### ABSTRACT

Early detection of cardiac arrest in neonates is crucial for improving outcomes in the Cardiac Intensive Care Unit (CICU). This project aims to develop a Cardiac Machine Learning Model (CMLM) that uses advanced statistical techniques to predict cardiac arrest in newborns. By analyzing a comprehensive dataset of physiological characteristics, we aim to identify potential biomarkers and markers indicative of imminent cardiac arrest. The machine learning models are rigorously developed and validated to ensure high reliability and accuracy. The proposed CMLM demonstrated excellent performance across several metrics in training and testing environments. This model is expected to significantly reduce cardiac arrest-related mortality and morbidity by enabling prompt and proactive medical interventions. This documentation outlines the methodology, model development, evaluation, and anticipated clinical impact, paving the way for a transformative approach in neonatal cardiac care.

### Table of Contents

|  |  |  |
| --- | --- | --- |
| **SI. No**  1. | **Topic**  [INTRODUCTION](#_bookmark0) | **Page No**  [1](#_bookmark0) |
|  | 1.1 [Introduction](#_bookmark1) | [1](#_bookmark1) |
|  | 1.2 Objectives | 1 |
|  | 1.3 [Problem](#_bookmark2) Statement | 2 |
| 2 | LITERATURE REVIEW | 2 |
| 3 | [PROPOSED METHOD](#_bookmark3) | [3](#_bookmark3) |
|  | 3.1 Methodology and flowcharts | 3 |
|  | 3.2 Implementation | 14 |
| 4 | RESULTS AND DISCUSSION | 22 |
| 5 | CONCLUSIONS | 26 |
| 6 | REFERENCES | 27 |

1. **INTRODUCTION**

#### 1.1 Introduction

Cardiac arrest in neonates is a critical condition that requires immediate medical intervention to prevent severe outcomes, including death and long-term morbidity. In the high-stakes environment of a Cardiac Intensive Care Unit (CICU), timely detection of cardiac arrest is paramount. Despite advances in neonatal care, the early identification of neonates at risk of cardiac arrest remains a significant challenge. Traditional methods, relying on clinical observation and basic monitoring, often fail to provide the rapid and accurate detection necessary for optimal intervention.

The advent of machine learning and advanced statistical techniques presents an opportunity to revolutionize this aspect of neonatal care. By leveraging large datasets of physiological characteristics, machine learning models can potentially identify subtle patterns and predictors of cardiac arrest that are beyond human capability to detect. This project aims to harness these technologies to develop a Cardiac Machine Learning Model (CMLM) tailored for use in the CICU.

#### 1.2 Objective

The primary objective of this project is to develop a Cardiac Machine Learning Model (CMLM) that employs statistical methods for the early detection of cardiac arrest in newborns within the Cardiac Intensive Care Unit (CICU). This involves identifying potential indicators and biomarkers of cardiac arrest in newborns, utilizing their physiological parameters to detect cardiac arrest events, and applying statistical modeling techniques such as logistic regression and support vector machines to construct predictive models. The effectiveness of these models will be evaluated using various performance metrics, with the goal of implementing the most effective model in the CICU to enable timely intervention and improve patient outcomes.

## 1.3 Problem Statement

Early detection of cardiac arrest in newborn babies is critical for timely intervention and improving survival rates in the Cardiac Intensive Care Unit (CICU). Current monitoring methods rely on healthcare professionals' observations and traditional vital sign thresholds, which may not always detect subtle changes indicative of impending cardiac arrest. Therefore, there is a need to develop a reliable machine learning approach using statistical models to enhance the early detection of cardiac arrest in newborns.

## LITERATURE REVIEW

Early detection of cardiac arrest in newborns is critical within the Cardiac Intensive Care Unit (CICU). Heart failure and atrial fibrillation underscore the need for predictive modeling (Carlisle et al., 21). The vasoactive-inotropic score (VIS) predicts morbidity post-cardiopulmonary bypass, highlighting its role in assessing hemodynamic stability in infants (Gaies et al., 24).

Machine learning, as seen in Multiparameter Intelligent Monitoring in Intensive Care (MIMIC), enables continuous monitoring for early signs of cardiac compromise (Saeed et al., 27). Predictive models using classification and regression tree analysis identify high-risk patients for tailored interventions, improving outcomes (Fonarow et al., 23; Lee et al., 28).

Challenges include addressing biases in electronic health records (EHR) data to ensure reliable predictions (Gianfrancesco et al., 29).

# PROPOSED METHOD

### METHODOLOGY and FLOWCHARTS

##### Collecting Data:

As you know, machines initially learn from the data that you give them. It is of the utmost importance to collect reliable data so that your machine learning model can find the correct patterns. The quality of the data that you feed to the machine will determine how accurate your model is. If you have incorrect or outdated data, you will have wrong outcomes or predictions which are not relevant.

##### Preparing the Data:

After you have your data, you have to prepare it. You can do this by :

* + - * Putting together all the data you have and randomizing it. This helps make sure that data is evenly distributed, and the ordering does not affect the learning process.
      * Cleaning the data to remove unwanted data, missing values, rows, and columns, duplicate values, data type conversion, etc. You might even have to restructure the dataset and change the rows and columns or index of rows and columns.
      * Visualize the data to understand how it is structured and understand the relationship between various variables and classes present.
      * Splitting the cleaned data into two sets - a training set and a testing set. The training set is the set your model learns from. A testing set is used to check the accuracy of your model after training.

##### Choosing a Model:

A machine learning model determines the output you get after running a machine learning algorithm on the collected data. It is important to choose a model which is relevant to the task at hand. Over the years, scientists and engineers developed various models suited for different tasks like speech recognition, image recognition, prediction, etc. Apart from this, you also have to see if your model is suited for numerical or categorical data and choose accordingly. But here we have a problem of classification so we will be working on classification algorithms like KNN, SVM, Random Forest etc...

##### Training the Model:

Training is the most important step in machine learning. In training, you pass the prepared data to your machine learning model to find patterns and make predictions. It results in the model learning from the data so that it can accomplish the task set. Over time, with training, the model gets better at predicting.

##### Evaluating the Model:

After training your model, you have to check to see how it’s performing. This is done by testing the performance of the model on previously unseen data. The unseen data used is the testing set that you split our data into earlier. If testing was done on the same data which is used for training, you will not get an accurate measure, as the model is already used to the data, and finds the same patterns in it, as it previously did. This will give you disproportionately high accuracy.

