# CHAPTER 1: HEART DISEASE PREDICTION

## 1.1 ABSTRACT

This project focuses on developing a heart disease prediction system using the powerful logistic regression algorithm. The objective is to construct a machine learning model that accurately predicts the likelihood of an individual having heart disease based on a range of health parameters, such as age, cholesterol level, and blood pressure. Leveraging Logistic Regression, a well-suited supervised learning algorithm for binary classification, the model is trained on a dataset containing patients' medical details and their heart disease status. Rigorous data preprocessing is applied, including analysis and normalization, to ensure the data's compatibility with the logistic regression model. The dataset is then divided into training and testing sets to evaluate the model's accuracy.

The logistic regression classifier is employed to categorize individuals into those likely to have heart disease and those not. This predictive capability establishes a robust mechanism for early detection and intervention. The project's significance lies in its potential to support healthcare professionals by providing a tool for the timely identification of individuals at risk of heart disease. The outlined workflow serves as a comprehensive guide for developing the Heart Disease Prediction System, contributing to advancements in healthcare through the application of machine learning.

## 1.2 INTRODUCTION

Heart disease is a major global health concern, contributing significantly to morbidity and mortality. Early detection and prediction of heart disease can be crucial in preventing its progression and improving patient outcomes. In this context, the Heart Disease Prediction System project aims to develop a robust and accurate system for predicting the likelihood of heart disease in individuals.

## 1.3 METHODOLOGY

First, we look into the workflow of our Heart disease Prediction.

A diagram of data processing

Description automatically generated

*Figure 1.3.1.1: Work Flow.*

### 1.3.1 Work Flow Explanation

**Heart data:**

This data set contains several health parameters that correspond to a person's healthiness of the heart.

**Data preprocessing:**

We need to process this data set; we cannot feed this raw data into our machine-learning algorithm.

We need to process this data set to make it fit and be compatible with our machine-learning algorithm.

**Test and train:**

Once we process the data, we need to split our data into training data and testing data.

This is because we often train our machine learning algorithm with training data, and we will evaluate the performance of our model using the test data. This part is called a **strain test split.**

We will split our original data set into training data and test data.

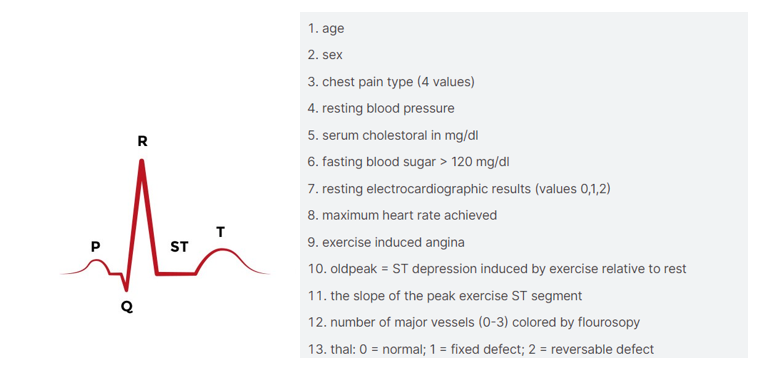
**Logistic regression model:**

We will feed our training data to our machine-learning model. In this case, we are going to use a logistic regression model because this particular use case is a binary classification. So, we are going to classify whether a person has a deceased heart or not.

In those binary classifications, the logistic regression model is very useful. So, it's the best model when it comes to binary classification. Once we train this logistic regression model with our training data, we get trained in logistic regression mode.

**Trained logistic regression model:**

When we feed new data, our model can predict whether the person has heart disease or not.



*Figure 1.3.1.2: Heart ECG signal.*

* If a person has a healthy heart, the ECG signal of the heart is shown in the first image.
* From that ECG signal, we add features or variables to our project. That is shown in the second image.

Now we will develop a heart disease prediction model.

### 1.3.2 Libraries

A screen shot of a computer

Description automatically generated

*Figure 1.3.2.1: Imported Libraries.*

**Numpy :**

The libraries include NumPy for arrays.

**Pandas:**

Pandas for creating structured tables (data frames)

**scikit-learn:**

* scikit-learn for model selection and evaluation. The logistic regression model is chosen for this project, and its implementation is facilitated by importing 'LogisticRegression' from the scikit-learn linear model module. Additionally, we import the 'train\_test\_split' function for data splitting and the 'accuracy\_score' for evaluating the model's performance.
* From this model, we are importing train\_test\_split and logistic regression.
* The accuracy score is used to evaluate our model and check how well it is performing.

