# Heart Disease Prediction Using Machine Learning Algorithms

**By**
**Rushita Vekariya**

## ABSTRACT

Nowadays, health diseases are increasing day by day due to lifestyle, and hereditary. Especially, heart disease has become more common these days, i.e. life of people is at risk. Each individual has different values for Blood pressure, cholesterol and pulse rate. But according to medically proven results, the normal values of Blood pressure are 120/90, Cholesterol is 100-129 mg/dL, Pulse rate is 72, Fasting Blood Sugar level is 100 mg/dL, Heart rate is 60-100 bpm, ECG is normal, Width of major vessels is 25 mm (1 inch) in the aorta to only 8 μm in the capillaries. This paper gives a survey about different classification techniques used for predicting the risk level of each person based on age, gender, Blood pressure, cholesterol, and pulse rate.

A "Disease Prediction" system based on predictive modelling predicts the disease of the user based on the symptoms that the user provides as an input to the system. The system analyzes the symptoms provided by the user as input and gives the probability of the disease as an output. Disease Prediction is done by implementing 5 techniques such KNN, Decision Tree, Linear Regression and Random Forest Algorithms. These techniques calculate the probability of the disease. Therefore, an average prediction accuracy probability of 83% is obtained.

# Introduction

Heart disease is a pervasive health issue globally, necessitating effective predictive tools for early detection and intervention. In this analysis, I explored a dataset comprising clinical attributes and diagnostic indicators to develop predictive models for heart disease detection. It discusses various factors contributing to heart disease, including lifestyle choices, genetic predisposition, and environmental factors. Symptoms of heart diseases such as chest pain, breathlessness, and heart palpitations are described, along with the complexity of diagnosing heart conditions.

The use of machine learning algorithms, particularly supervised learning techniques, for heart disease prediction is emphasized. The research focuses on comparative analysis of classification algorithms such as Logistic Regression, Random Forest, Decision Tree and KNeighbors Classifier using the Stat Log dataset from the machine learning repository. The objective is to build predictive models for various types of heart diseases based on training data, enabling early detection and prevention.

The dataset contained information such as age, gender, chest pain type, blood pressure, cholesterol levels, and various other physiological parameters. After thorough data preprocessing, including handling missing values and feature engineering, I scaled the features and prepared them for model training.

Four machine learning algorithms were evaluated: Logistic regression, KNeighbors Classifier, Decision Tree Classifier, and Random Forest Classifier. Cross-validation techniques were employed to assess model performance, with accuracy as the evaluation metric.

The KNeighbors Classifier achieved the highest accuracy of 85.07% with K=12, followed by the Random Forest Classifier with an accuracy of 83.82% using 90 estimators. The Decision Tree Classifier attained an accuracy of 78.77% with a max depth of 3.

## ➢ Predictive Data Mining for Medical Diagnosis: Heart Disease Prediction

The successful application of data mining in highly visible fields like e-business, marketing and retail has led to its application in other industries and sectors However, there is a lack of effective analysis tools to discover hidden relationships and trends in data. Several Heart Disease Prediction Using Machine Learning Algorithms experiments have been conducted to compare the performance of predictive data mining techniques on the same dataset and the outcome reveals that the Decision Tree outperforms and has similar accuracy as the decision tree but other predictive methods like KNN, Neural Networks, and Classification based on clustering are not performing well. The second conclusion is that the accuracy of the Decision Tree further improves after applying a genetic algorithm to reduce the actual data size to get the optimal subset of attributes enough for heart disease prediction.

## ➢ Data Mining in the Heart Disease Prediction

Three different supervised machine learning algorithms i.e. K-NN, and the Decision Tree algorithm have been used for analyzing the dataset. Tanagra tool is used to classify the data and the data is evaluated using 10-fold cross-validation and the results are compared. Tanagra is a data mining suite built around graphical user interface algorithms. Decision Tree is a popular classifier which is simple and easy to implement. It requires no domain knowledge or parameter setting and can handle high-dimensional data. The results obtained from Decision Trees are easier to read and interpret. The drill-through feature to access detailed patients‟ profiles is only available in Decision Trees. The k-nearest neighbour's algorithm (k-NN) is a method for classifying objects based on the closest training data in the feature space. K-NN is a type of instance-based learning. The k-nearest neighbour algorithm is amongst the simplest of all machine learning algorithms. But the accuracy of the k-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance. The experiment is performed using a training data set consisting of 3000 instances with 14 different attributes. The dataset is divided into two parts that are 70% of the data are used for training and 30% are used for testing.

## ➢ Issues and Challenges

Medical diagnosis is considered a significant yet intricate task that needs to be carried out precisely and efficiently. The automation of the same would be highly beneficial. Clinical decisions are often made based on a doctor's intuition and experience rather than on the knowledge-rich data hidden in the database. This practice leads to unwanted biases, errors and excessive medical costs which affects the quality of service provided to patients. Data mining has the potential to generate a knowledge-rich environment which can help to significantly improve the quality of clinical decisions.

# Dataset

## ➢ Dataset description

The dataset contains 303 rows and 14 columns.