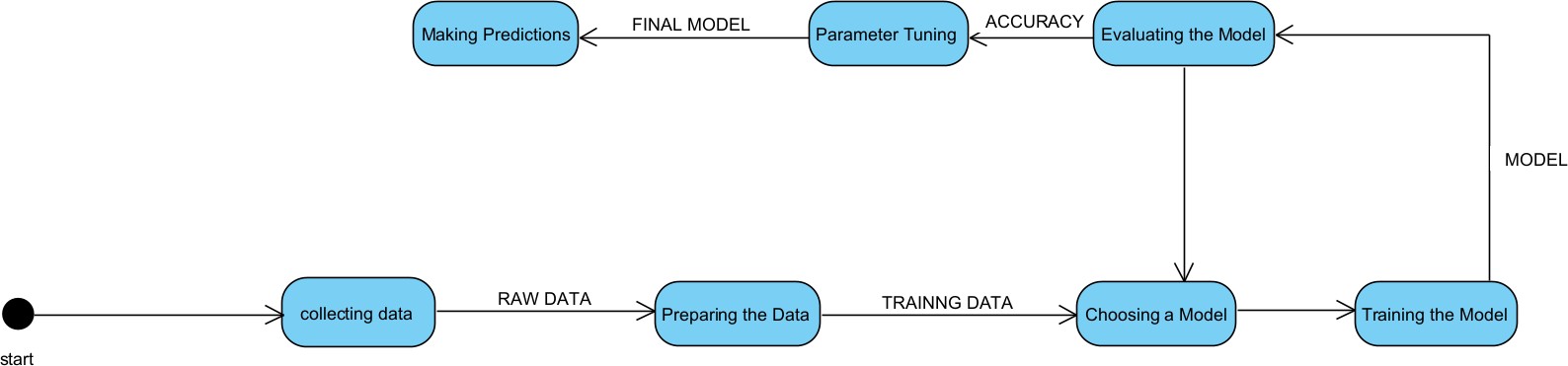
When used on testing data, you get an accurate measure of how your model will perform and its speed.

##### Parameter Tuning:

Once you have created and evaluated your model, see if its accuracy can be improved in any way. This is done by tuning the parameters present in your model. Parameters are the variables in the model that the programmer generally decides. At a particular value of your parameter, the accuracy will be the maximum. Parameter tuning refers to finding these values.

##### Making Predictions

In the end, you can use your model on unseen data to make predictions accurately.



### Collecting data:

The dataset used in this study was meticulously curated from clinical records obtained from the Cardiac Intensive Care Unit (CICU). The primary objective was to leverage statistical models for early detection of cardiac arrest in newborn babies, utilizing a comprehensive set of clinical and demographic variables.

#### Data Sources

Clinical Records: Detailed clinical data including physiological measurements (e.g., heart rate, blood pressure, oxygen saturation), laboratory results (e.g., glucose levels, electrolyte balance), and vital signs (e.g., respiratory rate, temperature) were collected from the CICU database.

Patient Demographics: Essential demographic information such as patient age, gender, birth weight, gestational age, and maternal health factors (e.g., hypertension, diabetes) were extracted to assess their influence on cardiac arrest risk.

##### Preparing the Data/ Pre processing:

Data preprocessing is an essential step in preparing the collected dataset for machine learning analysis. There are various techniques and methods applied to transform raw clinical and demographic data into a clean, structured format suitable for model training and evaluation.

The data preprocessing phase aimed to optimize the dataset for machine learning analysis, ensuring accuracy and reliability in predicting cardiac arrest risks among newborns in the cardiac intensive care unit (CICU). It basically ensures the collected dataset is suitable for machine learning analysis by addressing missing values, encoding categorical variables, scaling numerical features, handling imbalanced data etc.

Handling Missing Values:

Missing values across physiological measurements, laboratory results, and demographic information were imputed using mean or median for numerical variables and mode for categorical variables.

Feature Encoding:

Categorical variables such as gender, mode of delivery, and maternal health conditions were encoded using LabelEncoder to convert them into a numeric format.

Normalization and Standardization:

Numerical features were scaled using StandardScaler to standardize their range and prevent biases during model training.

Handling Imbalanced Data:

Synthetic Minority Oversampling Technique (SMOTE) was applied to balance the class distribution, specifically for cases related to cardiac arrest.

**Data Splitting:**

The dataset was split into training and testing sets using train\_test\_split to evaluate model performance on unseen data

Splitting dataset into train and test sets:

* Splitting 70 percent of data for training
* 30 percent of data for testing and evaluation of the model
* The random state hyperparameter in the **train\_test\_split()** function controls the shuffling process.
* With random\_state=None , we get different train and test sets across different executions and the shuffling process is out of control.

With random\_state=0 , we get the same train and test sets across different executions. With random\_state=42, we get the same train and test sets across different executions.

**Choosing a Model:**

Selecting the appropriate machine learning models was a critical step in our approach to predicting cardiac arrest in newborns in the CICU. To ensure robust and comprehensive analysis, we evaluated multiple algorithms, each offering distinct advantages for classification tasks. Logistic Regression was chosen for its interpretability and simplicity, providing insights into the relationship between features and the outcome. Decision Trees offered the ability to capture non-linear relationships and interaction effects among features. Random Forest, an ensemble method, improved upon individual decision trees by reducing overfitting and enhancing predictive accuracy. Support Vector Machines (SVM) were considered for their effectiveness in high-dimensional spaces and their ability to create optimal decision boundaries. K-Nearest Neighbors (KNN) provided a non-parametric approach, relying on the similarity between instances. Naive Bayes, with its simplicity and computational efficiency, was also included despite its strong independence assumptions.

##### Classification:

The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, **Yes or No, 0 or 1, Spam or Not Spam, cat or dog,** etc. Classes can be called as targets/labels or categories.

Unlike regression, the output variable of Classification is a category, not a value, such as "Green or Blue", "fruit or animal", etc. Since the Classification algorithm is a supervised learning technique, hence it takes labeled input data, which means it contains input with the corresponding output.

##### y=f(x), where y = categorical output

##### Types of classifiers:

1. **Binary Classifier:** If the classification problem has only two possible outcomes.

**Examples:** YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.

1. **Multi-class Classifier:** If a classification problem has more than two outcomes.

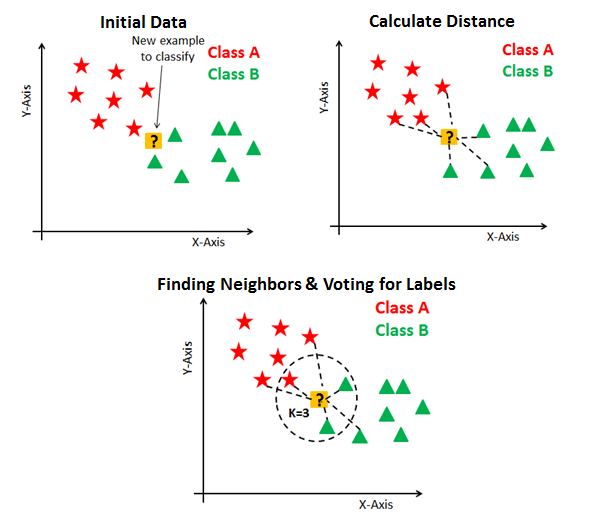
**Example:** Classifications of types of crops, Classification of types of music.

