COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY
KOCHI 682 022

DEPARTMENT OF COMPUTER APPLICATIONS

General Disease Prediction System

A PROJECT REPORT

Submitted by

KUMAR SHUBHAM
SAURABH KUMAR
SITANSHU SHEKHAR
RAJ KISHORE YADAV

in partial fulfillment for the award of the degree of

Master of Computer Applications

June 2022

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY
KOCHI 682 022

CERTIFICATE

This is to certify that this project report entitled "General Disease Prediction System" *is a bonafide record of work done by* "**Kumar Shubham, Saurabh Kumar, Raj Kishore Yadav, Sitanshu Shekhar**" *towards the partial fulfillment of the requirements for the award of Master of Computer Application degree at Department of Computer Application, Cochin University of Science and Technology*.

SIGNATURE

Dr. Judy MV

HEAD OF THE DEPARTMENT

Department of Computer Applications

CUSAT

SIGNATURE

Dr. Rafidha Rehiman KA

PROJECT GUIDE

Department of Computer Applications

CUSAT

DECLARATION

We KUMAR SHUBHAM (Register Number : 38119034), SAURABH KUMAR (Register Number : 38119048), RAJ KISHORE YADAV (Register Number : 38119043), SITANSHU SHEKHAR (Register Number : 38119052) hereby declare that the Report entitled "General Disease Prediction System" in partial fulfillment of the requirements for the award of the Degree of Master of Computer Application is a record of bonafide work done by us during the period from March 2022 to May 2022 under the supervision and guidance of Dr. Rafidha Rehiman K.A at Department of Computer Applications , CUSAT.

Signature of the Candidates

ACKNOWLEDGEMENT

With great pleasure we hereby acknowledge that the help given to by various individuals throughout the project itself is an acknowledgement to the inspiration driven and technical assistance contributed by many individuals.

We extend our sincere thanks to all the non-teaching staff for providing the necessary facilities and help. We thank lord god, almighty for his immeasurable blessing upon my life. We are pleased our indebtedness to Dr. Judy M V., Head of the Department, Department of Computer Applications, CUSAT for his gracious encouragement.

And also, we are thankful to our project guide Dr. Rafidha Rehiman K A, for their beloved support. We are obliged to the teaching staff for being helpful and co-operative during the period of project.

Last but not the least, we would like to extend our warm regards to our families and peers who have kept supporting us and always had faith in our work.

Raj Kishore Yadav.

Reg.No:- 38119043

Saurabh Kumar.

Reg.No:- 38119048

Kumar Shubham.

Reg.No:- 38119034

Sitanshu Shekhar.

Reg.No:- 38119052

TABLE OF CONTENTS

ABSTRACT

The project aims to develop a web application for “General Disease Prediction System”. Our project integrates a user friendly platform to cross validate results at the go and to spread general awareness and provide precautionary measures. With the advancement in technologies and mobile phones being the most used user-friendly device, our team has come with an application that provides a prediction of the two most caused lifestyle diseases like Diabetes and Heart Disease at your hand. General Disease Prediction System (GDPS) allows you to make important predictions about the severity of an ongoing disease with few inputs given by user of parameters. Today, time is the factor and being healthy is also an essential so this idea will help users and also encourage the idea of using Internet more widely.

CHAPTER – 1: INTRODUCTION

1.1 Project Title

General Disease Prediction System (GDPS).

1.2 Purpose

Our proposed General Disease Prediction System (GDPS) is for those who often feel reluctant to go to hospital or physician on minor symptoms. However, in many cases, these minor symptoms may trigger major health hazards. As online health advice is easily reachable, using GDPS can be a great head start for users. Moreover, existing online health care systems suffer from lack of reliability and accuracy. Herein, we propose a system that relies on guided user input. The system takes input from the user and provides a report of user's condition on mentioned diseases. Before doing anything we did a decent research on rapid escalation of Internet technology and data for which handheld devices has opened up new avenues for online healthcare system.

1.3 Objectives

The objectives of this study are summarized below:

1.3.1 General Objective

-To implement support vector machine or SVM and logistic regression that predicts the seriousness disease as per the input entered by the user.