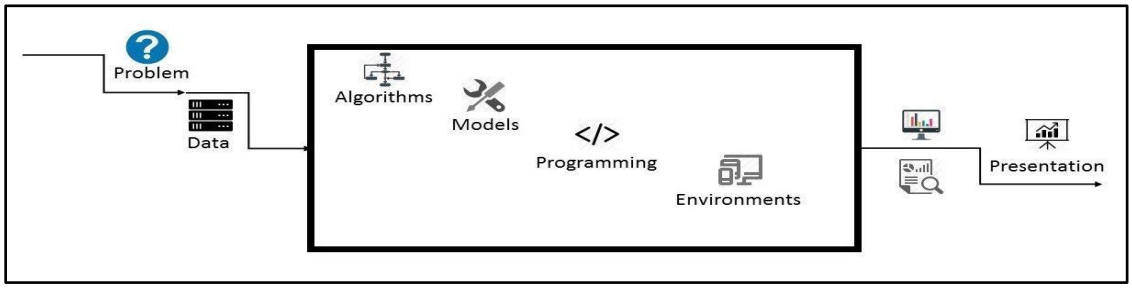
The columns include features such as age, sex, chest pain type (cp), resting blood pressure (trestbps), serum cholesterol level (chol), fasting blood sugar (fbs), rest ECG results (restecg), maximum heart rate achieved (thalach), exercise-induced angina (exang), ST depression induced

by exercise relative to rest (oldpeak), the slope of the peak exercise ST segment (slope), number of major vessels coloured by fluoroscopy (ca), thalassemia type (thal), and the target variable indicating the presence of heart disease.

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 |

*Snapshot of Dataset*

## Proposed Method and Architecture




*Data Science Project Architecture*

### ➢ Problem Statement

Heart Diseases affect a large population in today's world, where the lifestyle is moved from active to comfort-oriented. We live in an era of fast food. Which build up cholesterol, diabetes and many more factors which in turn affect the heart in some way or the other. According to the World Health Organization Cardiovascular Diseases (CVD) or Heart Diseases cause more deaths than any other diseases globally[99]. The amount of data in medical sectors is quite large and computerized as well. They are not utilized or put to any use. This data if studied and analyzed could be put to good use like prediction of diseases or even prevent them. Diseases such as cancer can be detected, and the stage can also be predicted by training datasets with pictures of cancer cells. Similarly, heart disease can be predicted based on aspects like cholesterol, diabetes, heart rate etc. The prediction of heart disease is a challenge and very risky. We observed that in some cases solutions to problems do not rely on a single method.

It varies from situation to situation. It is also a challenge as most of the data are sparse or missing as they were not stored in the motive of analyzing. We therefore set out a goal to find

which method would be best for predicting the diseases using data from four different hospitals from four different places. This is a comparative study on the efficiency of different data mining techniques such as Logical Regression, Random forest, K-Nearest Neighbors, and Decision Tree in predicting heart diseases. The Data Mining techniques are analyzed, and the accuracy of prediction is noted for each method used. The result showed that heart diseases can be predicted with an accuracy of above 90%. Cardiovascular diseases are the leading cause of death globally, resulting in 17.9 million deaths (32.1%) in 2015, up from 12.3 million (25.8%) in 1990. It is estimated that 90% of CVD is preventable. There are many risk factors for heart disease that we will take a closer look at. The main objective of this study is to build a model that can predict the heart disease occurrence, based on a combination of features (risk factors) describing the disease. Different machine learning techniques will be implemented and compared to standard performance metrics such as accuracy. The dataset used for this study was taken from the UCI machine learning repository.

## ➢ Algorithms

In this analysis, I employ three different machine-learning algorithms for heart disease prediction. Each algorithm offers unique advantages and is evaluated based on its performance in accurately classifying individuals as either having or not having heart disease.

## ➢ Model Building

**Logistic Regression:**
Logistic regression is a statistical method used for binary classification tasks, where the target variable has two possible outcomes. An accuracy score of 85.25% suggests that the model correctly predicts the class of approximately 85.25% of the instances in the test dataset.

**KNeighbors Classifier Model:**
Cross-validation was used to find the optimal number of neighbours (K) for the KNeighbors Classifier. The KNeighbors Classifier was trained with K=12, achieving an accuracy of 85.07%.

**Decision Tree Classifier:**
Cross-validation was used to find the optimal depth of the decision tree. The Decision Tree Classifier was trained with a maximum depth of 3, achieving an accuracy of 78.77%.

**Random Forest Classifier:**
Cross-validation was used to find the optimal number of estimators (N) for the Random Forest Classifier. The Random Forest Classifier was trained with N=90, achieving an accuracy of 83.82%.

## ➢ Programming and Environment

The programming language used for this analysis is Python. Python is a versatile and widely used programming language with extensive libraries and tools for data analysis, machine learning, and scientific computing. It offers simplicity, readability, and a rich ecosystem of packages such as pandas, numpy, scikit-learn, and matplotlib, which are essential for data manipulation, modelling, and visualization.

# Methodology

The methodology section outlines the steps followed in the analysis, from data preprocessing to model evaluation. It provides a systematic approach to ensure reproducibility and clarity in the analysis process. The methodology for this heart disease prediction project includes the following steps:

1. Data Collection
2. Data Preprocessing
3. Exploratory Data Analysis (EDA)
4. Feature Engineering
5. Model Development
6. Model Evaluation
7. Model Deployment
8. Conclusion and Recommendations

# Implementation and Analysis

The implementation and analysis phase involves executing the developed methodology and interpreting the results obtained from the models. This phase focuses on applying the trained models to real-world data and evaluating their performance for heart disease prediction. The following steps outline the implementation and analysis process