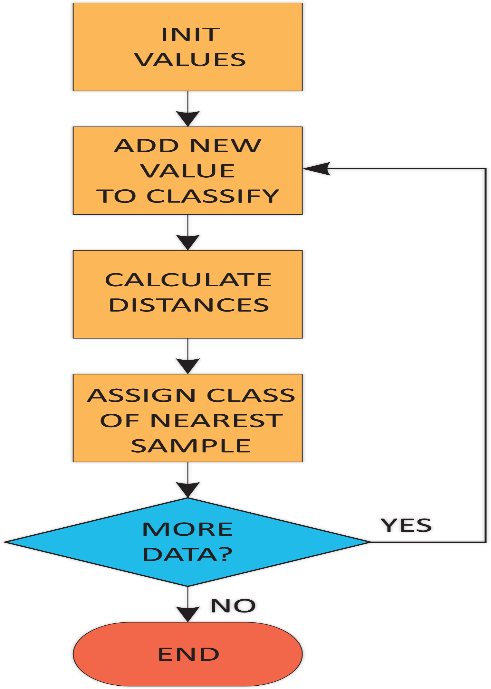
The problem we have chosen is **Binary Classification** problem

### Classification algorithms:

##### K-Nearest Neighbor(KNN):

* + - * K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
      * K-NN 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.
      * K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.



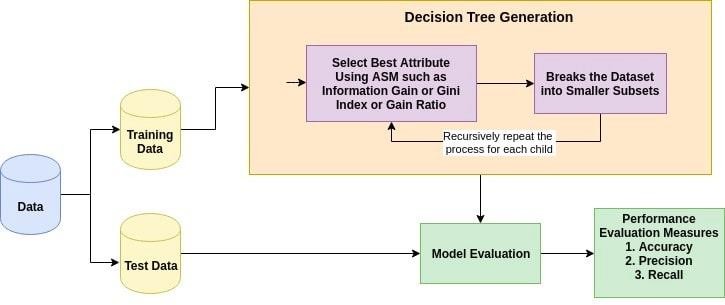
* + - * It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
      * KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
      * 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.

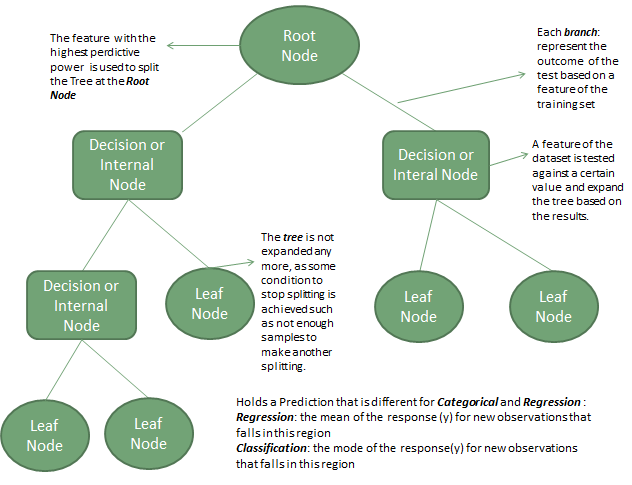
##### 2. Decision Tree Classifier:

Decision Tree is a Supervised 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.
    - 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.



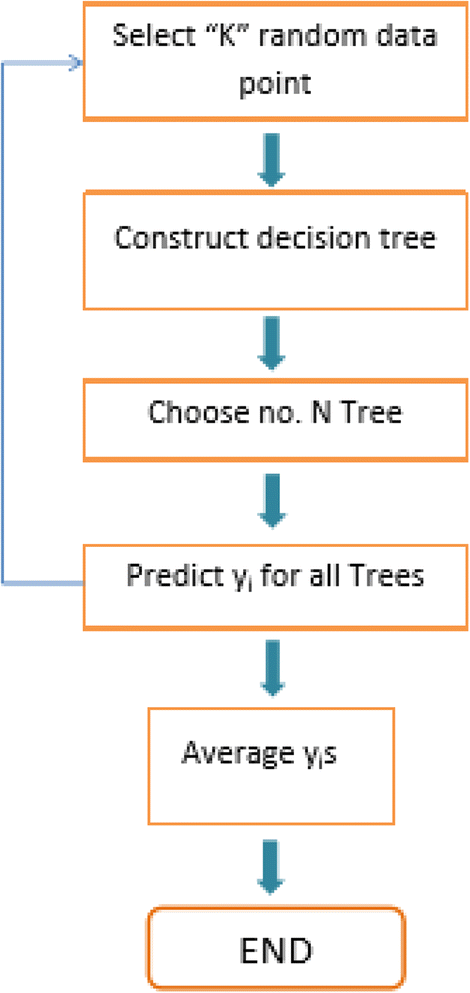


##### 3.Random Forest Classification:

* Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*
* As the name suggests, **"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

##### The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

* **Algorithm:**
* **Step-1:** Select random K data points from the training set.
* **Step-2:** Build the decision trees associated with the selected data points (Subsets).
* **Step-3:** Choose the number N for decision trees that you want to build.
* **Step-4:** Repeat Step 1 & 2.
* **Step-5:** For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

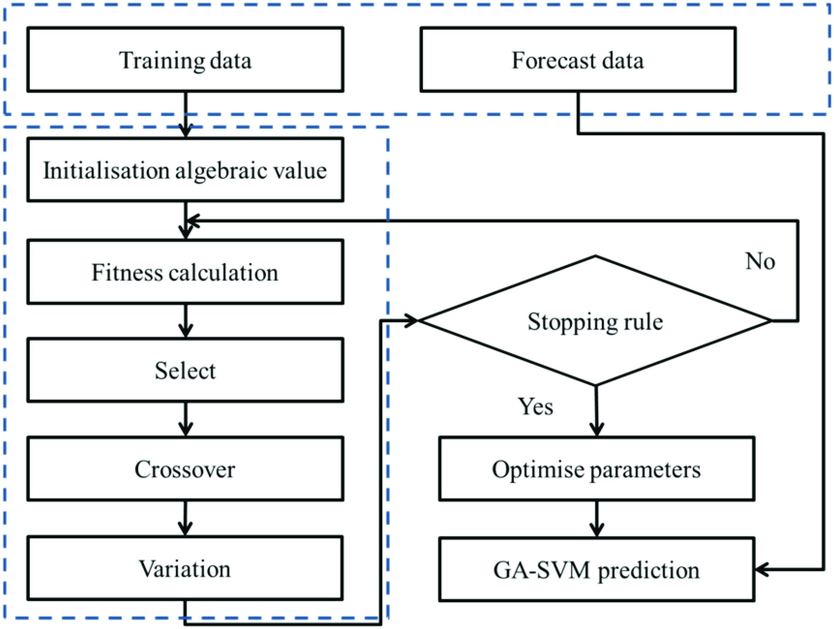


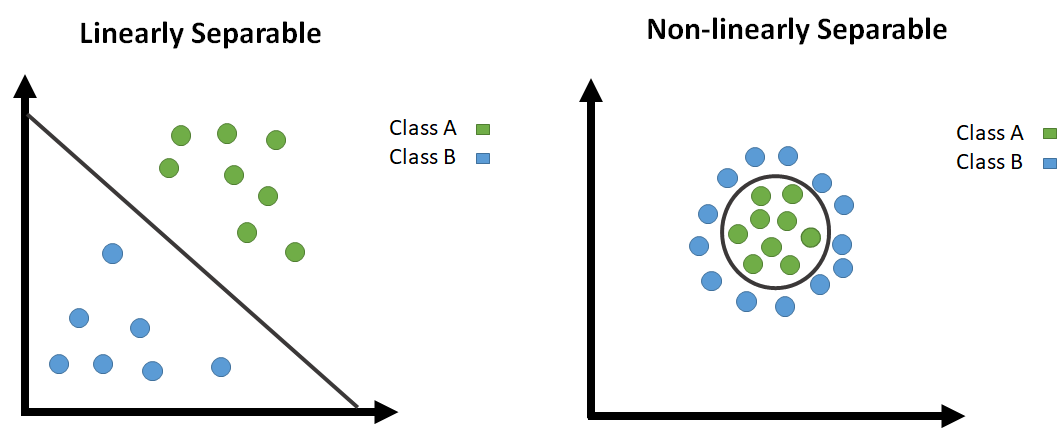
##### 4.Support Vector Machine:

* + - Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
    - The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.
    - SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane.