1.3.2 Specific Objective

-The main objective of the project is to design and develop a user friendly, efficient and web application based General Disease Prediction System (GDPS).

-A reliable system to spread awareness among users.

-Computerization can be helpful as means of saving time.

CHAPTER – 2: SYSTEM REQUIREMENTS

2.1 Hardware Interface

- RAM : 4 GB RAM
- PROCESSOR : Intel Core i3 CPU and above.
- DISPLAY : 720 Display
- HARD DISK : 1TB
- PROCESSOR SPEED : 2Ghz

2.2 SOFTWARE INTERFACE

Windows-10

Windows 10 is a series of operating systems developed by Microsoft and released as part of its Windows NT family of operating systems. Windows 10 was made available for download via MSDN and Technet, and as a free upgrade for retail copies of Windows 8 via the Windows Store. Windows 10 receives new builds on an ongoing basis, which are available at no additional cost to users, in addition to additional test builds of Windows 10, which are available to Windows Insiders. Devices in enterprise environments can receive these updates at a slower pace, or use long-term support milestones that only receive critical updates, such as security patches, over their ten-year lifespan of extended support.

Anaconda

Anaconda is a free and open-source distribution of the Python and R programming languages for data science and machine learning applications (large-scale data processing, predictive analytics, scientific computing), that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution is used by over 6 million users and includes more than 250 popular data-science packages suitable for Windows, Linux, and MacOS.

Jupyter Notebook

Jupyter is a non-profit organization created to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages." Spun-off from IPython in 2014 by Fernando Pérez project Jupyter supports execution environments in several dozen languages. Pandas library in computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three- clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

Pandas library

In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

Flask

Flask is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries.[2] It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

CHAPTER – 3: LITERATURE SURVEY

3.1 Heart Disease

Khaled Mohamad Almustafa et al. [1] highlighted that heart disease becomes one of the common diseases, and early diagnosis for this disease is challenging for healthcare providers. In this research, the researchers implemented various classifiers for classifying the heart disease dataset for predicting heart disease with minimal attributes. They have collected the dataset from Cleveland, Switzerland, that contains 76 characteristics with a class attribute of 1025 patients. Out of the 76 attributes in this work, using only 14 features. The researcher used various algorithms, including a k-nearest neighbor, decision tree, Naïve Bayes, SVM, stochastic gradient descent for best classification, and predicting heart disease cases. Using these classification algorithms, the researcher got accuracy results for KNN (k = 1), JRip a Decision tree with 99.70,97.26%, and 98.04, respectively.

Juan-Jose Beunza et al. [2] highlighted the usage of a supervised machine learning algorithm for predicting clinical events for its validity and accuracy. In this work, the data were brought from Framingham heart study data which contains 4240 observations. For this research, they focused on risk factors of heart disease with a combination of data mining. The researcher used different machine learning algorithms along with RapidMiner and R-Studio for analyzing the data. A neural network model was implemented to omit all the missing values when the AUC is 0.71. Later they used the same data by using RapidMiner and support vector machines and they got an AUC of 0.75.

Jiyang Want et al. [3] have proposed that reliable and effective load forecasting is one of the important factors for operation decisions and power system planning. The safety and economic operation of the power system are directly affected by forecasting accuracy. Due to the complexity and instability of power load, forecasting accuracy is a most challenging issue. Hence the researcher proposed a novel hybrid system for designing forecasting by embedding a multi-objective module.

3.2 Diabetes Disease

Qawqzeh et al. [4] proposed a logistic regression model based on photoplethysmogram analysis for diabetes classification. They used 459 patients' data for training and 128 data points to test and validate the model. Their proposed system correctly classified 552 persons as nondiabetic and achieved an accuracy of 92%. However, the proposed technique is not compared with state-of-the-art techniques. Pethunachiyar [5] presented a diabetes mellitus classification system using a machine learning algorithm. Mainly, he used a support vector machine with different kernel functions and diabetes data from the UCI Machine Repository. He found SVM with linear function more efficient than naïve Bayes, decision tree, and neural networks. Nevertheless, the state-of-the-art comparison is missing and parameter selection is not elaborated.