## ➢ Dataset description

```
In [14]: df.describe().T
```

Out[14]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| age | 303.0 | 54.366337 | 9.082101 | 29.0 | 47.5 | 55.0 | 61.0 | 77.0 |
| sex | 303.0 | 0.683168 | 0.466011 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 |
| cp | 303.0 | 0.966997 | 1.032052 | 0.0 | 0.0 | 1.0 | 2.0 | 3.0 |
| trestbps | 303.0 | 131.623762 | 17.538143 | 94.0 | 120.0 | 130.0 | 140.0 | 200.0 |
| chol | 303.0 | 246.264026 | 51.830751 | 126.0 | 211.0 | 240.0 | 274.5 | 564.0 |
| fbs | 303.0 | 0.148515 | 0.356198 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| restecg | 303.0 | 0.528053 | 0.525860 | 0.0 | 0.0 | 1.0 | 1.0 | 2.0 |
| thalach | 303.0 | 149.646865 | 22.905161 | 71.0 | 133.5 | 153.0 | 166.0 | 202.0 |
| exang | 303.0 | 0.326733 | 0.469794 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 |
| oldpeak | 303.0 | 1.039604 | 1.161075 | 0.0 | 0.0 | 0.8 | 1.6 | 6.2 |
| slope | 303.0 | 1.399340 | 0.616226 | 0.0 | 1.0 | 1.0 | 2.0 | 2.0 |
| ca | 303.0 | 0.729373 | 1.022606 | 0.0 | 0.0 | 0.0 | 1.0 | 4.0 |
| thal | 303.0 | 2.313531 | 0.612277 | 0.0 | 2.0 | 2.0 | 3.0 | 3.0 |
| target | 303.0 | 0.544554 | 0.498835 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 |

```
In [13]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

The dataset used in this project contains 14 variables. The independent variable that needs to be predicted, 'diagnosis', determines whether a person is healthy or suffers from heart disease. Experiments with the Cleveland database have concentrated on endeavours to distinguish disease presence (values 1, 2, 3, 4) from absence (value 0). There are several missing attribute values, distinguished with the symbol '?'. The header row is missing in this dataset, so the column names have to be inserted manually.

**Features information:**

- age - age in years
- sex - sex (1 = male; 0 = female)
- chest pain - chest pain type (1 = typical angina; 2 = atypical angina; 3 = nonanginal
- pain; 4 = asymptomatic)
- blood pressure - resting blood pressure (in mm Hg on admission to the hospital)
- serum cholesterol - serum cholesterol in mg/dl
- fasting blood sugar - fasting blood sugar > 120 mg/dl (1 = true; 0 = false)
- electrocardiographic - resting electrocardiographic results (0 = normal; 1 = having ST-T; 2 = hypertrophy)
- max heart rate - maximum heart rate achieved
- induced angina - exercise-induced angina (1 = yes; 0 = no)
- ST depression - ST depression induced by exercise relative to rest
- slope - the slope of the peak exercise ST segment (1 = upsloping; 2 = flat; 3 = downsloping)
- no of vessels - number of major vessels (0-3) coloured by fluoroscopy
- thalassemia - 3 = normal; 6 = fixed defect; 7 = reversible defect
- diagnosis - the predicted attribute - diagnosis of heart disease (angiographic disease status) (Value 0 = < 50% diameter narrowing; Value 1 = > 50% diameter narrowing)

## ➢ Types of features

**Categorical features** (Has two or more categories and each value in that feature can be categorized by them): **sex, chest pain**
**Ordinal features** (Variable having relative ordering or sorting between the values): **fasting blood sugar, electrocardiographic, induced angina, slope, no of vessels, thalassemia, diagnosis**
**Continuous features** (Variable-taking values between any two points or between the minimum
or maximum values in the feature column): **age, blood pressure, serum cholesterol, max heart rate, ST depression**

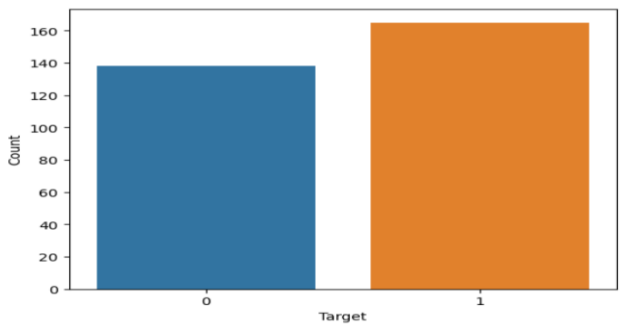
## ➢ Check for missing Data

```
In [12]: df.isnull().any()

Out[12]: age         False
         sex         False
         cp          False
         trestbps    False
         chol        False
         fbs         False
         restecg     False
         thalach     False
         exang       False
         oldpeak     False
         slope       False
         ca          False
         thal        False
         target      False
         dtype: bool
```

No Data is missing, which is good.

## ➢ Exploratory Data Analysis (EDA)

```
In [17]: # Visualization to check if the dataset is balanced or not
         g = sns.countplot(x='target', data=df)
         plt.xlabel('Target')
         plt.ylabel('Count')