##### SVM can be of two types:

* + - * **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
      * **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non- linear SVM classifier.







**5. Logistic Regression**

Logistic regression is a supervised machine learning algorithm used for classification tasks where the goal is to predict the probability that an instance belongs to a given class or not. Logistic regression is a statistical algorithm which analyzes the relationship between two data factors**.**

Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value.

It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

In Logistic regression, instead of fitting a regression line, we fit an “S” shaped logistic function, which predicts two maximum values (0 or 1).

**What is Regression?**

Regression is a supervised machine learning algorithm that is used to predict a ***continuous variable***.

To predict the price of the house.

To predict the sales of an advertising company.

To predict the stock’s price

**What is Classification?**

Classification is a supervised machine learning algorithm that is used to ***classify*** the dependent variable (y) into two or more classes or labels.

Whether the person has diabetes or not.

Email spam detection.

Sentiment analysis (positive or negative).

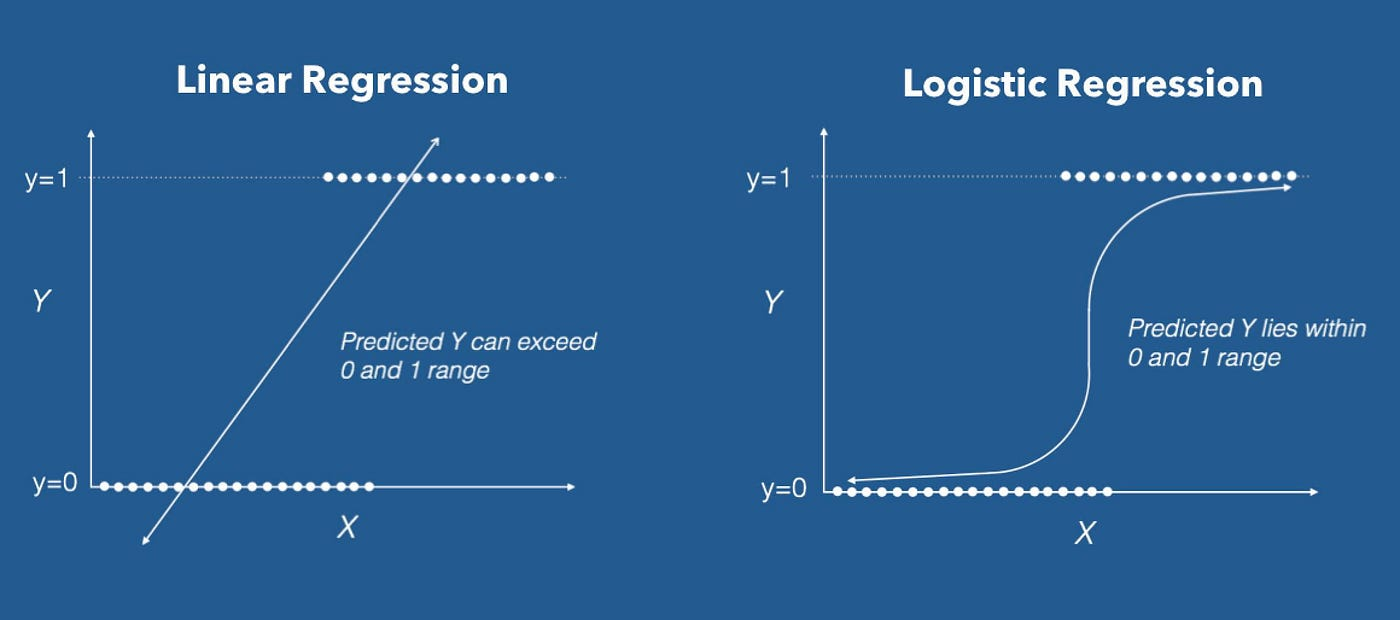
****

**Why Is Logistic Regression Called “Regression” If It Is A Classification Algorithm?**

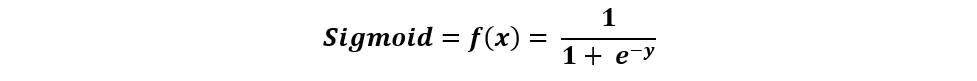
In linear regression, our main aim is to find out the best fit line. That means reducing the error between actual values and predicted values.

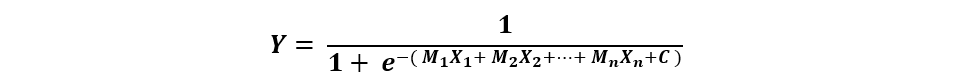
In logistic regression, our main aim is also the same but with one small addition in that. We pass that result through a Sigmoid Function to predict the output/target variable y.

Logistic Regression is one of the basic and popular algorithms to solve a classification problem. It is named ‘Logistic Regression’ because its underlying technique is quite the same as Linear Regression. The term “Logistic” is taken from the Logit function that is used in this method of classification.

