October 6th University

Faculty of Information System and Computer Science

Heart Disease Detection

Course title:

Artificial Intelligence

Course instructor:

DR. Hussam Elbehiery

Supervision of:

Eng. Doaa Bliedy

Prepared by

Name	ID	Grade
Moataz Ibrahim Gaber	202124938	
Ali Farouk Ali	202116264	
Mario Emad Edward	202116342	
Omar Mohamed Elsayed	202122997	
Nancy Khaled Sayed	202124689	
Nada Fathy Sleiman	202116211	
Nourhan Khaled Mohamed	202125050	

Table of contents :

Introduction

Summary :

Heart disease remains a formidable global health challenge, with its various forms affecting millions of lives annually. The quest for accurate and timely detection of heart-related conditions has prompted the integration of Artificial Intelligence (AI) into healthcare solutions.

The Role of AI in Healthcare :

Artificial Intelligence, particularly machine learning, has emerged as a transformative force in healthcare diagnostics. By leveraging sophisticated algorithms, AI systems can analyze complex datasets, identifying patterns and correlations that might elude traditional diagnostic methods. In the context of heart disease, the application of classification algorithms holds promise for improving accuracy and efficiency in the diagnostic process data.

Project Objective :

The primary objective of this project is to develop a robust and accurate heart disease detection system. By harnessing the capabilities of these classification algorithms, we aim to create a tool that not only enhances diagnostic accuracy but also provides valuable insights into the contributing factors associated with heart disease. The Python programming language ensures the project's accessibility, flexibility, and potential for future integration into healthcare systems.

Body

What is classification?

In the context of artificial intelligence (AI), classification refers to the task of categorizing input data into predefined classes or categories. It is a type of supervised learning, where the algorithm is trained on a labeled dataset, meaning that each input is associated with a corresponding output label.

The goal of a classification algorithm is to learn mapping from input features to the correct output class so that it can make accurate predictions on new, unseen data. The process typically involves two main steps: training and testing.

1. Training:

- During the training phase, the algorithm is provided with a dataset in which each example is paired with the correct class label. The algorithm learns the patterns and relationships within the features of the data and their corresponding labels.

- The training process involves adjusting the parameters of the model to minimize the difference between the predicted output and the actual labels.

2. Testing (or Inference):

- Once the model is trained, it is tested on new, unseen data to evaluate its performance. The model predicts the class labels for the test data, and the predictions are compared to the actual labels to assess the accuracy of the model.

Common algorithms used for classification tasks include:

- Decision Trees: These are tree-like structures where each node represents a decision based on a specific feature, leading to different branches and ultimately resulting in a class label.

- K-Nearest Neighbors (KNN): KNN classifies data points based on the majority class among their k-nearest neighbors in the feature space.

- Random Forests: Random forests are ensembles of decision trees, combining the predictions of multiple trees to improve accuracy and robustness.

Classification is a fundamental concept in machine learning and AI, and it finds applications in various domains, such as image recognition, spam detection, medical diagnosis, and more.

Algorithms

- ## K-Neighbors Classification (KNN):

  - **Definition of K-Neighbors**: *It* is an algorithm that allows the system to classify points based on their proximity to adjacent points in space.

  - **Uses of K-Neighbors:**

    1.Data classification: KNN is used to classify points in certain categories based on their similarity with adjacent points.

    2.Value estimation: can be used to estimate a new value through average neighbor values.

  - **Advantages of K-Neighbors:**

    1. Simple and effective in many cases.

    2.does not include training period as the data itself is a model which will be the reference for future prediction.

    3.No assumptions about the data. This is different than some other algorithms.

  - **Disadvantages of K-Neighbors**:

    1.He suffers from significant time consumption when the data are large.

    2.Sensitivity to abnormal values or noise in data.

    3.He needs a correct choice of k value, which may affect the performance of the algorithm.

  - **The Accuracy of K-Neighbors in this code:** 0.62

• Decision Tree Classification:

- **Definition of Decision Tree**: *It is a popular machine learning algorithm that is widely used for both classification and regression tasks. Here's an overview of key aspects of decision trees and some areas of research.*

- **Uses of Decision Tree:**

1. Classification: Assigning categories or labels to input data based on decision rules.
2. Regression: Predicting numerical values by mapping input features to an output value.
3. Decision Making: Providing a clear and interpretable decision-making structure for various scenarios.