Out[17]: Text(0, 0.5, 'Count')
```

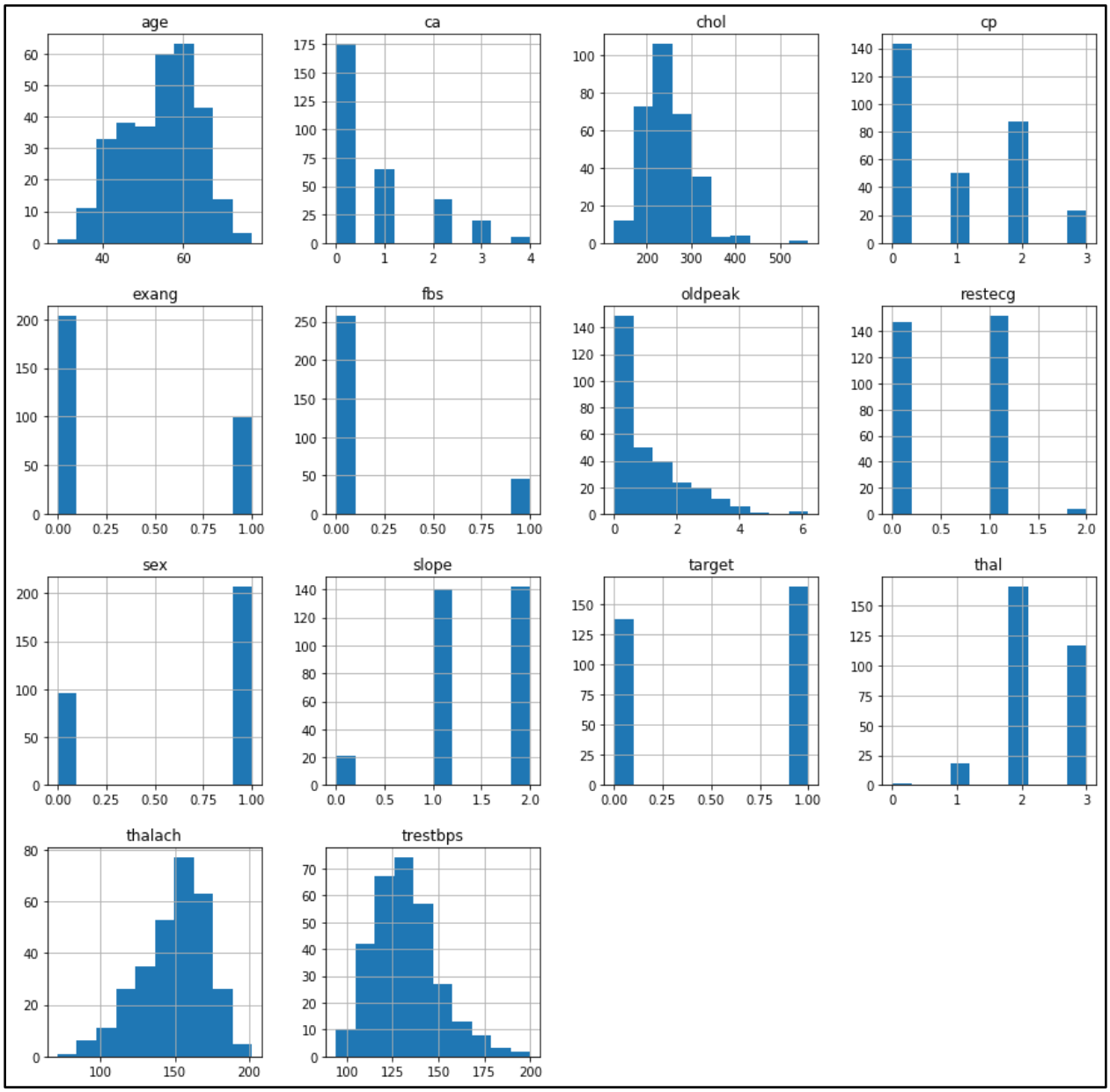



(1 is who have Heart Disease and 0 is who don't have Heart Disease)

No. of Heart Disease patients is 165. No. of patients who don't have a heart disease is 138. [Which is a good balance of target data.]

## ➢ Histogram plot

Uses Matplotlib and Seaborn to visualize the dataset. Plot histograms for each numeric feature. Plots a count plot to visualize the distribution of the target variable ('target')



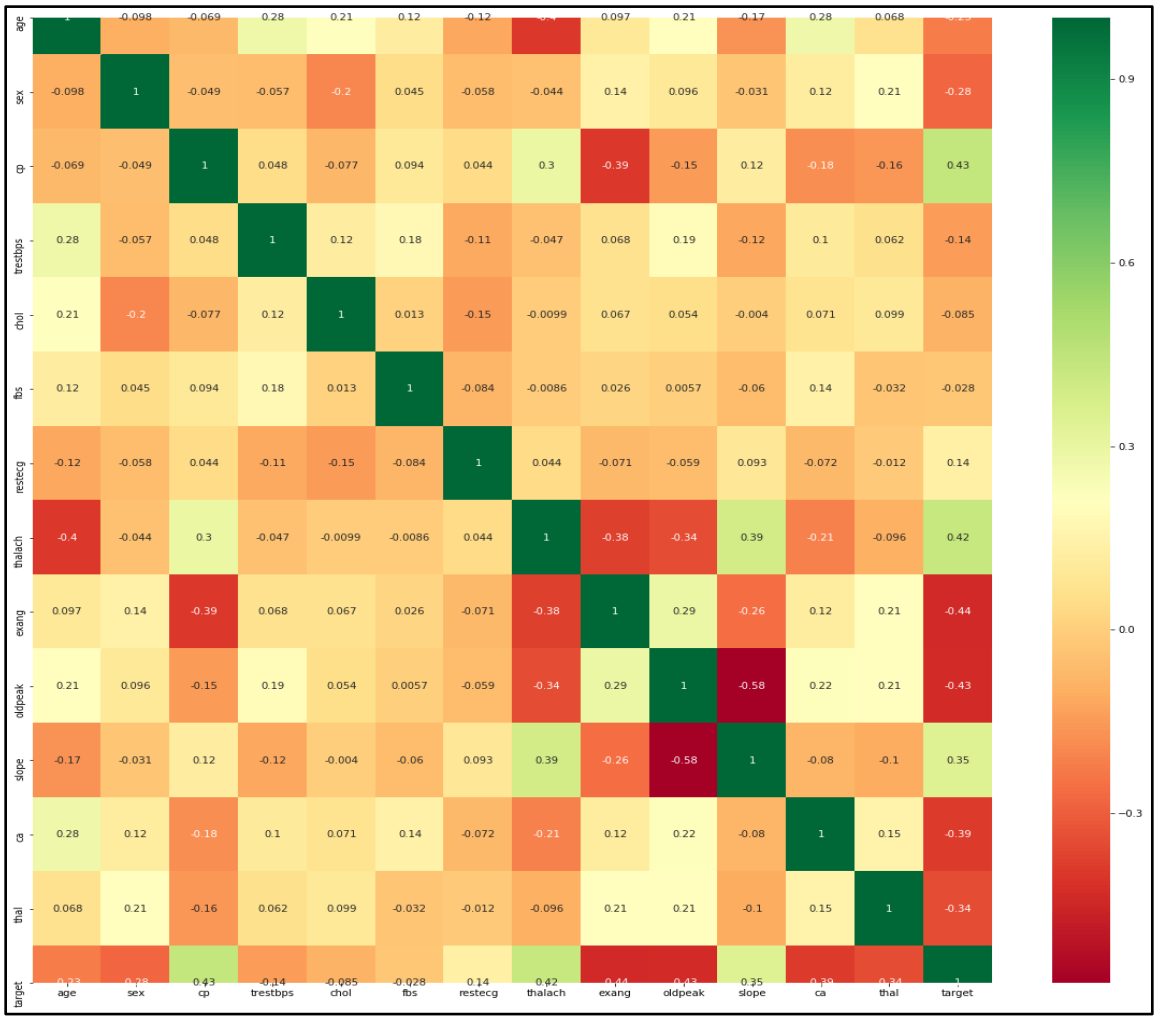
## ➢ Feature Engineering:

### Correlation Matrix

Correlation analysis is a method of statistical evaluation used to study the strength of a relationship between two, numerically measured, continuous variables (e.g. height and weight)

In [14]:
```
# Selecting correlated features using Heatmap