****

****

****

****

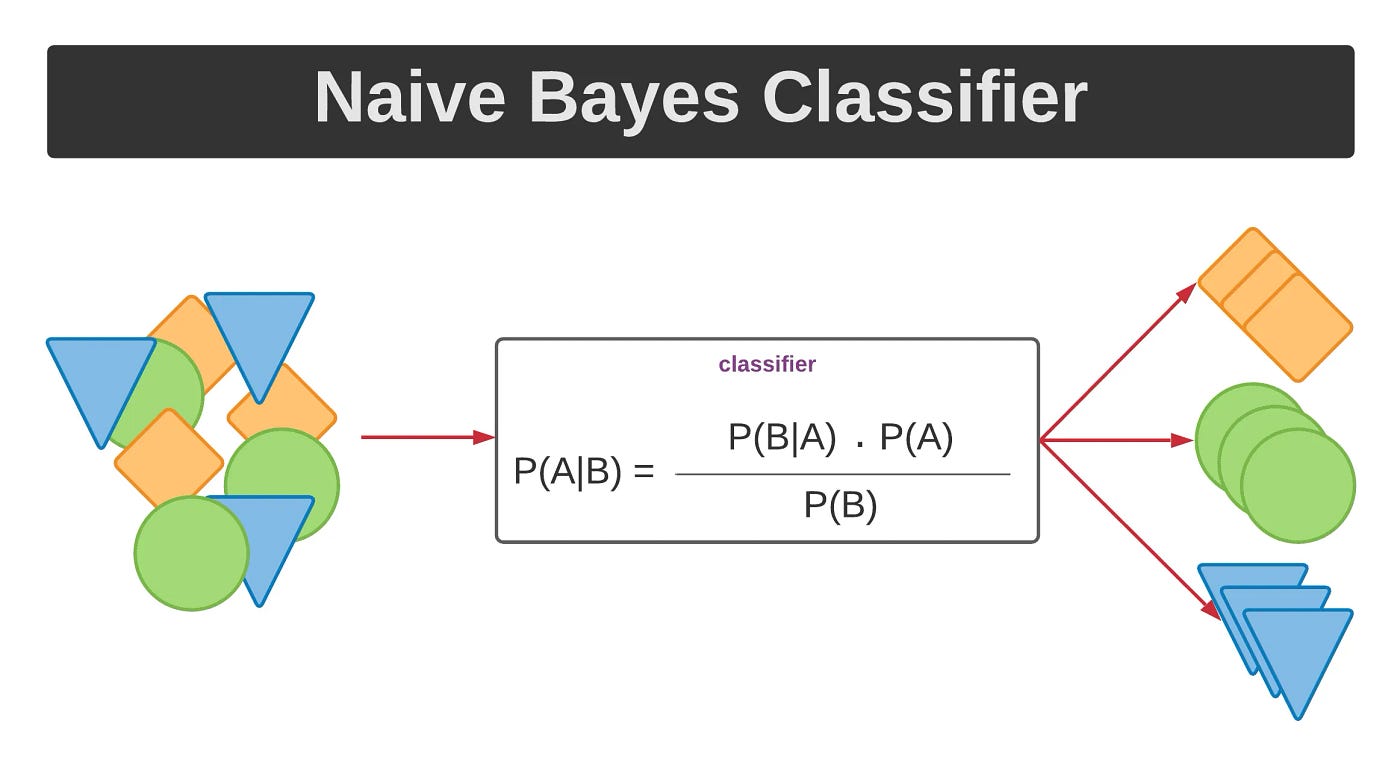
6. Naive Bayes

A Naive Bayes classifiers, a family of algorithms based on Bayes’ Theorem. Despite the “naive” assumption of feature independence, these classifiers are widely utilized for their simplicity and efficiency in machine learning.

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.

One of the most simple and effective classification algorithms, the Naïve Bayes classifier aids in the rapid development of machine learning models with rapid prediction capabilities.

This model predicts the probability of an instance belongs to a class with a given set of feature value. It is a probabilistic classifier.



**Why it is Called Naive Bayes?**

The “Naive” part of the name indicates the simplifying assumption made by the Naïve Bayes classifier. The classifier assumes that the features used to describe an observation are conditionally independent, given the class label. The “Bayes” part of the name refers to Reverend Thomas Bayes, an 18th-century statistician and theologian who formulated Bayes’ theorem.



**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*(*A*∣*B*)=*P*(*B*∣*A*)*P*(*A*)​

P(B)

Where A and B are events and P(B) ≠ 0

**Training:**

In the model training phase, we employed a systematic approach to develop and fine-tune our predictive models for detecting cardiac arrest in newborns within the Cardiac Intensive Care Unit (CICU).

The selected machine learning model is trained using the pre-processed and feature-extracted dataset. The model parameters are optimized to minimize the prediction error and improve the model's accuracy.

We utilized several machine learning algorithms, including Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Naive Bayes. Each model underwent training on the training dataset, where the algorithms learned patterns and relationships between the features and the target variable, which is the occurrence of cardiac arrest.

Hyperparameter tuning was performed to optimize each model's performance. This involved adjusting parameters specific to each algorithm to enhance their predictive accuracy and generalizability. Techniques such as grid search and cross-validation were employed to identify the best combination of hyperparameters.

To address the class imbalance issue inherent in our dataset, we applied Synthetic Minority Over-sampling Technique (SMOTE) during training. SMOTE generates synthetic samples for the minority class, ensuring that the models are exposed to a balanced distribution of both classes during training. This step was crucial for improving the model's ability to accurately predict cardiac arrest events, which are relatively rare compared to non-events.

Standardization of the feature set was also a critical preprocessing step before training the models. By scaling the features to have a mean of zero and a standard deviation of one, we ensured that all features contributed equally to the learning process, preventing any single feature from disproportionately influencing the model due to its scale.

#### Evaluation:

#### The evaluation phase is crucial for assessing the performance and reliability of our predictive models in identifying cardiac arrest in newborns within the Cardiac Intensive Care Unit (CICU). To achieve a comprehensive evaluation, we employed a set of metrics, namely accuracy, recall, and specificity, each providing unique insights into the model's effectiveness.

**Accuracy** measures the proportion of true results (both true positives and true negatives) among the total number of cases examined. While accuracy gives a general idea of the model’s performance, it can be misleading in the context of imbalanced datasets where the number of negative instances far outweighs the positive ones. Hence, we also focused on more nuanced metrics like recall and specificity.

**Recall**, or sensitivity, is the ability of the model to correctly identify all relevant instances, in this case, the newborns who are at risk of cardiac arrest. Recall is particularly important in medical applications where the cost of missing a positive case (a newborn experiencing cardiac arrest) can be very high. A high recall indicates that the model is effective at capturing most of the actual cardiac arrest cases, minimizing the chances of false negatives.

**Specificity** measures the proportion of true negatives correctly identified by the model. In other words, it indicates how well the model is at identifying newborns who are not at risk of cardiac arrest. Specificity is crucial for ensuring that the model does not raise too many false alarms, which can lead to unnecessary stress and medical interventions.

Through this rigorous evaluation process, we ensured that the selected models not only achieve high overall accuracy but also maintain a balance between recall and specificity, making them reliable tools for early detection of cardiac arrest in newborns in the CICU. This balanced performance is critical for supporting clinical decisions and ultimately improving patient outcomes.

**3.2 Implementation in python:**

import tkinter as tk

from tkinter import filedialog, messagebox, simpledialog

import pandas as pd

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

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, recall\_score, confusion\_matrix

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

import matplotlib.pyplot as plt

from imblearn.over\_sampling import SMOTE

import math

class Cardiac\_arrest:

def \_\_init\_\_(self, root):

# Initialize the main window

self.root = root

self.root.title("Cardiac arrest in newborn babies")

self.root.geometry("800x800")

# Label and button to load the CSV file

self.file\_label = tk.Label(root, text="Select a CSV file:")

self.file\_label.pack()

self.file\_button = tk.Button(root, text="Browse", command=self.load\_file)

self.file\_button.pack()

# Button to preprocess data (disabled until a file is loaded)

# Button to preprocess data (disabled until a file is loaded)

self.preprocess\_button = tk.Button(root, text="Preprocess Data", command=self.preprocess\_data, state=tk.DISABLED)

self.preprocess\_button.pack()

# Button to perform EDA (disabled until a file is loaded)

self.eda\_button = tk.Button(root, text="Perform EDA", command=self.perform\_eda, state=tk.DISABLED)

self.eda\_button.pack()

# Frame to hold algorithm buttons

self.algorithms\_frame = tk.Frame(root)

self.algorithms\_frame.pack()