- **Advantages of Decision Tree:**

1. Interpretability: Easily understandable and interpretable.
2. Versatility: Handles both numerical and categorical data.
3. No Feature Scaling: Does not require feature scaling and automatically deals with missing values.

- **Disadvantages of Decision Tree**:

1.Overfitting: Prone to overfitting, especially with deep trees.

2.Sensitivity to Noisy Data: Can be sensitive to noisy data and outliers.

3.Instability: Small changes in data can lead to different tree structures.

4.Limited Expressiveness: May struggle to capture complex relationships compared to more advanced models.

- **The Accuracy of Random Forest in this code: 0.97727**

- # Random Forest Classification:

- **Definition of Random Forest**: It is a machine learning model that relies on the idea of constructing multiple decision trees and combining their results to obtain a final prediction or decision.

- **Uses of Random Forest:**
  1. Classification: Used in data classification problems.
  2. Prediction: Can be employed for predicting future values or events.
  3. Pattern Discovery: Assists in examining relationships and patterns in data.

- **Advantages of Random Forest:**

  1. Robust to Noise: Can handle inconsistent or missing data.

  2. Accuracy: Effective in dealing with large and complex datasets.

  3. Capable of handling a large number of variables.

- **Disadvantages of Random Forest**:

1. Interpretability Challenge: May be difficult to understand how the model makes decisions due to the complexity of multiple tree structures.

2. Time and Resources: Building multiple trees and aggregating results can require time and resources.

- **The Accuracy of Random Forest in this code:** 0.981

Dataset

```
In [131]: # loading the csv data to a Pandas DataFrame
data=pd.read_csv("C:\\Users\\20212\\OneDrive\\Desktop\\Heart Attack.csv")
# print first 5 rows of the dataset
data
```

Out[131]:

	age	gender	impluse	pressurehight	pressurelow	glucose	kcm	troponin	class
0	64	1	66	160	83	160.0	1.80	0.012	negative
1	21	1	94	98	46	296.0	6.75	1.060	positive
2	55	1	64	160	77	270.0	1.99	0.003	negative
3	64	1	70	120	55	270.0	13.87	0.122	positive
4	55	1	64	112	65	300.0	1.08	0.003	negative
...	...	...	...	...	...	...	...	...	...
1314	44	1	94	122	67	204.0	1.63	0.006	negative
1315	66	1	84	125	55	149.0	1.33	0.172	positive
1316	45	1	85	168	104	96.0	1.24	4.250	positive
1317	54	1	58	117	68	443.0	5.80	0.359	positive
1318	51	1	94	157	79	134.0	50.89	1.770	positive

1319 rows × 9 columns

	A	B	C	D	E	F	G	H	I
1	age	gender	impluse	pressurehi	pressurelc	glucose	kcm	troponin	class
2	64	1	66	160	83	160	1.8	0.012	negative
3	21	1	94	98	46	296	6.75	1.06	positive
4	55	1	64	160	77	270	1.99	0.003	negative
5	64	1	70	120	55	270	13.87	0.122	positive
6	55	1	64	112	65	300	1.08	0.003	negative
7	58	0	61	112	58	87	1.83	0.004	negative
8	32	0	40	179	68	102	0.71	0.003	negative
9	63	1	60	214	82	87	300	2.37	positive
10	44	0	60	154	81	135	2.35	0.004	negative
11	67	1	61	160	95	100	2.84	0.011	negative
12	44	0	60	166	90	102	2.39	0.006	negative
13	63	0	60	150	83	198	2.39	0.013	negative
14	64	1	60	199	99	92	3.43	5.37	positive
15	54	0	94	122	67	97	1.42	0.012	negative
16	47	1	76	120	70	319	2.57	0.003	negative
17	61	1	81	118	66	134	1.49	0.017	positive
18	86	0	73	114	68	87	1.11	0.776	positive

Heart Attack

The main purpose here is to collect characteristics of Heart Attack or factors that contribute to it.

