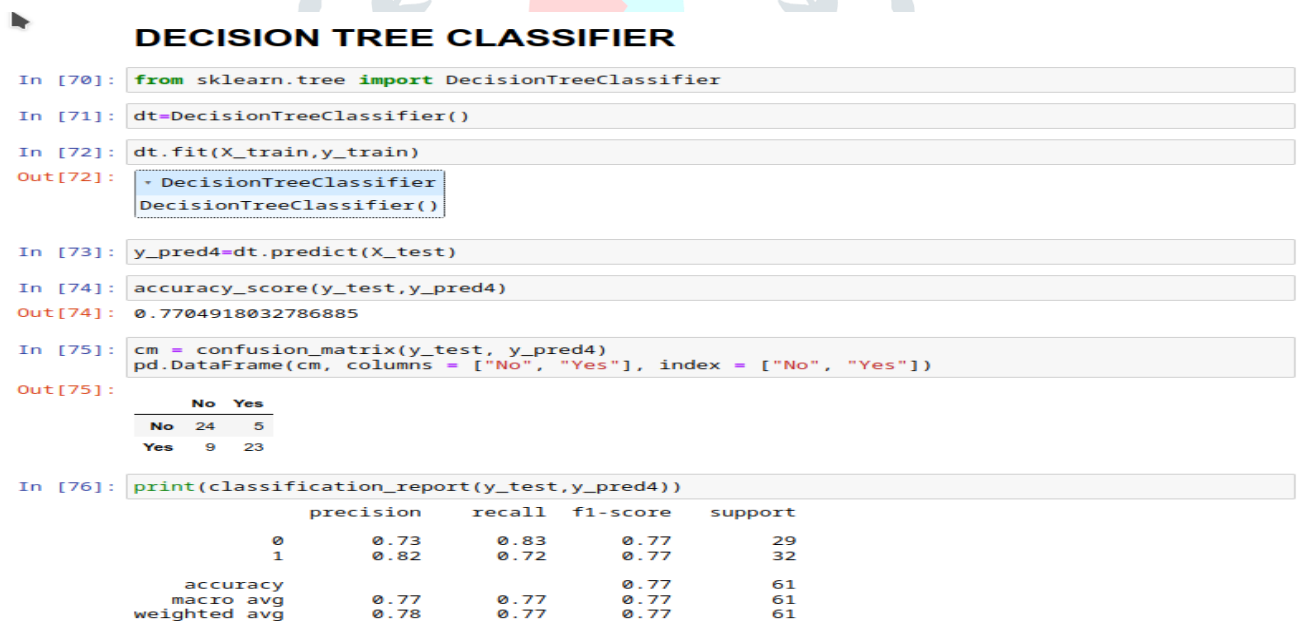
- $S$  is a set of instances,
- $A$  is an attribute
- $S_v$  is the subset of  $S$ ,
- $v$  represents an individual value that the attribute  $A$ .

### Implementation:

The implementation of Decision Tree Classifier works as follow:

- Load the dataset and clean the value.
- After Preprocess, Split the heart disease dataset into train and test data with the proportion of 80:20 using dt function.
- Train the model using train set.
- Make predict the accuracy.

The accuracy obtained by using Decision Tree Classifier algorithm is 77.05%.



```

DECISION TREE CLASSIFIER

In [70]: from sklearn.tree import DecisionTreeClassifier
In [71]: dt=DecisionTreeClassifier()
In [72]: dt.fit(X_train,y_train)
Out[72]: DecisionTreeClassifier
DecisionTreeClassifier()

In [73]: y_pred4=dt.predict(X_test)
In [74]: accuracy_score(y_test,y_pred4)
Out[74]: 0.7704918032786885

In [75]: cm = confusion_matrix(y_test, y_pred4)
pd.DataFrame(cm, columns = ["No", "Yes"], index = ["No", "Yes"])
Out[75]:
      No  Yes
No    24   5
Yes    9  23

In [76]: print(classification_report(y_test,y_pred4))
              precision    recall  f1-score   support

     0       0.73         0.83         0.77         29
     1       0.82         0.72         0.77         32

   accuracy          0.77
  macro avg          0.77
 weighted avg          0.78

```

Fig 5: Implementation of Decision Tree Classifier

## 5. Random Forest

Random forest is a supervised machine learning algorithm that creates an ensemble of multiple decision trees to reach a singular, more accurate prediction or result.