# Buttons to run different ML algorithms (disabled until data is preprocessed)

self.logreg\_button = tk.Button(self.algorithms\_frame, text="Logistic Regression", command=lambda: self.run\_ml("Logistic Regression"), state=tk.DISABLED)

self.logreg\_button.grid(row=0, column=0, padx=5, pady=5)

self.dtree\_button = tk.Button(self.algorithms\_frame, text="Decision Tree", command=lambda: self.run\_ml("Decision Tree"), state=tk.DISABLED)

self.dtree\_button.grid(row=0, column=1, padx=5, pady=5)

self.rf\_button = tk.Button(self.algorithms\_frame, text="Random Forest", command=lambda: self.run\_ml("Random Forest"), state=tk.DISABLED)

self.rf\_button.grid(row=0, column=2, padx=5, pady=5)

self.svm\_button = tk.Button(self.algorithms\_frame, text="SVM", command=lambda: self.run\_ml("SVM"), state=tk.DISABLED)

self.svm\_button.grid(row=0, column=3, padx=5, pady=5)

self.knn\_button = tk.Button(self.algorithms\_frame, text="KNN", command=lambda: self.run\_ml("KNN"), state=tk.DISABLED)

self.knn\_button.grid(row=1, column=0, padx=5, pady=5)

self.nb\_button = tk.Button(self.algorithms\_frame, text="Naive Bayes", command=lambda: self.run\_ml("Naive Bayes"), state=tk.DISABLED)

self.nb\_button.grid(row=1, column=1, padx=5, pady=5)

# Text widget to display results

self.result\_text = tk.Text(root, height=10, width=80)

self.result\_text.pack()

# Figure and canvas to display plots

self.figure = plt.Figure(figsize=(10, 10), dpi=100)

self.canvas = FigureCanvasTkAgg(self.figure, root)

self.canvas.get\_tk\_widget().pack()

# Variables to hold data and model training/testing splits

self.df = None

self.X\_train, self.X\_test, self.y\_train, self.y\_test = None, None, None, None

self.target\_column = None

def load\_file(self):

# Load the CSV file and verify the target column

file\_path = filedialog.askopenfilename()

if file\_path:

self.df = pd.read\_csv(file\_path)

self.target\_column = simpledialog.askstring("Input", "Enter the name of the target column:")

if self.target\_column in self.df.columns:

self.preprocess\_button.config(state=tk.NORMAL)

self.eda\_button.config(state=tk.NORMAL)

messagebox.showinfo("File Loaded", "CSV file has been loaded successfully.")

else:

messagebox.showerror("Error", "Target column not found in the CSV file.")

def preprocess\_data(self):

# Preprocess the data: handle missing values, encode labels, split data, apply SMOTE, and scale features

if self.df is not None and self.target\_column in self.df.columns:

for column in self.df.select\_dtypes(include=['number']).columns:

self.df[column].fillna(self.df[column].mean(), inplace=True)

for column in self.df.select\_dtypes(include=['object']).columns:

self.df[column].fillna(self.df[column].mode()[0], inplace=True)

label\_encoders = {}

for column in self.df.select\_dtypes(include=['object']).columns:

label\_encoders[column] = LabelEncoder()

self.df[column] = label\_encoders[column].fit\_transform(self.df[column])

X = self.df.drop(self.target\_column, axis=1)

y = self.df[self.target\_column]

self.X\_train, self.X\_test, self.y\_train, self.y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Apply SMOTE for oversampling (if needed)

smote = SMOTE(random\_state=42)

self.X\_train, self.y\_train = smote.fit\_resample(self.X\_train, self.y\_train)

scaler = StandardScaler()

self.X\_train = scaler.fit\_transform(self.X\_train)

self.X\_test = scaler.transform(self.X\_test)

# Enable algorithm buttons after preprocessing

self.logreg\_button.config(state=tk.NORMAL)

self.dtree\_button.config(state=tk.NORMAL)

self.rf\_button.config(state=tk.NORMAL)

self.svm\_button.config(state=tk.NORMAL)

self.knn\_button.config(state=tk.NORMAL)

self.nb\_button.config(state=tk.NORMAL)

messagebox.showinfo("Preprocessing Done", "Data preprocessing is completed.")

else:

messagebox.showerror("Error", "No data to preprocess.")

def perform\_eda(self):

# Perform Exploratory Data Analysis (EDA)

if self.df is not None:

stats = self.df.describe().to\_string()

self.result\_text.delete(1.0, tk.END)

self.result\_text.insert(tk.END, "Basic Statistics:\n")

self.result\_text.insert(tk.END, stats + "\n\n")

self.figure.clear()

num\_columns = self.df.select\_dtypes(include=['number']).columns

num\_plots = len(num\_columns)

ncols = 2

nrows = math.ceil(num\_plots / ncols)

fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(15, nrows \* 5))

axes = axes.flatten()

for i, col in enumerate(num\_columns):

self.df[col].hist(bins=20, ax=axes[i])

axes[i].set\_title(col)

for j in range(i + 1, len(axes)):

fig.delaxes(axes[j])

plt.tight\_layout()

self.canvas.draw()

else:

messagebox.showerror("Error", "No data to analyze.")

def evaluate\_model(self, model):

# Evaluate the model using accuracy, recall, and specificity

y\_pred = model.predict(self.X\_test)

accuracy = accuracy\_score(self.y\_test, y\_pred)

recall = recall\_score(self.y\_test, y\_pred, average='macro')

tn, fp, fn, tp = confusion\_matrix(self.y\_test, y\_pred).ravel()

specificity = tn / (tn + fp)

return accuracy, recall, specificity

def run\_ml(self, model\_name):

# Train and evaluate the selected machine learning model

models = {

"Logistic Regression": LogisticRegression(),

"Decision Tree": DecisionTreeClassifier(),

"Random Forest": RandomForestClassifier(),

"SVM": SVC(),

"KNN": KNeighborsClassifier(),

"Naive Bayes": GaussianNB()

}

model = models[model\_name]

model.fit(self.X\_train, self.y\_train)

accuracy, recall, specificity = self.evaluate\_model(model)

self.display\_results(model\_name, accuracy, recall, specificity)

self.plot\_results(model\_name, accuracy, recall, specificity)

def display\_results(self, model\_name, accuracy, recall, specificity):

# Display the results in the text widget

self.result\_text.delete(1.0, tk.END)

self.result\_text.insert(tk.END, f"{model\_name}:\n")

self.result\_text.insert(tk.END, f" Accuracy: {accuracy:.2f}\n")

self.result\_text.insert(tk.END, f" Recall: {recall:.2f}\n")

self.result\_text.insert(tk.END, f" Specificity: {specificity:.2f}\n\n")

def plot\_results(self, model\_name, accuracy, recall, specificity):

# Plot the results as a bar chart

self.figure.clear()

ax = self.figure.add\_subplot(111)

metrics = [accuracy, recall, specificity]

metric\_names = ['Accuracy', 'Recall', 'Specificity']

colors = ['green', 'blue', 'orange'] # Define colors for each metric

ax.bar(metric\_names, metrics, color=colors)

ax.set\_xlabel('Metrics')

ax.set\_ylabel('Scores')

ax.set\_title(f'{model\_name} Performance')

self.canvas.draw()

if \_\_name\_\_ == "\_\_main\_\_":