Ahuja et al. [6] performed a comparative analysis of various machine learning approaches, i.e., NB, DT, and MLP, on the PIMA dataset for diabetic classification. They found MLP superior as compared to other classifiers. The authors suggested that the performance of MLP can be enhanced by fine-tuning and efficient feature engineering. Recently, Mohapatra et al. [7] have also used MLP to classify diabetes and achieved an accuracy of 77.5% on the PIMA dataset but failed to perform state-of-the-art comparisons. MLP has been used in the literature for various healthcare disease classifications such as cardiovascular and cancer classification.

CHAPTER – 4: PROBLEM STATEMENT & DISCUSSION

4.1Problem Statement

It is estimated that more than 70% of people in India are prone to heart disease, diabetes and there is crude prevalence rate of 14.1 per 100,000 in Parkinson in every year. This may because many people don't realize that the general body diseases could be symptoms to something more harmful, 25% of the population succumbs to death because of ignoring the early general body symptoms. Hence identifying or predicting the disease at the earliest is very pivotal to avoid any unwanted casualties.

4.2Discussion

The purpose of this system is to provide prediction for the general and more commonly occurring disease that when unchecked can turn into fatal disease. The system applies support vector machine or SVM and logistic regression algorithms. This system will predict the severity of mentioned diseases based on the given inputs and will show precautionary measures required to avoid the aggression of disease. It will also help the doctors to analyse the pattern of presence of diseases in the society. In this project, the disease prediction system will be trained using machine learning.

CHAPTER – 5: MACHINE LEARNING

The term **Machine Learning** was coined by **Arthur Samuel** in 1959, an American pioneer in the field of computer gaming and artificial intelligence, and stated that "it gives computers the ability to learn without being explicitly programmed".
And in 1997, **Tom Mitchell** gave a "well-posed" mathematical and relational definition that A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

Machine Learning is the latest buzzword floating around. It deserves to, as it is one of the most interesting subfields of Computer Science.

Within the field of data analytics, machine learning is used to devise complex models and algorithms that lend themselves to prediction; in commercial use, this is known as predictive analytics. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data set (input).

5.1 Classification of Machine Learning

Machine learning implementations are classified into three major categories, depending on the nature of the learning "signal" or "response" available to a learning system which is as follows:-

1. **Supervised learning:** When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of Supervised learning. This approach is indeed similar to human learning under the supervision of a teacher. The teacher provides good examples for the student to memorize, and the student then derives general rules from these specific examples.

2. **Unsupervised learning:** Whereas when an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of un-correlated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.
   As a kind of learning, it resembles the methods humans use to figure out that certain objects or events are from the same class, such as by observing the degree of similarity between objects. Some recommendation systems that you find on the web in the form of marketing automation are based on this type of learning.

3. **Reinforcement learning:** When you present the algorithm with examples that lack labels, as in unsupervised learning. However, you can accompany an example with positive or negative feedback according to the solution the algorithm proposes comes under the category of Reinforcement learning, which is connected to applications for which the algorithm must make decisions (so the product is prescriptive, not just descriptive, as in unsupervised learning), and the decisions bear consequences. In the human world, it is just like learning by trial and error. Errors help you learn because they have a penalty added (cost, loss of time, regret, pain, and so on), teaching you that a certain course of action is less likely to succeed than others. An interesting example of reinforcement learning occurs when computers learn to play video games by themselves.

4. **Semi-supervised learning:** where an incomplete training signal is given: a training set with some (often many) of the target outputs missing. There is a special case of this principle known as Transduction where the entire set of problem instances is known at learning time, except that part of the targets are missing.

5.2 List of Popular Machine Learning Algorithms

1. Linear Regression

To understand the working functionality of this algorithm, imagine how you would arrange random logs of wood in increasing order of their weight. There is a catch; however – you cannot weigh each log. You have to guess its weight just by looking at the height and girth of the log (visual analysis) and arrange them using a combination of these visible parameters. This is what linear regression in machine learning is like.