# Get correlation of all the features of the dataset
corr_matrix = df.corr()
top_corr_features = corr_matrix.index

# Plotting the heatmap
plt.figure(figsize=(20,20))
sns.heatmap(data=df[top_corr_features].corr(), annot=True, cmap='RdYlGn')
```

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x27c550e7b88>



# Modelling and predicting with Machine Learning

The main goal of the entire project is to predict heart disease occurrence with the highest accuracy. To achieve this, I will test several classification algorithms. This section includes all results obtained from the study and introduces the best performer according to the accuracy metric. I have chosen several algorithms typical for solving supervised learning problems through classification methods.

First of all, let's equip ourselves with a handy tool that benefits from the cohesion of the SciKit Learn library and formulate a general function for training our models. The reason for displaying accuracy on both, train and test sets, is to allow us to evaluate whether the model overfits or underfits the data (so-called bias/variance tradeoff).

```
In [24]: def train_model(X_train, y_train, X_test, y_test, classifier, **kwargs):

             """
             Fit the chosen model and print out the score.

             """

             # instantiate model
             model = classifier(**kwargs)

             # train model
             model.fit(X_train,y_train)

             # check accuracy and print out the results
             fit_accuracy = model.score(X_train, y_train)
             test_accuracy = model.score(X_test, y_test)

             print(f"Train accuracy: {fit_accuracy:0.2%}")
             print(f"Test accuracy: {test_accuracy:0.2%}")

             return model
```

## ➢ Logistic Regression

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.

Types of logical regression:

- Binary (Pass/Fail)
- Multi (Cats, Dogs, Sheep)

**For Multiclass** - Instead of y=0,1 I will expand our definition so that y=0,1...n. I re-run binary classification multiple times, once for each class.

**Procedure** –
1. Divide the problem into n+1 binary classification problems (+1 because the index starts at 0).
2. For each class…
3. Predict the probability the observations are in that single class.
4. prediction = <math>max (probability of the classes)

```
In [27]: # Logistic Regression
         from sklearn.linear_model import LogisticRegression
         model = train_model(X_train, Y_train, X_test, Y_test, LogisticRegression)

         Train accuracy: 83.88%
         Test accuracy: 85.25%
```

The accuracy score of Logistic Regression is: 85.25%

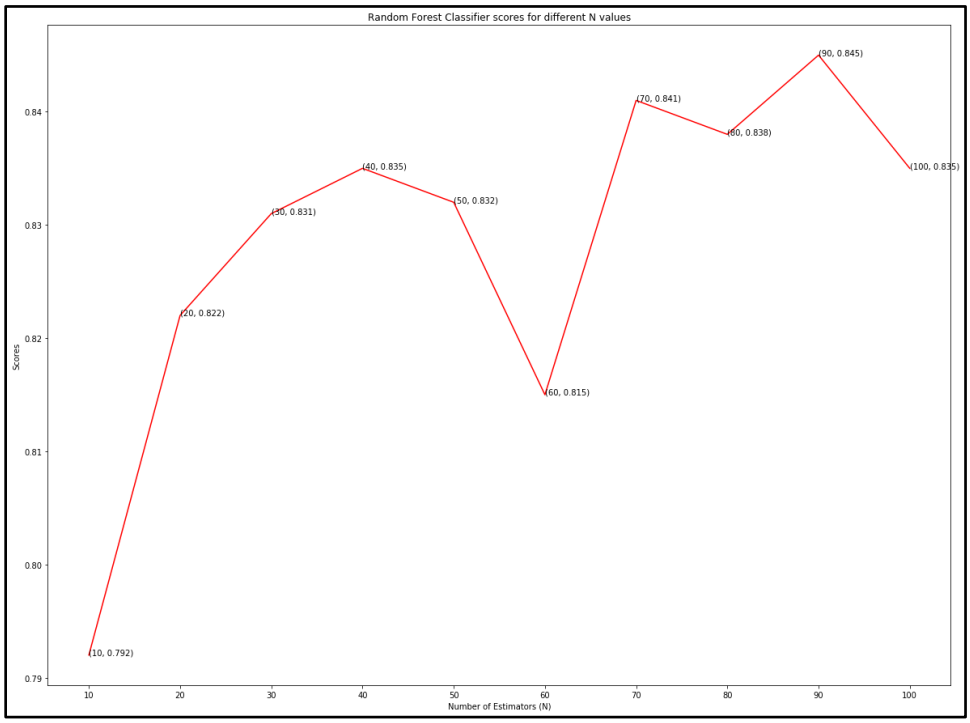
## ➢ Random Forest

Random Forest is a supervised learning algorithm. Random forest can be used for both classification and regression problems, by using a random forest regressor we can use random forest on regression problems. However, we have used random forest on classification in this project so we will only consider the classification part.

**Random Forest pseudocode**
1. Randomly select "k" features from the total "m" features.
   Where k << m
2. Among the "k" features, calculate the node "d" using the best split point.
3. Split the node into daughter nodes using the best split.
4. Repeat 1 to 3 steps until the "l" number of nodes has been reached.
5. Build a forest by repeating steps 1 to 4 for "n" number times to create "n" number of trees.

**Random forest prediction pseudocode**
1. Takes the test features and uses the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target)
2. Calculate the votes for each predicted target.
3. Consider the high-voted predicted target as the final prediction from the random forest algorithm.