### 1.3.3 Data Collection and Processing

After importing the necessary libraries, the next step is to load the dataset into a Pandas data frame.

heart\_data = pd.read\_csv('heart.CSV)

* For that, we are using **Pd.read\_csv for loading CSV data into a Pandas Dataframe.**

heart\_data.head()

heart\_data.tail()

heart\_data.shape

* To load sample data from a CSV file, use the keywords **head() and tail().** This can get sample data from a CSV file.
* To get the number of rows and columns in the CSV file, we use the keyword “**shape”.**
* To get information from data, we use the keyword **info().**
* We get the info as below image.

heart\_data.info()

A screenshot of a computer program

Description automatically generated

*Figure 1.3.3.1: CSV file information.*

* Here, we observe 1025 entries, 14 columns, and 1025 non-null values.
* Null values mean missing values; non-null values mean normal values. Dtype represents the data type.
* To check missing values, the keyword we used is heart\_data. **isnull(). sum().**
* It gives us the number of missing values in each column.

To load statistical measures of the table, we use the keyword **describe()**

heart\_data.describe()

The 'describe' function provides statistical measures for each column, including count (number of data points), mean (average), standard deviation (a measure of data dispersion), minimum values for each column, 25th percentile, 50th percentile (median), 75th percentile, and maximum values for each column.

A screen shot of a black screen

Description automatically generated

*Figure 1.3.3.2: Statistical measures.*

Checking the distribution of target variables to see how many people with heart disease.

heart\_data['target'].value\_counts()

From the above keyword, we get a count of heart disease people in the given data set.

### 1.3.4 Splitting the Features and Target

From the above steps, we process the data. Now we have to split the data into training data and testing data.

We are splitting the data into **features** and **targets**.

X = heart\_data.drop(columns='target', axis=1)

Y = heart\_data ['target’]

Here we dropped the target column from our data set using the keyword **drop(),** and axis = 1**represents the** column.

## 1.3.5 Splitting the data into training data and test data

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, stratify=Y, random\_state=2)

**train\_test\_split:**

* This function is part of scikit-learn's **model\_selection** module and is used to split the dataset into training and testing sets.

**X and Y:**

* **X** typically represents the feature matrix (independent variables), and **Y** represents the target variable (dependent variable) in a machine learning model.

**test\_size=0.2:**

* This parameter specifies the proportion of the dataset to include in the test split. In this case, it's set to 0.2, indicating that 20% of the data will be used for testing and the remaining 80% will be used for training.

**stratify=Y:**

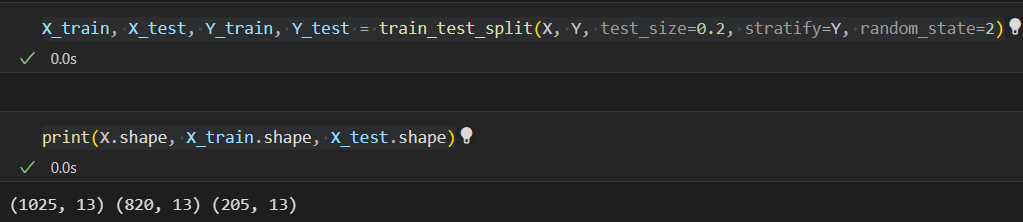
* The **stratify** parameter ensures that the splitting process maintains the same distribution of the target variable **Y** in both the training and testing sets. This is particularly useful when dealing with imbalanced datasets. If you don’t mention stratify = y, there is a possibility that all the values in the x test may contain 0 or all the values may contain 1.

**random\_state=2:**

* **random\_state** is used to set a seed for reproducibility. By providing a fixed seed (in this case, 2), the randomness in the splitting process is controlled, resulting in the same train-test split whenever the code is executed. This is useful for reproducibility in machine-learning experiments.

**X\_train, X\_test, Y\_train, Y\_test:**

* These variables store the resulting training and testing sets for features (**X\_train** and **X\_test**) and target variables (**Y\_train** and **Y\_test**).



*Figure 1.3.5.1: Splitting of data.*

* It split X into 80% training data and 20% testing data.

### 1.3.6 Model Training with Logistic Regression

In the process of model training for logistic regression, the algorithm learns from a set of labeled data (features and corresponding outcomes). Using techniques like gradient descent, the model adjusts its parameters to minimize the difference between its predictions and the actual outcomes, effectively learning the patterns associated with heart disease.

**model = LogisticRegression()**

The line **model = LogisticRegression()** initializes a Logistic Regression model, creating a blank slate for subsequent training and prediction tasks in your code.

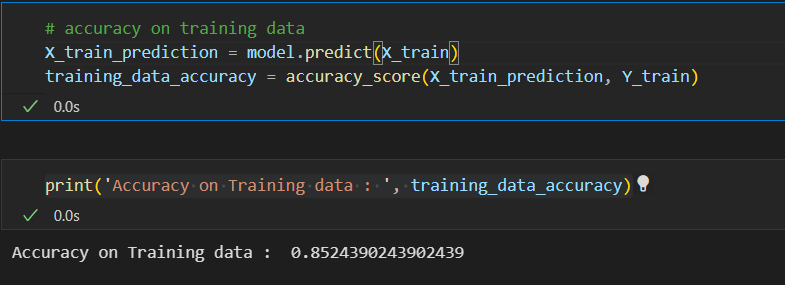
**model.fit(X\_train, Y\_train)**

**model. fit(X\_train, Y\_train)** is like the phase where our Logistic Regression model learns from examples. The **X\_train** contains the features or details about heart health (like age, cholesterol, etc.), and the **Y\_train** has the corresponding outcomes (whether a person has heart disease or not). The **fit** process is where the model analyzes these examples, figuring out the relationships between features and outcomes so that it can later predict whether new individuals are likely to have heart disease based on similar characteristics.