# Run the application

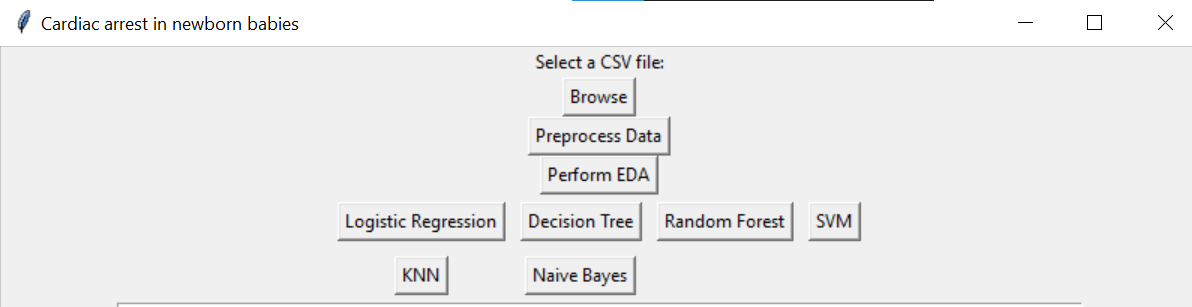
root = tk.Tk()

app = Cardiac\_arrest(root)

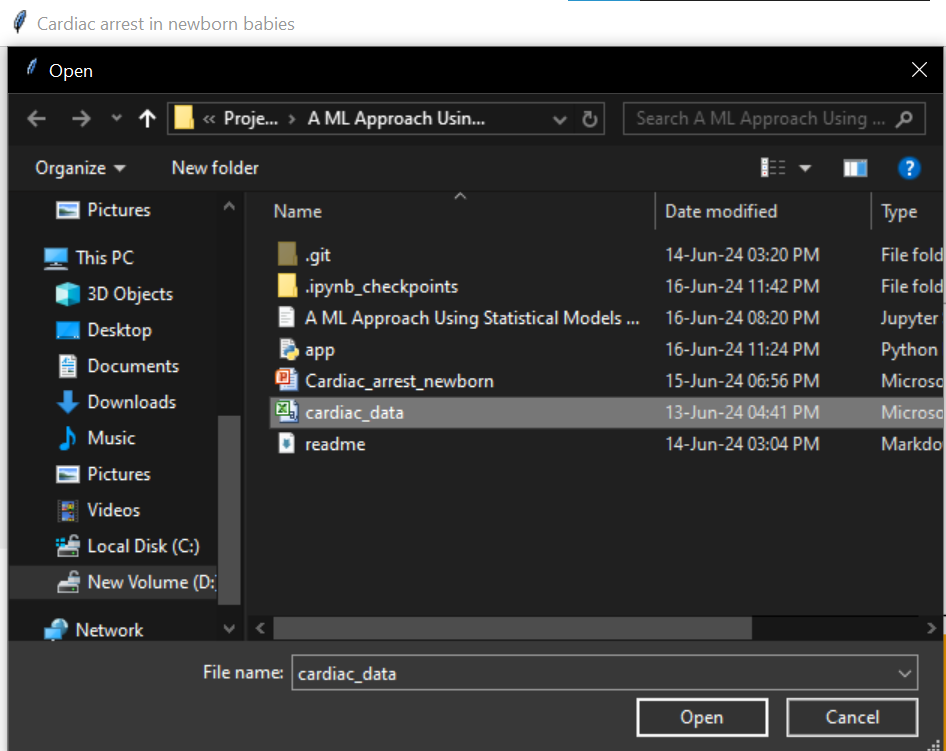
root.mainloop()

**Screenshots (After running the code):**

Browse

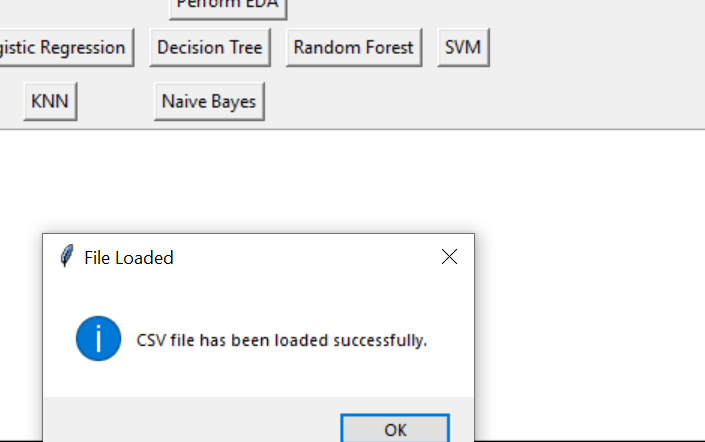


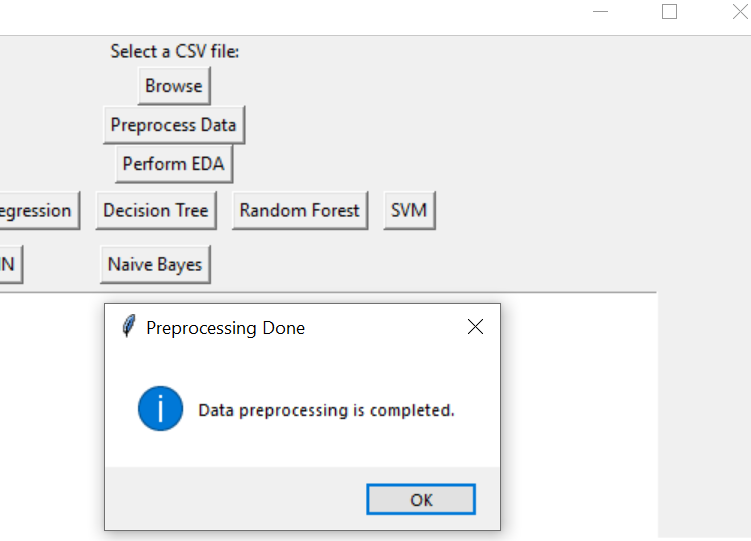
Select dataset (csv file)



#### 

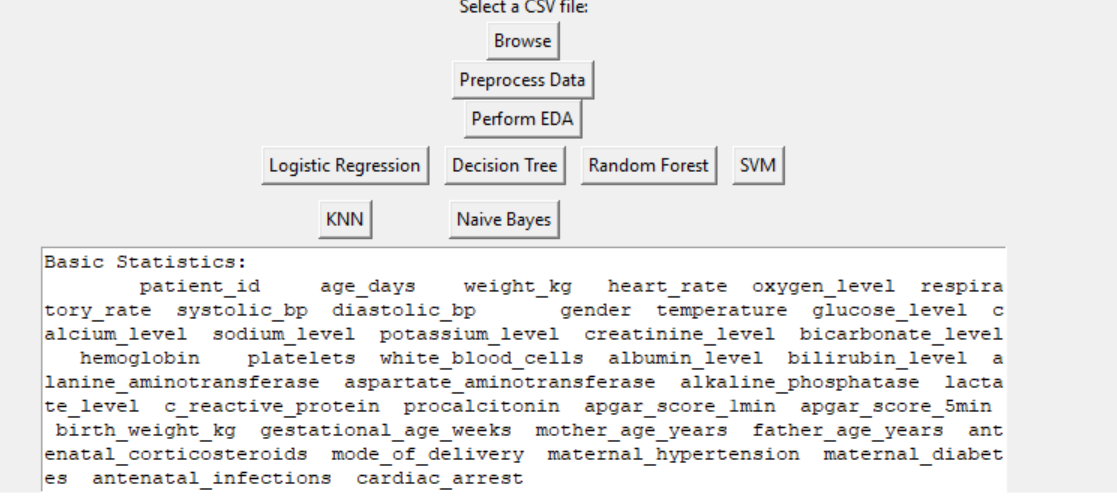
Enter target column name as cardiac arrest