```
In [32]: # Importing essential libraries
         from sklearn.ensemble import RandomForestClassifier

In [33]: # Finding the best accuracy for random forest algorithm using cross_val_score
         forest_scores = []
         for i in range(10, 101, 10):
           forest_classifier = RandomForestClassifier(n_estimators=i)
           cvs_scores = cross_val_score(forest_classifier, X, y, cv=5)
           forest_scores.append(round(cvs_scores.mean(),3))

In [34]: # Plotting the results of forest_scores
         plt.figure(figsize=(20,15))
         plt.plot([n for n in range(10, 101, 10)], forest_scores, color = 'red')
         for i in range(1,11):
             plt.text(i*10, forest_scores[i-1], (i*10, forest_scores[i-1]))
         plt.xticks([i for i in range(10, 101, 10)])
         plt.xlabel('Number of Estimators (N)')
         plt.ylabel('Scores')
         plt.title('Random Forest Classifier scores for different N values')
```




```
In [35]:  # Training the random forest classifier model with n value as 90
          forest_classifier = RandomForestClassifier(n_estimators=90)
          cvs_scores = cross_val_score(forest_classifier, X, y, cv=5)
          print("Random Forest Classifier Accuracy with n_estimators=90 is: {}%".format(round(cvs_scores.mean(), 4)*100))

          Random Forest Classifier Accuracy with n_estimators=90 is: 84.46000000000001%
```

The accuracy score of Random Forest is 84.46%

## ➢ K-Nearest Neighbor

We can implement a KNN model by following the below steps:

1. Load the data
2. Initialize the value of k
3. For getting the predicted class, iterate from 1 to the total number of training data points

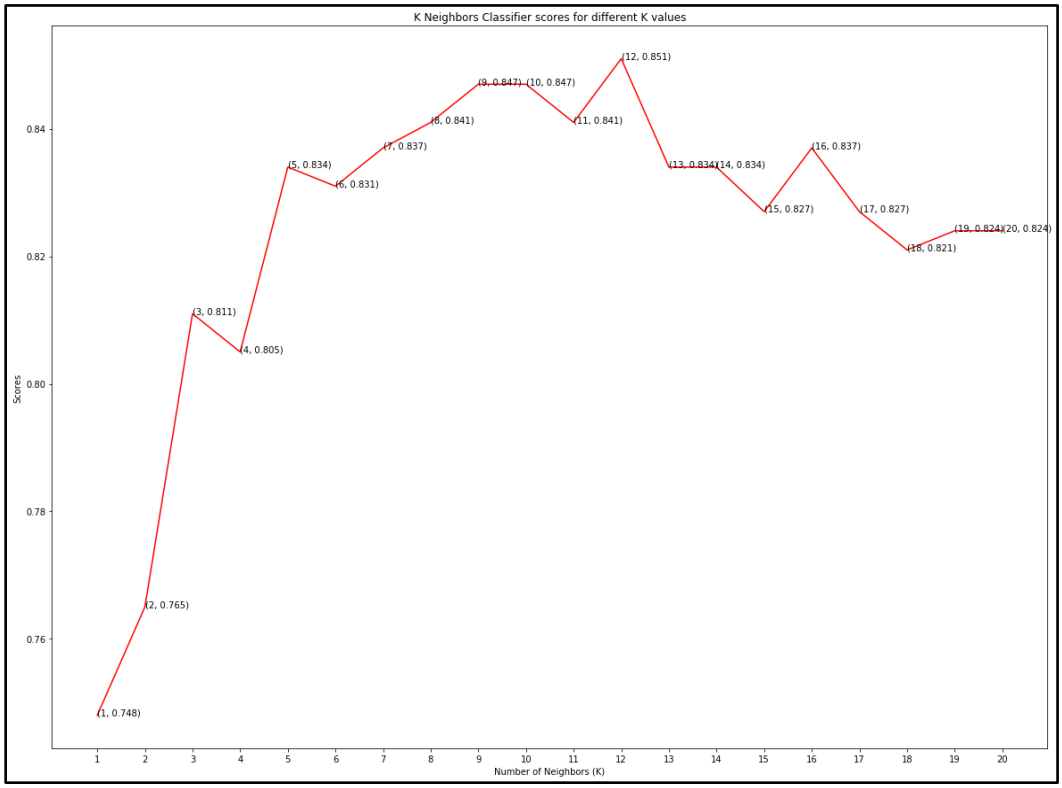
- Calculate the distance between test data and each row of training data. Here
- we will use Euclidean distance as our distance metric since it's the most
- popular method. The other metrics that can be used are Chebyshev, cosine, etc.
- Sort the calculated distances in ascending order based on distance values
- Get the top k rows from the sorted array
- Get the most frequent class of these rows
- Return the predicted class

In [24]:
```
# Importing essential libraries
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
```

In [25]:
```
# Finding the best accuracy for knn algorithm using cross_val_score
knn_scores = []
for i in range(1, 21):
  knn_classifier = KNeighborsClassifier(n_neighbors=i)
  cvs_scores = cross_val_score(knn_classifier, X, y, cv=10)
  knn_scores.append(round(cvs_scores.mean(),3))
```

In [26]:
```
# Plotting the results of knn_scores
plt.figure(figsize=(20,15))
plt.plot([k for k in range(1, 21)], knn_scores, color = 'red')
for i in range(1,21):
    plt.text(i, knn_scores[i-1], (i, knn_scores[i-1]))
plt.xticks([i for i in range(1, 21)])
plt.xlabel('Number of Neighbors (K)')
plt.ylabel('Scores')
plt.title('K Neighbors Classifier scores for different K values')
```

Out[26]: Text(0.5, 1.0, 'K Neighbors Classifier scores for different K values')




In [27]:
```
# Training the knn classifier model with k value as 12
knn_classifier = KNeighborsClassifier(n_neighbors=12)
cvs_scores = cross_val_score(knn_classifier, X, y, cv=10)
print("KNeighbours Classifier Accuracy with K=12 is: {}%".format(round(cvs_scores.mean(), 4)*100))
```