In this process, a relationship is established between independent and dependent variables by fitting them to a line. This line is known as the regression line and represented by a linear equation Y= a *X + b.

In this equation:

- Y – Dependent Variable
- a – Slope
- X – Independent variable
- b – Intercept

The coefficients a & b are derived by minimizing the sum of the squared difference of distance between data points and the regression line.

2. Logistic Regression

Logistic Regression is used to estimate discrete values (usually binary values like 0/1) from a set of independent variables. It helps predict the probability of an event by fitting data to a logit function. It is also called logit regression.

These methods listed below are often used to help improve logistic regression models:

- include interaction terms
- eliminate features

- regularize techniques
- use a non-linear model

3. Decision Tree

Decision Tree algorithm in machine learning is one of the most popular algorithm in use today; this is a supervised learning algorithm that is used for classifying problems. It works well classifying for both categorical and continuous dependent variables. In this algorithm, we split the population into two or more homogeneous sets based on the most significant attributes/ independent variables.

4. SVM (Support Vector Machine) Algorithm

SVM algorithm is a method of classification algorithm in which you plot raw data as points in an n-dimensional space (where n is the number of features you have). The value of each feature is then tied to a particular coordinate, making it easy to classify the data. Lines called classifiers can be used to split the data and plot them on a graph.

5. Naive Bayes Algorithm

A Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Even if these features are related to each other, a Naive Bayes classifier would consider all of these properties independently when calculating the probability of a particular outcome.

A Naive Bayesian model is easy to build and useful for massive datasets. It's simple and is known to outperform even highly sophisticated classification methods.

6. KNN (K- Nearest Neighbors) Algorithm

This algorithm can be applied to both classification and regression problems. Apparently, within the Data Science industry, it's more widely used to solve classification problems. It’s a simple algorithm

that stores all available cases and classifies any new cases by taking a majority vote of its k neighbors. The case is then assigned to the class with which it has the most in common. A distance function performs this measurement.

KNN can be easily understood by comparing it to real life. For example, if you want information about a person, it makes sense to talk to his or her friends and colleagues!

Things to consider before selecting K Nearest Neighbours Algorithm:

- KNN is computationally expensive
- Variables should be normalized, or else higher range variables can bias the algorithm
- Data still needs to be pre-processed.

7. K-Means

It is an unsupervised learning algorithm that solves clustering problems. Data sets are classified into a particular number of clusters (let's call that number K) in such a way that all the data points within a cluster are homogenous and heterogeneous from the data in other clusters.

How K-means forms clusters:

- The K-means algorithm picks k number of points, called centroids, for each cluster.
- Each data point forms a cluster with the closest centroids, i.e., K clusters.
- It now creates new centroids based on the existing cluster members.
- With these new centroids, the closest distance for each data point is determined. This process is repeated until the centroids do not change.

8. Random Forest Algorithm

A collective of decision trees is called a Random Forest. To classify a new object based on its attributes, each tree is classified, and the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

Each tree is planted & grown as follows:

- If the number of cases in the training set is N, then a sample of N cases is taken at random. This sample will be the training set for growing the tree.
- If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M, and the best split on this m is used to split the node. The value of m is held constant during this process.
- Each tree is grown to the most substantial extent possible. There is no pruning.

CHAPTER – 6: PROPOSED SOLUTION & RESULT ANALYSIS

6.1 Proposed Solution

Concerning to the problem stated above we are going to implement support vector machine or SVM and logistic regression that predicts the seriousness of selected disease as per the input entered by the user. Besides this we aim to design and develop a user friendly and efficient web application based disease prediction System.

6.1.1 Feasibility Analysis

Technical feasibility

The project is technically feasible as it can be built using the existing available technologies. It is a web based applications that uses flask Framework. The technology required by GDPS is available and hence it is technically feasible.

Economic feasibility

The project is economically feasible as there is no cost involve in the project. As the data samples increases, our system will get more efficient.