The size of the dataset is 1319 samples, which have nine fields, where eight fields are for input fields and one field for an output field.

- Age
- gender (0 for Female, 1 for Male)
- heart rate (impulse)
- systolic BP (pressurehight)
- diastolic BP (pressurelow)
- blood sugar(glucose)
- CK-MB (kcm)
- and Test-Troponin (troponin) are representing the input fields,
- while the output field pertains to the presence of heart attack (class), which is divided into two categories (negative and positive) negative refers to the absence of a heart attack, while positive refers to the presence of a heart attack.

Code

importing libraries

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Data Collection and Processing

In [4]:
```
# Loading the csv data to a Pandas DataFrame
data=pd.read_csv("C:\\Users\\20212\\OneDrive\\Desktop\\Heart Attack.csv")
# print first 5 rows of the dataset
data.head()
```

Out[4]:

	age	gender	impluse	pressurehight	pressurelow	glucose	kcm	troponin	class
0	64	1	66	160	83	160.0	1.80	0.012	negative
1	21	1	94	98	46	296.0	6.75	1.060	positive
2	55	1	64	160	77	270.0	1.99	0.003	negative
3	64	1	70	120	55	270.0	13.87	0.122	positive
4	55	1	64	112	65	300.0	1.08	0.003	negative

In [5]:
```
# getting some info about the data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1319 entries, 0 to 1318
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   age            1319 non-null   int64
 1   gender         1319 non-null   int64
 2   impluse        1319 non-null   int64
 3   pressurehight  1319 non-null   int64
 4   pressurelow    1319 non-null   int64
 5   glucose        1319 non-null   float64
 6   kcm            1319 non-null   float64
 7   troponin       1319 non-null   float64
 8   class          1319 non-null   object
dtypes: float64(3), int64(5), object(1)
memory usage: 92.9+ KB
```

In [6]:
```
# number of rows and columns in the dataset
data.shape
```

Out[6]: (1319, 9)

In [7]:
```
# checking for missing values
data.isnull().sum()
```

Out[7]:
```
age              0
gender           0
impluse          0
pressurehight    0
pressurelow      0
glucose          0
kcm              0
troponin         0
class            0
dtype: int64
```

```
In [8]: # statistical measures about the data
data.describe()
```

Out[8]:

	age	gender	impluse	pressurehight	pressurelow	glucose	kcm	troponin
count	1319.000000	1319.000000	1319.000000	1319.000000	1319.000000	1319.000000	1319.000000	1319.000000
mean	56.191812	0.659591	78.336619	127.170584	72.269143	146.634344	15.274306	0.360942
std	13.647315	0.474027	51.630270	26.122720	14.033924	74.923045	46.327083	1.154568
min	14.000000	0.000000	20.000000	42.000000	38.000000	35.000000	0.321000	0.001000
25%	47.000000	0.000000	64.000000	110.000000	62.000000	98.000000	1.655000	0.006000
50%	58.000000	1.000000	74.000000	124.000000	72.000000	116.000000	2.850000	0.014000
75%	65.000000	1.000000	85.000000	143.000000	81.000000	169.500000	5.805000	0.085500
max	103.000000	1.000000	1111.000000	223.000000	154.000000	541.000000	300.000000	10.300000

```
In [9]: #check if there is any duplicated data
data.duplicated().sum()
```

Out[9]: 0

Data preprocessing

```
In [10]: from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
encoder=preprocessing.LabelEncoder()
data['class']=encoder.fit_transform(data['class'])
```

```
In [72]: data.head(10)
```

Out[72]:

	age	gender	impluse	pressurehight	pressurelow	glucose	kcm	troponin	class
0	64	1	66	160	83	160.0	1.80	0.012	0
1	21	1	94	98	46	296.0	6.75	1.060	1
2	55	1	64	160	77	270.0	1.99	0.003	0
3	64	1	70	120	55	270.0	13.87	0.122	1
4	55	1	64	112	65	300.0	1.08	0.003	0
5	58	0	61	112	58	87.0	1.83	0.004	0
6	32	0	40	179	68	102.0	0.71	0.003	0
7	63	1	60	214	82	87.0	300.00	2.370	1
8	44	0	60	154	81	135.0	2.35	0.004	0
9	67	1	61	160	95	100.0	2.84	0.011	0