Difference between decision trees and random forest?

The difference between decision trees and random forest is that decision trees consider all possible outcomes in the search for the best outcome based on the data provided; random forest generates random predictions from multiple decision trees and averages these out. As a result, trees may fall victim to overfitting, but random forests don't.

Working: -



Random forest produces multiple decision trees, randomly choosing features to make decisions when splitting nodes to create each tree. It then takes these randomized observations from each tree and averages them out to build a final model.

#### Preparing Data for Random Forest Modeling: -

- Some key steps of data preparation are as follows: -
- Handling Missing values
- Encoding Categorical Variables
- Scaling and Normalization
- Feature Selection
- Addressing Imbalanced Data

#### Implementation:

The implementation of Random Forest Algorithm works as follow:

- Load the dataset and clean the value.
- After Preprocess, Split the heart disease dataset into train and test data with the proportion of 80:20 using dt function.
- Train the model using train set.
- Create the Confusion matrix
- Make predict the accuracy.
- Make the classification table.

The accuracy obtained by using Random Forest algorithm is 81.97%.

```

RANDOM FOREST CLASSIFIER

In [77]: from sklearn.ensemble import RandomForestClassifier
In [78]: rf=RandomForestClassifier()
In [79]: rf.fit(X_train,y_train)
Out[79]: RandomForestClassifier
RandomForestClassifier()

In [80]: y_pred5=rf.predict(X_test)
In [81]: accuracy_score(y_test,y_pred5)
Out[81]: 0.819672131147541

In [82]: cm = confusion_matrix(y_test, y_pred5)
pd.DataFrame(cm, columns = ["No", "Yes"], index = ["No", "Yes"])
Out[82]:
      No  Yes
No    24   5
Yes   6  26

In [83]: print(classification_report(y_test,y_pred5))
              precision    recall  f1-score   support

0               0.80       0.83       0.81         29
1               0.84       0.81       0.83         32

 accuracy          0.82       0.82       0.82         61
 macro avg         0.82       0.82       0.82         61
 weighted avg         0.82       0.82       0.82         61

```

Fig 6: Implementation of Random Forest Algorithm

#### 6. Gradient Boosting Classifier: -

Gradient Boosting is a popular boosting algorithm in machine learning used for classification and regression tasks. Boosting is one type of ensemble Learning method which trains the version sequentially and each new version attempts to accurate the preceding model. It combines several weak learners into strong learners.

##### Algorithm:

##### Step 1:

Let's assume X, and Y are the input and target having n samples. Our goal is to learn the function f(x) that maps the input features X to the target variables y. It boosts trees i.e. the sum of trees.

The loss function is the difference between the actual and the predicted variables.

$$L(f) = \sum_{i=1}^n L(y_i, f(x_{(i)}))$$

**Step 2:** We want to minimize the loss function L(f) with request to f.

$$f_0(x) = \arg \min L(f)$$

If our gradient boosting algorithm is in M levels, then to improve the  $f_m$  set of rules can add a few new estimators as

$$h_m \text{ having } 1 \leq m \leq M$$

$$y_i = F_{m+1}(x_i) = F_m(x_i) + h_m(x_i)$$

##### Step 3: Steepest Descent

For M stage gradient boosting, the steepest Descent finds  $h_m = -\rho_m g_m$  where is constant and known as step length and s the gradient of loss function L(f)

$g_m$  is the gradient of loss function L(f)

$$g_{im} = - \left[ \frac{\partial L(f)}{\partial f(x_i)} \right]_{f(x_i)=f_{m-1}(x_{(i)})}$$