Operational feasibility

The project is operationally feasible as the user having basic knowledge about computer and Internet. General Disease Prediction System is based on client-server architecture where client is users and server is the machine.

6.1.2 Requirement Analysis

Functional requirements

6.1.2.1 Predict disease with the given inputs by user.

6.1.2.2 Compare the given inputs with the input datasets and predicts the severity of disease.

Non-functional requirements

a. Display whether the user is affected by particular disease or not.

6.1.3 System Design

Methodology

Disease Prediction has been already implemented using different techniques like Neural Network, decision tree and various algorithms. So, our disease predictor uses SVM and logistic regression algorithms for the prediction of different diseases.

Data collection

Data collection has been done from the internet to identify the disease here the real symptoms of the disease are collected i.e. no dummy values are entered. The symptoms of the disease are collected from different health related websites.

Algorithm implemented

The algorithm implemented in this project is support vector machine or SVM and logistic regression algorithms.

6.2 Result Analysis

The system will initially be fed data from different sources, the data will then be pre-processed before further process is carried out, this is done to get clean data from the raw initial data, as the raw data would be noisy, or flawed. This data will be processed using algorithms, the system and will be trained to predict the disease based on the input data given by the user.

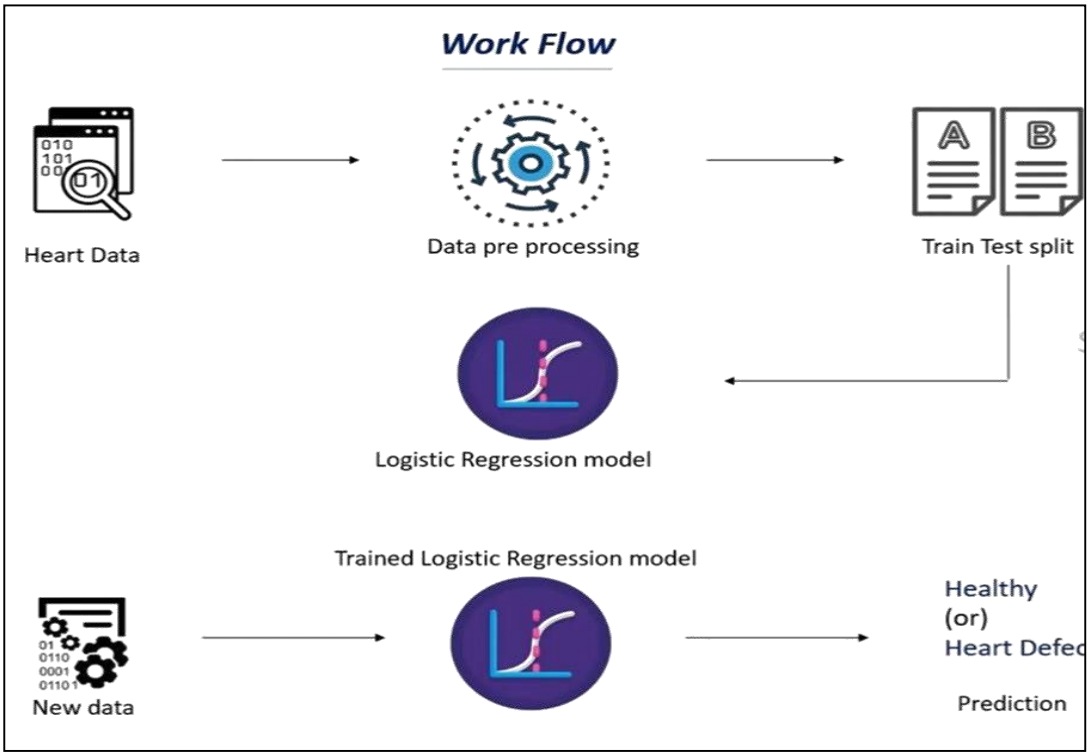
The work flow of Heart disease is as follows:

The term “heart disease” refers to several types of heart conditions. The most common type of heart disease in the United States is coronary artery disease (CAD), which affects the blood flow to the heart. Decreased blood flow can cause a heart attack. Your risk for heart disease increases with age, especially with people of colour and for those who are over 65. While the average age for a heart attack is 64.5 for men, and 70.3 for women, nearly 20 percent of those who die of heart disease are under the age of 65.