In [8]:

```
# statistical measures about the data
data.describe()
```

Out[8]:

	age	gender	impluse	pressurehight	pressurelow	glucose	kcm	troponin
count	1319.000000	1319.000000	1319.000000	1319.000000	1319.000000	1319.000000	1319.000000	1319.000000
mean	56.191812	0.659591	78.336619	127.170584	72.269143	146.634344	15.274306	0.360942
std	13.647315	0.474027	51.630270	26.122720	14.033924	74.923045	46.327083	1.154568
min	14.000000	0.000000	20.000000	42.000000	38.000000	35.000000	0.321000	0.001000
25%	47.000000	0.000000	64.000000	110.000000	62.000000	98.000000	1.655000	0.006000
50%	58.000000	1.000000	74.000000	124.000000	72.000000	116.000000	2.850000	0.014000
75%	65.000000	1.000000	85.000000	143.000000	81.000000	169.500000	5.805000	0.085500
max	103.000000	1.000000	1111.000000	223.000000	154.000000	541.000000	300.000000	10.300000

In [9]:

```
#check if there is any duplicated data
data.duplicated().sum()
```

Out[9]: 0

Data preprocessing

In [10]:

```
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
encoder=preprocessing.LabelEncoder()
data['class']=encoder.fit_transform(data['class'])
```

In [72]:

```
data.head(10)
```

Out[72]:

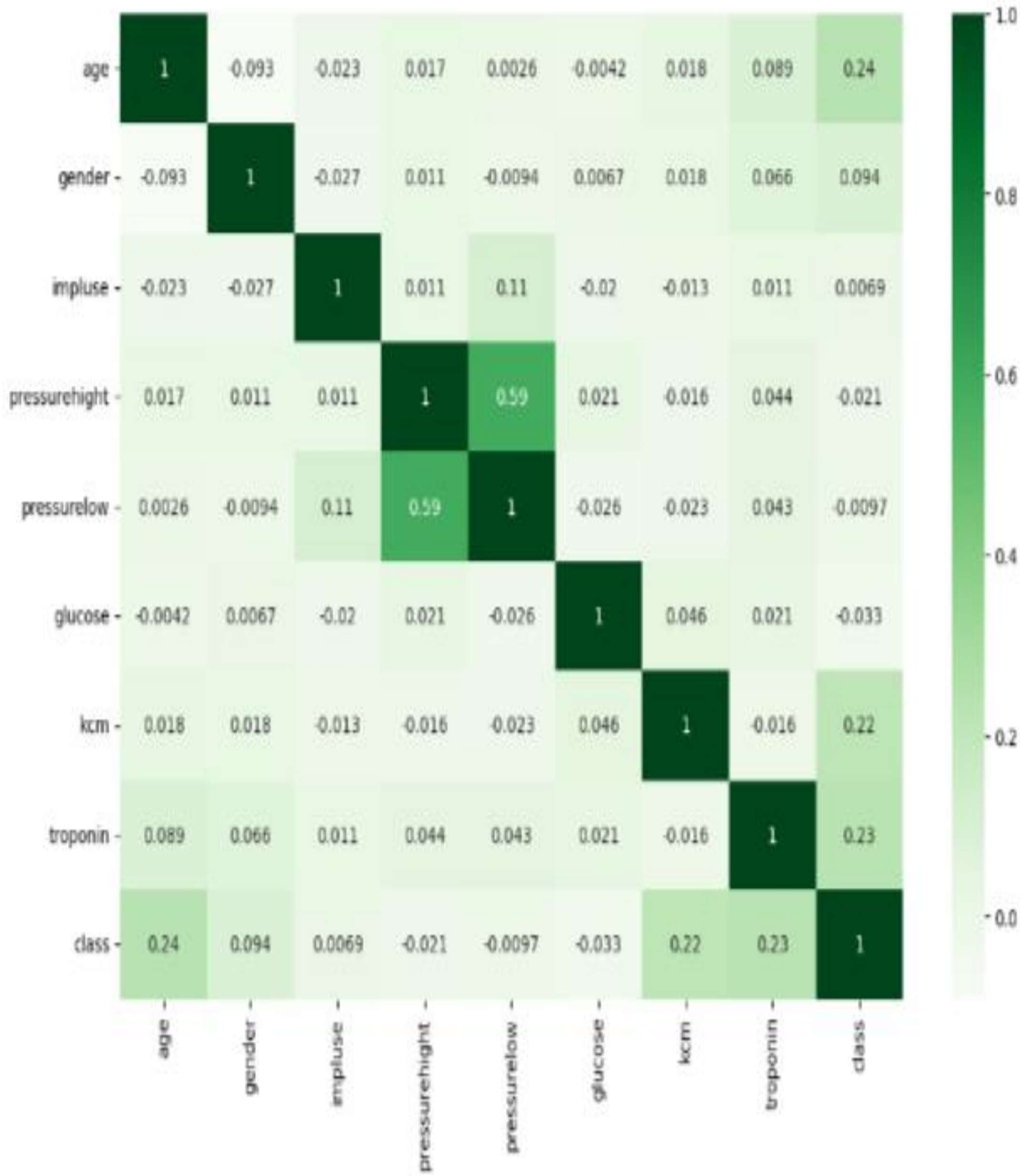
	age	gender	impluse	pressurehight	pressurelow	glucose	kcm	troponin	class
0	64	1	66	160	83	160.0	1.80	0.012	0
1	21	1	94	98	46	296.0	6.75	1.060	1
2	55	1	64	160	77	270.0	1.99	0.003	0
3	64	1	70	120	55	270.0	13.87	0.122	1
4	55	1	64	112	65	300.0	1.08	0.003	0
5	58	0	61	112	58	87.0	1.83	0.004	0
6	32	0	40	179	68	102.0	0.71	0.003	0
7	63	1	60	214	82	87.0	300.00	2.370	1
8	44	0	60	154	81	135.0	2.35	0.004	0
9	67	1	61	160	95	100.0	2.84	0.011	0