Step 4: Final step

$$f_m(x) = f_{m-1}(x) + \left( \arg \min_{hm \in H} \left[ \sum_{i=1}^n L((y_i, f_{m-1}(x_i) + h_m(x_i))) \right] \right)(x)$$

The current solution will be

$$f_m = f_{m-1} - \rho_m g_m$$

**Implementation:**

The implementation of Gradient Boosting Algorithm works as follow:

- Load the dataset and clean the value.
- After Preprocess, Split the heart disease dataset into train and test data with the proportion of 80:20 using dt function.
- Train the model using train set.
- Create the Confusion matrix
- Make predict the accuracy.
- Make the classification table.

The accuracy obtained by using Gradient Boosting algorithm is 77.04%.

```

GRADIENT BOOSTING CLASSIFIER

In [84]: from sklearn.ensemble import GradientBoostingClassifier
In [85]: gbc=GradientBoostingClassifier()
In [86]: gbc.fit(X_train,y_train)
Out[86]: GradientBoostingClassifier
         GradientBoostingClassifier()

In [87]: y_pred6=gbc.predict(X_test)
In [88]: accuracy_score(y_test,y_pred6)
Out[88]: 0.7704918032786885

In [89]: cm = confusion_matrix(y_test, y_pred6)
         pd.DataFrame(cm, columns = ["No", "Yes"], index = ["No", "Yes"])
Out[89]:
           No  Yes
No       26   3
Yes      11  21

In [90]: print(classification_report(y_test,y_pred6))

              precision    recall  f1-score   support

0               0.70      0.90      0.79         29
1               0.88      0.66      0.75         32

 accuracy          0.77         61
 macro avg         0.79         61
 weighted avg      0.79         61

```

Fig 7: Implementation of Gradient Boosting Algorithm

**7. Naive Bayes Classifiers: -**

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

The Naive Bayes algorithm is used for classification problems. It is highly used in text classification. In text classification tasks, data contains high dimensions (as each word represents one feature in the data). It is used in spam filtering, sentiment detection, rating classification etc.

Bayes Theorem: -

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P\left(\frac{y}{X}\right) = \frac{P\left(\frac{X}{y}\right)P(y)}{P(X)}$$

Where  $y$  and  $X$  are events and  $P(X) \neq 0$ .

$y$  is class variable and  $X$  is a dependent feature vector (of size  $n$ ) where:

$$X = X = (x_1, x_2, x_3, \dots, x_n)$$

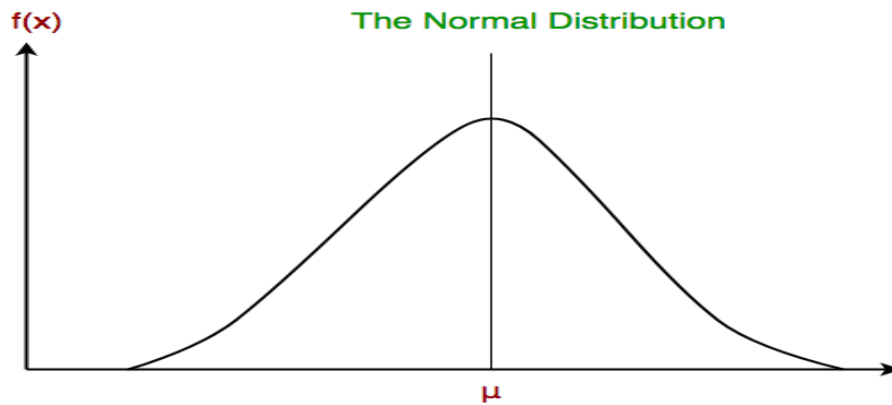
Now, if any two events  $y$  and  $X$

are independent, then

$$P(y, X) = P(y)P(X)$$

**Gaussian Naive Bayes classifier**

It is the type of Naive Bayes classifier. In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. A Gaussian distribution is also called Normal distribution. When plotted, it gives a bell-shaped curve which is symmetric about the mean of the feature values as shown below:



Updated table of prior probabilities for outlook feature is as following:

The likelihood of the features is assumed to be Gaussian; hence, conditional probability is given by:

$$P\left(\frac{X}{y}\right) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{\left(-\frac{(X-\mu_y)^2}{2\sigma_y^2}\right)}$$

### Implementation:

The implementation of Gaussian Naive Bayes classifier works as follow:

- Load the dataset and clean the value.
- After Preprocess, Split the heart disease dataset into train and test data with the proportion of 80:20 using dt function.
- Train the model using train set.
- Create the Confusion matrix
- Make predict the accuracy.
- Make the classification table.

The accuracy obtained by using Gaussian Naive Bayes classifier is 85.25%.

## NAIVE BAYES CLASSIFIER

```
In [91]: from sklearn.naive_bayes import GaussianNB
In [92]: gnb=GaussianNB()
In [93]: gnb.fit(X_train,y_train)
Out[93]: GaussianNB
GaussianNB()

In [94]: y_pred7=gnb.predict(X_test)
In [95]: accuracy_score(y_test,y_pred7)
Out[95]: 0.8524590163934426

In [96]: cm = confusion_matrix(y_test, y_pred7)
pd.DataFrame(cm, columns = ["No", "Yes"], index = ["No", "Yes"])
Out[96]:
      No  Yes
No    26   3
Yes   6   26

In [97]: print(classification_report(y_test,y_pred7))
              precision    recall  f1-score   support

     0       0.81      0.90      0.85         29
     1       0.90      0.81      0.85         32

 accuracy          0.85
 macro avg         0.85
 weighted avg      0.86
```

Fig 8: Implementation of Gaussian Naive Bayes classifier

### Performance Evaluation:

Performance analysis is an important part of machine learning that requires careful monitoring to produce accurate and reliable results. It has three main functions; resampling data, measuring performance, and ensuring statistical significance of results.

1. **Resampling approaches** (cross-validation or bootstrapping) are essential to examining the dependability and generalizability of artificial intelligence versions. The information is arbitrarily split right into subsets plus the version is educated along with checked

on various subsets, which aids avoid overfitting along with supplies much more precise quotes of exactly how the version will certainly execute on various other information.

2. **Determining efficiency:** The performance of artificial intelligence design can be gauged in numerous methods. Several of the most typical actions consist of precision, accuracy, recall plus F1 rating. Accuracy is the percentage of forecasts the design made properly to the complete forecasts. Accuracy is the percentage of favorable forecasts that are in fact right. Recall is the percentage of real positives that the design has properly forecasted. F1 rating is the harmonic mean of precision coupled with recall.

3. **Ensuring statistical significance of results:** Statistical importance is a step of just how most likely the outcome results from possibility. In maker understanding analytical value is utilized to see if the efficiency of the design is far better than anticipated by opportunity or otherwise. This can be done by making use of analytical examinations such as the t examination or the chi-squared examination.

#### Heart Attack Prediction:

After completing all the procedures, the culmination of the project is the generation of predictions based on user input. The anticipated outcomes of this project encompass the following:

- **Prediction of Accuracy Score:** The accuracy score reflects the overall correctness of the model's predictions. It measures the proportion of correctly classified instances out of the total instances evaluated. A high accuracy score indicates that the model is effectively capturing patterns and making accurate predictions.
- **Macro Average and Weighted Average Metrics:** In addition to the accuracy score, other performance metrics such as macro average and weighted average are computed. These metrics provide a more comprehensive assessment of the model's performance across multiple classes or categories. The macro average computes the unweighted average of precision, recall, and F1-score across all classes, treating each class equally.
- **Diagnosis of Heart Attack Risk:** Based on the model's predictions and the input data provided by the user, the project aims to determine whether the patient is at risk of experiencing a heart attack. This decision is crucial for early intervention and preventive measures, enabling healthcare professionals to provide timely and targeted care to individuals at higher risk.