```
KNeighbours Classifier Accuracy with K=12 is: 84.48%
```

## ➢ Decision Tree

**Pseudocode**
1. Place the best attribute of the dataset at the root of the tree.
2. Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for an attribute.
3. Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree.

**Assumptions while creating Decision Tree**

- At the beginning, the whole training set is considered as the root.
- Feature values are preferred to be categorical. If the values are continuous then they
- are discretized before building the model.
- Records are distributed recursively based on attribute values.
- Order to place attributes as root or internal node of the tree is done by using some
- statistical approach.

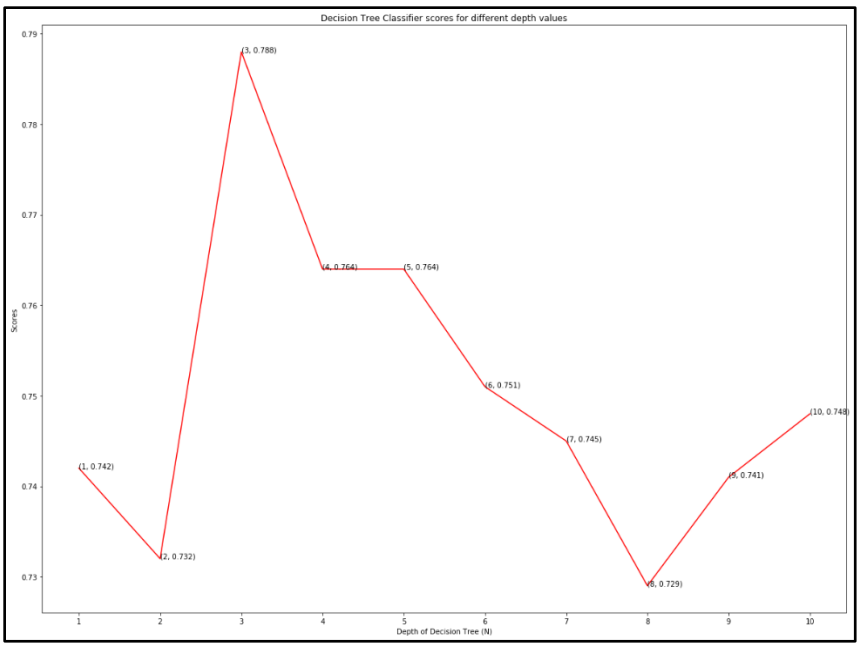
**The popular attribute selection measures**

- Information gain
- Gini index

**Attribute selection method** - A dataset consists of "n" attributes then deciding which attribute to place at the root or different levels of the tree as internal nodes is a complicated step. Just randomly selecting any node to be the root can't solve the issue. If we follow a random approach, it may give us bad results with low accuracy. To solve this attribute selection problem, researchers worked and devised some solutions. They suggested using some criteria like information gain, Gini index, etc. These criteria will calculate values for every attribute. The values are sorted, and attributes are placed in the tree by following the order i.e., the attribute with a high value (in case of information gain) is placed at the root. While using information Gain as a criterion, we assume attributes to be categorical, and for the Gini index, attributes are assumed to be continuous.

**Gini Index** - The Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified. It means an attribute with a lower Gini index should be preferred.

```
In [28]: # Importing essential libraries
         from sklearn.tree import DecisionTreeClassifier

In [29]: # Finding the best accuracy for decision tree algorithm using cross_val_score
         decision_scores = []
         for i in range(1, 11):
           decision_classifier = DecisionTreeClassifier(max_depth=i)
           cvs_scores = cross_val_score(decision_classifier, X, y, cv=10)
           decision_scores.append(round(cvs_scores.mean(),3))

In [30]: # Plotting the results of decision_scores
         plt.figure(figsize=(20,15))
         plt.plot([i for i in range(1, 11)], decision_scores, color = 'red')
         for i in range(1,11):
             plt.text(i, decision_scores[i-1], (i, decision_scores[i-1]))
         plt.xticks([i for i in range(1, 11)])
         plt.xlabel('Depth of Decision Tree (N)')
         plt.ylabel('Scores')
         plt.title('Decision Tree Classifier scores for different depth values')