Corellation

In [12]:
```
correlation=data.corr()
```

In [13]:
```
plt.figure(figsize=(12,8))
sns.heatmap(correlation ,annot=True ,cbar=True ,cmap='Greens')
```

Out[13]: <Axes: >

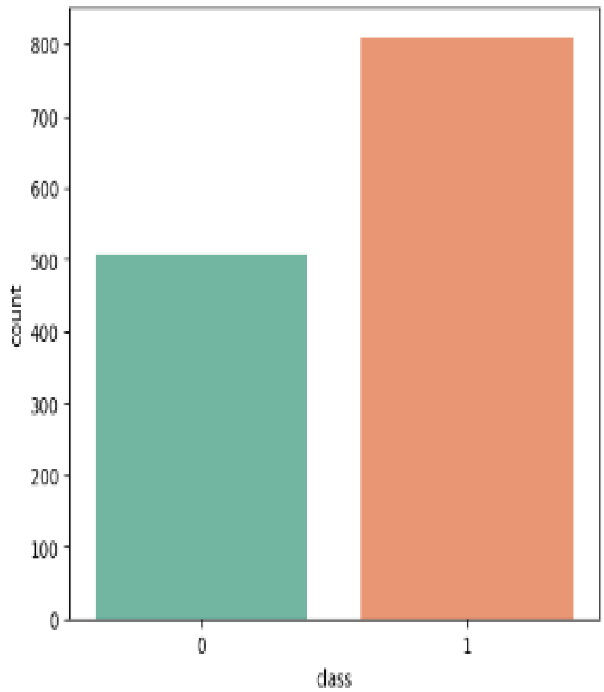


Data Analysis

In [14]:
```python
sns.countplot(data=data,x='class',palette='Set2')
data['class'].value_counts().reset_index(name='count')
```

Out[14]:

	class	count
0	1	810
1	0	509



Spliting data

In [15]:
```python
X = data.drop(["class"] , axis = 1)
y = data["class"].values
```