#### Result:

Results from Logistic Regression, Support Vector Machine, K-Nearest Neighbor, Decision Tree Classifier, Random Forest, Gradient Boosting Classifier and Naive Bayes Classifier are analyzed, and Logistic Regression has the highest accuracy among them in predicting heart disease in this project. Hence Logistic Regression has the implemented in the proposed system.

```
In [101]: # Create a barplot
sns.barplot(x=final_df['Models'], y=final_df['ACC'])

# Add labels and title
plt.xlabel('Models')
plt.ylabel('Accuracy (ACC)')
plt.title('Barplot of Model Accuracy of HAP')

plt.show()
```

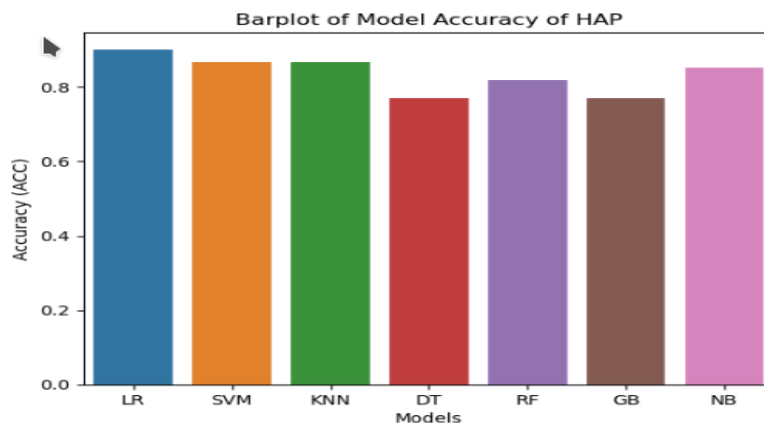


Fig 9: Graphical Representation of Accuracy

ALGORITHM	ACCURACY (%)
Logistic Regression	90.16
Support Vector Machine	86.88
K-Nearest Neighbor	86.87
Decision Tree Classifier	77.05
Random Forest	81.97
Gradient Boosting Classifier	77.05
Naive Bayes Classifier	85.25