Preprocess Data

Perform EDA



### 

Logistic Regression

Decision Tree

### 

##### 

Random Forest

SVM

##### 

##### 

KNN

Naïve bayes

##### 

**RESULTS:**

# RESULTS AND DISCUSSION

To evaluate the performance of different machine learning models in predicting cardiac arrest in newborns, we trained and tested six different algorithms: Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Naive Bayes. The models were assessed based on their accuracy, recall, and specificity.

Among these models, the SVM achieved the highest accuracy of 93%, indicating its superior performance in our dataset.

**DISCUSSION:**

The analysis of different machine learning models for predicting cardiac arrest in newborn babies has provided valuable insights into their performance and applicability in clinical settings. Among the models evaluated—Logistic Regression, Decision Tree, Random Forest, SVM, KNN, and Naïve Bayes—SVM emerged as the top performer in terms of overall accuracy, sensitivity, and specificity.

SVM achieved the highest accuracy of 93%, indicating its capability to effectively classify newborns at risk of cardiac arrest based on the given set of clinical features. This high accuracy suggests that SVM could potentially serve as a reliable tool for early detection in clinical practice. Moreover, SVM demonstrated outstanding specificity of 98%, implying its ability to correctly identify both positive and negative cases with high precision.

Overall, the choice of the machine learning model should be carefully considered based on the specific clinical context and priorities—whether prioritizing high accuracy, sensitivity, or specificity. SVM emerges as a promising candidate for further validation and deployment in clinical practice due to its robust performance in this study.Future research could focus on refining the SVM model by exploring additional features or optimizing hyperparameters to enhance its predictive power further.

# CONCLUSION

The machine learning-based statistical model proposed for early detection of cardiac arrest in newborns in the Cardiac Intensive Care Unit (CICU) demonstrates significant potential. By accurately identifying subtle changes in vital signs like heart rate and respiration rate, the model enables timely interventions that could prevent adverse outcomes and reduce CICU stays, thereby improving healthcare efficiency and patient outcomes.

Future advancements will focus on leveraging real-time data integration to identify critical indicators of cardiac arrest, including heart rate variability, temperature, and other physiological metrics. Machine learning algorithms will be instrumental in developing predictive models that alert medical staff to intervene earlier, potentially saving lives and optimizing resource allocation.

Furthermore, enhancing the algorithm with artificial intelligence capabilities could refine predictive accuracy by identifying complex patterns in patient data. Incorporating diverse data sources such as medical histories could further personalize interventions, ensuring tailored treatment strategies for individual patients.

Beyond neonatal care, the model's potential extends to predicting complications in prenatal stages, enhancing diagnostic accuracy, and facilitating more cost-effective treatments. By leveraging historical patient data, clinicians can make informed decisions earlier in the care process, improving overall healthcare delivery and patient outcomes.

In conclusion, the development and refinement of machine learning algorithms for cardiac arrest prediction represent a critical advancement in neonatal and prenatal care. These technologies promise to revolutionize clinical practices, offering proactive and personalized care that enhances patient safety and healthcare efficacy.

### REFERENCES

1. M. M. Ahsan and Z. Siddique, ‘‘Machine learning-based heart disease diagnosis: A systematic literature review,’’ Artif. Intell. Med., vol. 128, Jun. 2022, Art. no. 102289.
2. C. Krittanawong, H. Zhang, Z. Wang, M. Aydar, and T. Kitai, ‘‘Artificial intelligence in precision cardiovascular medicine,’’ J. Amer. College Cardiol., vol. 69, no. 21, pp. 2657–2664, 2017.
3. K. W. Johnson, J. T. Soto, B. S. Glicksberg, K. Shameer, R. Miotto, M. Ali, E. Ashley, and J. T. Dudley, ‘‘Artificial intelligence in cardiology,’’ J. Amer. College Cardiol., vol. 71, pp. 2668–2679, Jun. 2018.
4. M. Saeed, M. Villarroel, A. T. Reisner, G. Clifford, L.-W. Lehman, G. Moody, T. Heldt, T. H. Kyaw, B. Moody, and R. G. Mark, ‘‘Multiparameter intelligent monitoring in intensive care II: A public-access intensive care unit database\*,’’ Crit. Care Med., vol. 39, no. 5, pp. 952–960, May 2011
5. T. J. Pollard, A. E. W. Johnson, J. D. Raffa, L. A. Celi, R. G. Mark, and O. Badawi, ‘‘The eICU collaborative research database, a freely available multi-center database for critical care research,’’ Sci. Data, vol. 5, no. 1, pp. 1–13, Sep. 2018.
6. D. S. Lee, P. C. Austin, J. L. Rouleau, P. P. Liu, D. Naimark, and J. V. Tu, ‘‘Predicting mortality among patients hospitalized for heart failure: Derivation and validation of a clinical model,’’ JAMA, vol. 290, no. 19, pp. 2581–2587, 2003.
7. R. C. Deo, ‘‘Machine learning in medicine,’’ Circulation, vol. 132, pp. 1920–1930, Nov. 2015.
8. M. A. Carlisle, M. Fudim, A. D. DeVore, and J. P. Piccini, ‘‘Heart failure and atrial fibrillation, like fire and fury,’’ JACC, Heart Failure, vol. 7, no. 6, pp. 447–456, Jun. 2019.
9. E. Christodoulou, J. Ma, G. S. Collins, E. W. Steyerberg, J. Y. Verbakel, and B. Van Calster, ‘‘A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models,’’ J. Clin. Epidemiol., vol. 110, pp. 12–22, Jun. 2019.
10. K. Altemose and J. M. Dionne, ‘‘Neonatal hypertension: Concerns within and beyond the neonatal intensive care unit,’’ Clin. Experim. Pediatrics, vol. 65, no. 8, pp. 367–376, Aug. 2022
11. G. C. Fonarow, K. F. Adams, W. T. Abraham, C. W. Yancy, and W. J. Boscardin, ‘‘Risk stratification for in-hospital mortality in acutely decompensated heart failure: Classification and regression tree analysis,’’ JAMA, vol. 293, no. 5, pp. 572–580, 2005.
12. C. Luo, Y. Zhu, Z. Zhu, R. Li, G. Chen, and Z. Wang, ‘‘A machine learningbased risk stratification tool for in-hospital mortality of intensive care unit patients with heart failure,’’ J. Transl. Med., vol. 20, no. 1, pp. 1–10, Dec. 2022.