In [16]:
```python
from sklearn.model_selection import train_test_split
```

In [17]:
```python
X_train , X_test , y_train ,y_test = train_test_split(X,y , test_size= 0.2 , random_state= 42)
```

In [18]:
```python
X.shape,X_train.shape, X_test.shape,y.shape, y_train.shape, y_test.shape
```

Out[18]:
```
((1319, 8), (1055, 8), (264, 8), (1319,), (1055,), (264,))
```

Random Forest & Grid search

In [19]:
```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
```

In [20]:
```python
criterion = ['gini', 'entropy']
# Number of trees in random forest
n_estimators = [10, 50, 100, 250, 500]

# Maximum number of levels in tree
max_depth = [None, 2,  4, 6, 8]

# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]

# Create the random grid
param_grid  = {'criterion':criterion,
               'n_estimators': n_estimators,
               'max_depth': max_depth,
               'min_samples_leaf': min_samples_leaf
               }
print(param_grid)
```

```
{'criterion': ['gini', 'entropy'], 'n_estimators': [10, 50, 100, 250, 500], 'max_depth': [None, 2, 4, 6, 8], 'min_samples_lea
f': [1, 2, 4]}
```

In [74]:
```python
#rf = RandomForestClassifier()
rf_random = GridSearchCV(RandomForestClassifier(), param_grid )
rf_random.fit(X_train,y_train)
```

Out[74]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

In [75]:
```python
rf_random.best_params_
```

Out[75]:
```
{'criterion': 'entropy',
 'max_depth': None,
 'min_samples_leaf': 1,
 'n_estimators': 50}
```

In [76]:
```python
randomforestmodel=RandomForestClassifier(n_estimators= 50,
 min_samples_leaf = 1,
 max_depth= None,
 criterion='entropy')
```

In [84]:
```python
randomforestmodel.fit(X_train,y_train)
```

Out[84]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=50)
```

In [85]:
```python
y_pred_rf=randomforestmodel.predict(X_test)
```

In [86]:
```python
rf_accuracy = accuracy_score(y_test, y_pred_rf)
```

In [87]:
```python
rf_accuracy
```

Out[87]:
```
0.9810606060606061
```

DECISION TREE MODEL

In [28]:
```python
from sklearn.tree import DecisionTreeClassifier
```

In [29]:
```python
tree_para = {'criterion':['gini','entropy'],
             'max_depth':[1,2,3,4,5,6,7,8,9,10,11,12],
            'min_samples_leaf':range(1,5)}
```

In [30]:
```python
#DecisionTreemodel=DecisionTreeClassifier()
clf = GridSearchCV(DecisionTreeClassifier(), tree_para)
```

In [31]:
```python
#DecisionTreemodel=DecisionTreeClassifier()
clf.fit(X_train, y_train)
```

Out[31]:
```
GridSearchCV
estimator: DecisionTreeClassifier
DecisionTreeClassifier
```

In [32]:
```python
clf.best_params_
```

Out[32]:
```
{'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 3}
```

In [33]:
```python
decision_tree_model=DecisionTreeClassifier(criterion='gini',
  max_depth= 4,
  min_samples_leaf= 2)
```

In [34]:
```python
decision_tree_model.fit(X_train, y_train)
```

Out[34]:
```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=4, min_samples_leaf=2)
```

In [35]:
```python
y_pred_dt=decision_tree_model.predict(X_test)
```

In [36]:
```python
dt_accuracy = accuracy_score(y_test, y_pred_dt)
dt_accuracy
```

Out[36]:
```
0.9772727272727273
```

KNN Model

In [107]:
```python
from sklearn.neighbors import KNeighborsClassifier
```

In [111]:
```python
my_params={'n_neighbors':[3,5,7,9,1],
          'p':[1,2,3]}
```

In [112]:
```python
knn=KNeighborsClassifier()
knngrid=GridSearchCV(knn,my_params,cv=5).fit(X_train,y_train)
```

In [113]:
```python
knngrid.best_params_
```

Out[113]:
```
{'n_neighbors': 9, 'p': 2}
```

In [114]:
```python
knn_model=KNeighborsClassifier(n_neighbors = 9, p=2)
knn_model.fit(X_train, y_train)
y_pred=model.predict(X_test)
```

In [115]:
```python
accuracy = accuracy_score(y_test, y_pred)
accuracy
```

Out[115]:
```
0.625
```

Output

system evaluation

1. randomforest Model

In [116]:
```
input_data = (44,0,60,154,81,135.0,2.35,0.004)