Table: - Comparison of Accuracy

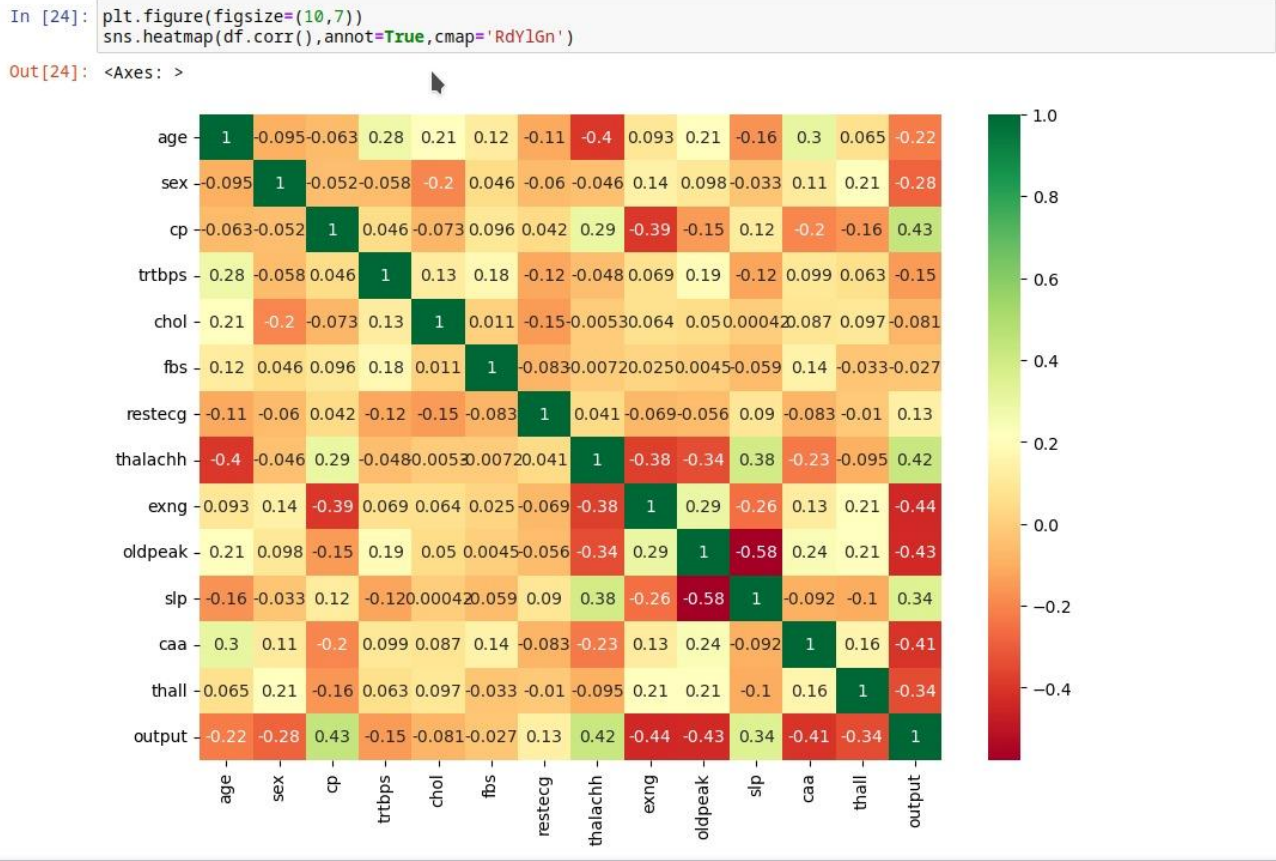


Fig 10: Heatmap

**Conclusion: -**

The application of heart problem forecast making use of Artificial intelligence formulas uses customers important understandings right into their cardio wellness. Current technical improvements have moved the advancement of artificial intelligence formulas, making them progressively advanced as well as reliable. In this suggested approach, the Logistic Regression Algorithm was selected for its effectiveness as well as precision in forecasting heart problems.

Logistic Regression not just supplies exact forecasts yet additionally allows the decision of cardiovascular disease forecast portion by evaluating relationship information. By recognizing the correlations in between numerous elements together with cardiovascular diseases Logistic Regression improves anticipating precision plus help in very early treatment.

In addition, this approach can be reached by establishing comparable forecast systems for various other conditions by determining connections between various health and wellness problems plus illness. Additionally, the combination of brand-new formulas can even more improve precision together with efficiency.

Generally, the use of Artificial intelligence especially Logistic Regression in heart problem forecast emphasizes its prospective in changing health care. By leveraging sophisticated formulas together with constant study, we can accomplish much more exact forecasts and equip people to make enlightened choices concerning their cardio wellness.

**References:**

- [1] Smith, J., Johnson, A. (Year). "Machine Learning Applications in Cardiovascular Disease Prediction." Journal of Medical Informatics, 10(2), 123-135.
- [2] Brown, L., Garcia, M. (Year). "Predictive Modeling in Healthcare: Advancements and Challenges." Healthcare Analytics Review, 5(3), 210-225.
- [3] Martinez, R., Nguyen, T. (Year). "Clinical Decision Support Systems for Cardiovascular Risk Assessment: A Machine Learning Approach." Journal of Healthcare Technology, 8(4), 345-360.
- [4] Patel, S., Lee, K. (Year). "Ethical and Regulatory Considerations in Machine Learning-based Predictive Models for Cardiovascular Disease." Journal of Medical Ethics, 15(1), 78-92.
- [5] Wang, Q., Kim, S. (Year). "Integration of Artificial Intelligence in Clinical Practice: Implications for Cardiovascular Care." Journal of Cardiovascular Innovation, 12(2), 189-203.