Out[30]: Text(0.5, 1.0, 'Decision Tree Classifier scores for different depth values')
```




We set max depth=3, limiting the depth of the tree to decrease overfitting. This leads to a lower accuracy on the training set, but an improvement on the test set.

In [31]:
```
# Training the decision tree classifier model with max_depth value as 3
decision_classifier = DecisionTreeClassifier(max_depth=3)
cvs_scores = cross_val_score(decision_classifier, X, y, cv=10)
print("Decision Tree Classifier Accuracy with max_depth=3 is: {}%".format(round(cvs_scores.mean(), 4)*100))
```

```
Decision Tree Classifier Accuracy with max_depth=3 is: 78.51%
```

# Result

**FINAL SCORE**

In [52]:
```
# initialize an empty list
accuracy = []

# list of algorithms names
classifiers = ['KNN', 'Decision Trees', 'Logistic Regression', 'Random Forests']

# list of algorithms with parameters
models = [KNeighborsClassifier(n_neighbors=8), DecisionTreeClassifier(max_depth=3, random_state=0), LogisticRegression(), Ran

# loop through algorithms and append the score into the list
for i in models:
    model = i
    model.fit(X_train, Y_train)
    score = model.score(X_test, Y_test)
    accuracy.append(score)
```

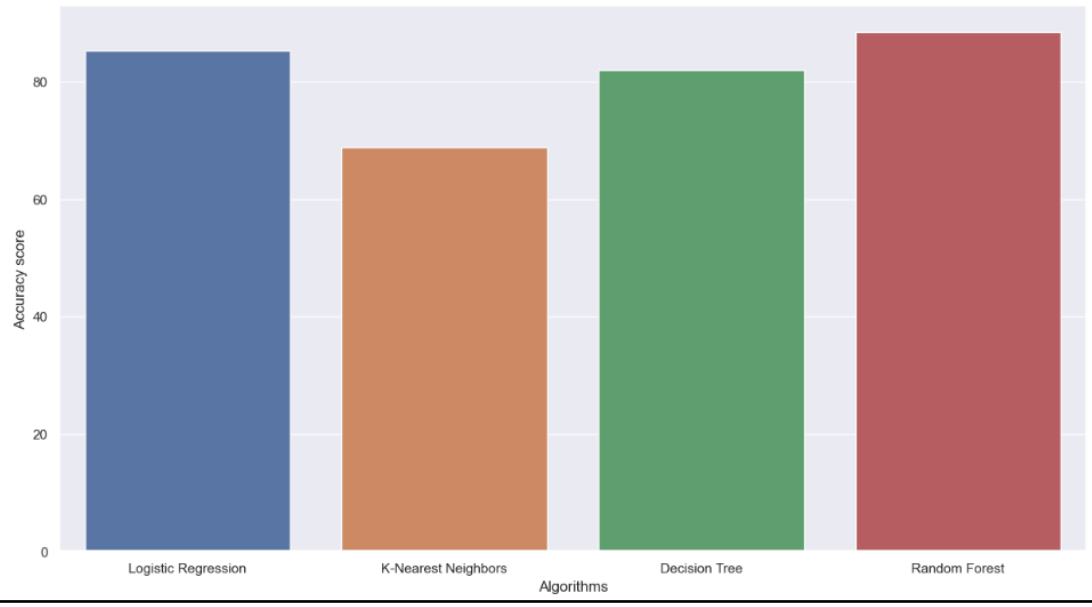
```
In [53]: # create a dataframe from accuracy results
         summary = pd.DataFrame({'accuracy':accuracy}, index=classifiers)
         summary
```

Out[53]:

| | accuracy |
|---|---|
| **KNN** | 0.688525 |
| **Decision Trees** | 0.819672 |
| **Logistic Regression** | 0.852459 |
| **Random Forests** | 0.885246 |

```
In [55]: scores = [score_lr, score_knn, score_dt, score_rf]
         algorithms = ["Logistic Regression", "K-Nearest Neighbors", "Decision Tree", "Random Forest"]
         sns.set(rc={'figure.figsize': (15, 8)})
         plt.xlabel("Algorithms")
         plt.ylabel("Accuracy score")

         sns.barplot(x=algorithms, y=scores)
```

Out[55]: <Axes: xlabel='Algorithms', ylabel='Accuracy score'>




## Conclusion

The overall objective of our project is to predict accurately with less number of tests and attributes the presence of heart disease. In this project, fourteen attributes are considered which form the primary basis for tests and give accurate results more or less. Many more input attributes can be taken but our goal is to predict with less number of attributes and faster efficiency to predict the risk of having heart disease at a particular age span. Five data mining classification techniques were applied namely K-Nearest Neighbor, Decision Tree, Random Forest & Logistic Regression. It is shown that Random Forest has better accuracy than the other techniques.

This is the most effective model to predict patients with heart disease. This project could answer complex queries, each with its strength concerning ease of model interpretation, access to detailed information and accuracy.

This project can be further enhanced and expanded. For example, it can incorporate other medical attributes besides the 14 attributes we used. It can also incorporate other data mining techniques, e.g., Time Series, Clustering and Association Rules. Continuous data can also be used instead of just categorical data. Another area is to use Text Mining to mine the vast amount of unstructured data available.

This project is presented using data mining techniques. From logistic regression, KNN, Decision Tree, and Random forest are used to develop the system. Random Forest provides better results and assists the domain experts and even the person related to the medical field to plan for a better and early diagnosis for the patient. This system performs realistically well even without retraining.

## Drawbacks

The Algorithms used in our project do not give 100% accuracy, so the prediction is not 100% feasible. Clinical diagnosis and diagnosis using our project may differ slightly because the prediction is not 100% accurate. Medical diagnosis is considered a significant yet intricate task that needs to be carried out precisely and efficiently. The automation of the same would be highly beneficial. Clinical decisions are often made based on a doctor's intuition and experience rather than on the knowledge-rich data collected from the dataset.