# change the input data to a numpy array
input_data_as_numpy_array= np.asarray(input_data)

# reshape the numpy array as we are predicting for only on instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = randomforestmodel.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('The Person does not have a Heart Disease')
else:
  print('The Person has Heart Disease')
```

```
[0]
The Person does not have a Heart Disease
```

2. DECISION TREE MODEL

In [123]:
```
input_data = (63,1,60,214,82,87.0,300.00,2.370)

# change the input data to a numpy array
input_data_as_numpy_array_dt= np.asarray(input_data)

# reshape the numpy array as we are predicting for only on instance
input_data_reshaped_dt = input_data_as_numpy_array_dt.reshape(1,-1)

prediction_dt = decision_tree_model.predict(input_data_reshaped_dt)
print(prediction_dt)

if (prediction_dt[0]== 0):
  print('The Person does not have a Heart Disease')
else:
  print('The Person has Heart Disease')
```

```
[1]
The Person has Heart Disease
```

3. KNN Model

In [121]:
```
input_data = (63,1,60,214,82,87.0,300.00,2.370)

# Change the input data to a numpy array
input_data_as_numpy_array_knn = np.asarray(input_data)

# Reshape the numpy array as we are predicting for only one instance
input_data_reshaped_knn = input_data_as_numpy_array_knn.reshape(1, -1)

prediction_knn = knn_model.predict(input_data_reshaped_knn)
print(prediction_knn)

if prediction_knn[0] == 0:
    print('The Person does not have a Heart Disease')
else:
    print('The Person has Heart Disease')
```

```
[1]
The Person has Heart Disease
```

References

1. Understanding Heart Disease :

- The World Health Organization (WHO) provides clear and concise information about cardiovascular diseases (CVDs), including heart disease. You can find facts and figures on their official website: [WHO Cardiovascular Diseases] (https://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds))

2. AI and Machine Learning in Healthcare :

Read about how artificial intelligence is making a difference in healthcare:

- Esteva et al.'s paper on dermatologist-level classification of skin cancer using deep neural networks: [Link to Paper] (https://www.nature.com/articles/nature21056)

- Rajkomar et al.'s review on machine learning in medicine: [Link to Paper](https://www.nejm.org/doi/full/10.1056/NEJMra1814259)

3. Heart Disease detection Using Machine Learning :

Explore how machine learning is used to predict heart disease :

- Dey et al.'s paper on using a hybrid feature selection approach with support vector machine for heart disease prediction: [Link to Paper]

  (https://www.sciencedirect.com/science/article/pii/S001048251830472X)

- Puthiya Parambath et al.'s book chapter on heart disease prediction using machine learning algorithms.

4. Python and Machine Learning Libraries :

If you're new to Python and machine learning, these resources are great for learning:

- Python for Data Analysis" by Wes McKinney is a book that helps you get started with data analysis in Python.

- Scikit-learn is a popular machine learning library in Python. You can find documentation and tutorials on their official website: [Scikit-learn Documentation]

  (https://scikit-learn.org/stable/documentation.html)

- "Deep Learning with Python" by François Chollet is a book that introduces you to deep learning using Python.

5. Datasets for Heart Disease :

   Access datasets that can be used for heart disease prediction projects :

- The Cleveland Heart Disease dataset is a well-known dataset for heart disease detection.

- The UCI Machine Learning Repository is a repository of various datasets, including those related to heart disease. You can explore available datasets here: (http://archive.ics.uci.edu/ml)

- Kaggle , a platform for data science competitions and datasets, provides a rich source of resources, including heart disease datasets and competitions. You can view our dataset here: (https://www.kaggle.com/datasets/bharath011/heart-disease-classification-dataset)