### 1.3.7 Model Evaluation with Accuracy Score

Once the logistic regression model is trained, it is essential to evaluate its performance. The accuracy score is a metric that measures the proportion of correctly predicted instances. It is calculated by dividing the number of correct predictions by the total number of predictions. While accuracy provides an overall performance assessment, it's important to consider other metrics, especially in imbalanced datasets, to gain a comprehensive understanding of the model's effectiveness.

### 1.3.7.1 Accuracy Score of training data

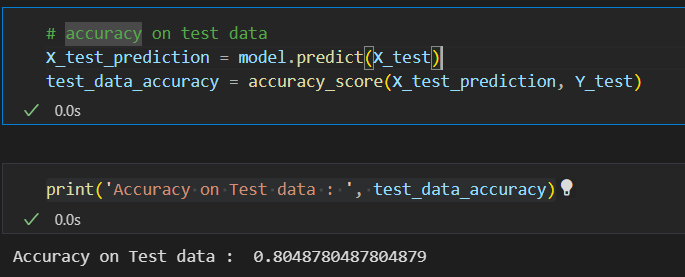


*Figure 1.3.7.1.1: Training data Accuracy.*

* **model.predict(X\_train)**: The model makes predictions on the training data (**X\_train**), trying to guess whether each person has heart disease or not based on the features in the training set.
* **accuracy\_score(X\_train\_prediction, Y\_train)**: The accuracy score is then calculated by comparing these predictions (**X\_train\_prediction**) to the actual outcomes in the training data (**Y\_train**). It measures the proportion of correct predictions out of the total predictions, giving you an idea of how well your model is performing on the data it was trained on.

If we print, the training data accuracy score is found to be 85%.

### 1.3.7.2 Accuracy Score of testing data

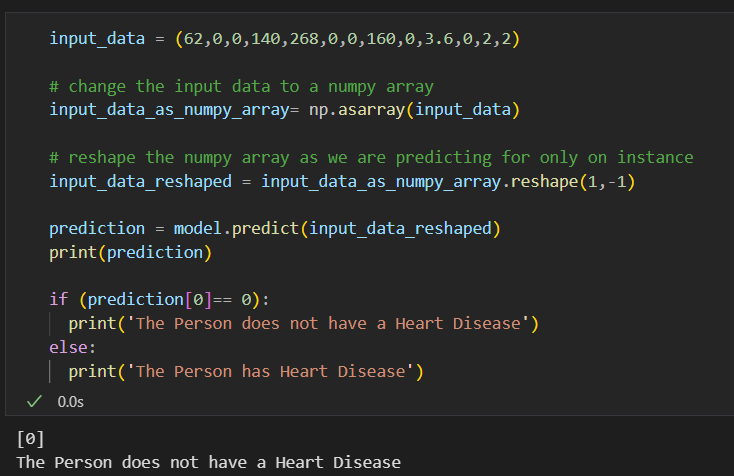


*Figure 1.3.7.2.1: Testing data Accuracy.*

* **model.predict(X\_test)**: The model makes predictions on the test data (**X\_test**), attempting to forecast whether each individual in the test set has heart disease or not based on the learned patterns.
* **accuracy\_score(X\_test\_prediction, Y\_test)**: The accuracy score is then computed by comparing these predictions (**X\_test\_prediction**) to the actual outcomes in the test data (**Y\_test**). This score indicates how well your model generalizes to new, unseen data, helping you assess its overall performance.

If we print, the training data accuracy score is found to be 80%.

## 1.4 EXPERIMENTS AND RESULTS



*Figure 1.4.1: Experiment output.*



*Figure 1.4.2: Streamlit output.*

## 1.5 CONCLUSION

In conclusion, the Heart Disease Prediction System employing Logistic Regression stands as a promising tool for early detection and intervention in cardiovascular health. The meticulously crafted model, trained on a comprehensive dataset and utilizing robust preprocessing techniques, demonstrates the potential of machine learning in assisting healthcare professionals. By accurately predicting the likelihood of heart disease based on individual health parameters, the system holds the key to proactive and personalized care. Moving forward, this project not only contributes to advancements in predictive healthcare but also underscores the significance of integrating machine learning into medical practices for improved patient outcomes and early disease management.

## 1.6 REFERENCES

[1] [https://www.youtube.com/watch?v=qmqCYC-MBQo&list=PPSV](https://www.youtube.com/watch?v=qmqCYC-MBQo&list=PPSV%20)

[2] <https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset>

[3] <https://github.com/g-shreekant/Heart-Disease-Prediction-using-Machine-Learning>