To diagnose this we are developing a system using machine learning algorithm which is Logistic Regression. At first we collect the heart data (about the patient who have Heart disease and who doesn’t

have Heart disease). This dataset contains several health parameters which correspond to a person's healthiness of the heart. Once we have this dataset we need to process this dataset because we can't feed this raw data into our machine learning algorithm. We need to process this dataset to make it fit & compatible for our machine learning algorithm to learn. Once we process the data we need to split our data into training data & testing data. This is because we often train our machine learning algorithm with training data & we will evaluate our model. We will evaluate the performance of our model using the test data, this part is called a train test split where we will split our original dataset into training & test data. Once we do that we will feed our training data to our machine learning model. In this case

we are going to use logistic regression model because this particular use case is a binary classification. We are going to classify whether a person has a heart disease or not. This is a binary classification either yes or no kind of questions & in that binary classifications logistic regression model is very useful. It's the best model when it comes to binary classification once we train this logistic regression model with our training data. We will do some evaluation on our model to check its performance. After that we will get a trained logistic regression model & to this model when we feed new data our model can predict whether that person has heart disease or not. So, this is the workflow which we have followed.



Heart Disease Detection:

Importing Dependencies :

These are the libraries or dependencies that we need.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Data collection & Processing:

Here, we will load the data from csv file and analysed the data.

```
# loading the csv data to a Pandas DataFrame
heart_data = pd.read_csv('/content/data.csv')
```

```
# print first 5 rows of the dataset
heart_data.head()
```

	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

```
# print last 5 rows of the dataset
heart_data.tail()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

```
# number of rows and columns in the dataset
heart_data.shape
```

```
(303, 14)
```

```
# getting some info about the data
heart_data.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)
```

```
heart_data.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

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

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000

```
# checking the distribution of Target Variable
heart_data['target'].value_counts()
```

```
1    165
0    138
Name: target, dtype: int64
```

1 --> Defective Heart

0 --> Healthy Heart

Splitting the Features and Target:

We are splitting the data into feature and target.

```
X = heart_data.drop(columns='target', axis=1)
Y = heart_data['target']
```

```
print(X)
```

```
     age  sex  cp  trestbps  chol  ...  exang  oldpeak  slope  ca  thal
0     63    1   3       145   233  ...      0      2.3      0   0     1
1     37    1   2       130   250  ...      0      3.5      0   0     2
2     41    0   1       130   204  ...      0      1.4      2   0     2
3     56    1   1       120   236  ...      0      0.8      2   0     2
4     57    0   0       120   354  ...      1      0.6      2   0     2
..   ...  ...  ..       ...   ...  ...    ...      ...    ...  ..   ...
298   57    0   0       140   241  ...      1      0.2      1   0     3
299   45    1   3       110   264  ...      0      1.2      1   0     3
300   68    1   0       144   193  ...      0      3.4      1   2     3
301   57    1   0       130   131  ...      1      1.2      1   1     3
302   57    0   1       130   236  ...      0      0.0      1   1     2

[303 rows x 13 columns]
```

```
print(Y)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

Splitting the data into training data and Test data:

We are splitting the data into training data and testing data.

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(303, 13) (242, 13) (61, 13)
```

Model Training (Logistic Regression):

This step is training the SVM model with training data.

```
model = LogisticRegression()

# training the LogisticRegression model with Training data
model.fit(X_train, Y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra warning msg= LOGISTIC SOLVER CONVERGENCE MSG)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Model Evaluation:

This is about model evaluation where accuracy score of model will be found.

```
# accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

print('Accuracy on Training data : ', training_data_accuracy)
```

Prediction:

Here, the prediction is printed in the form of a list. The name of the list is prediction.

```
input_data = (62,0,0,140,268,0,0,160,0,3.6,0,2,2)

# 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 = model.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')
```

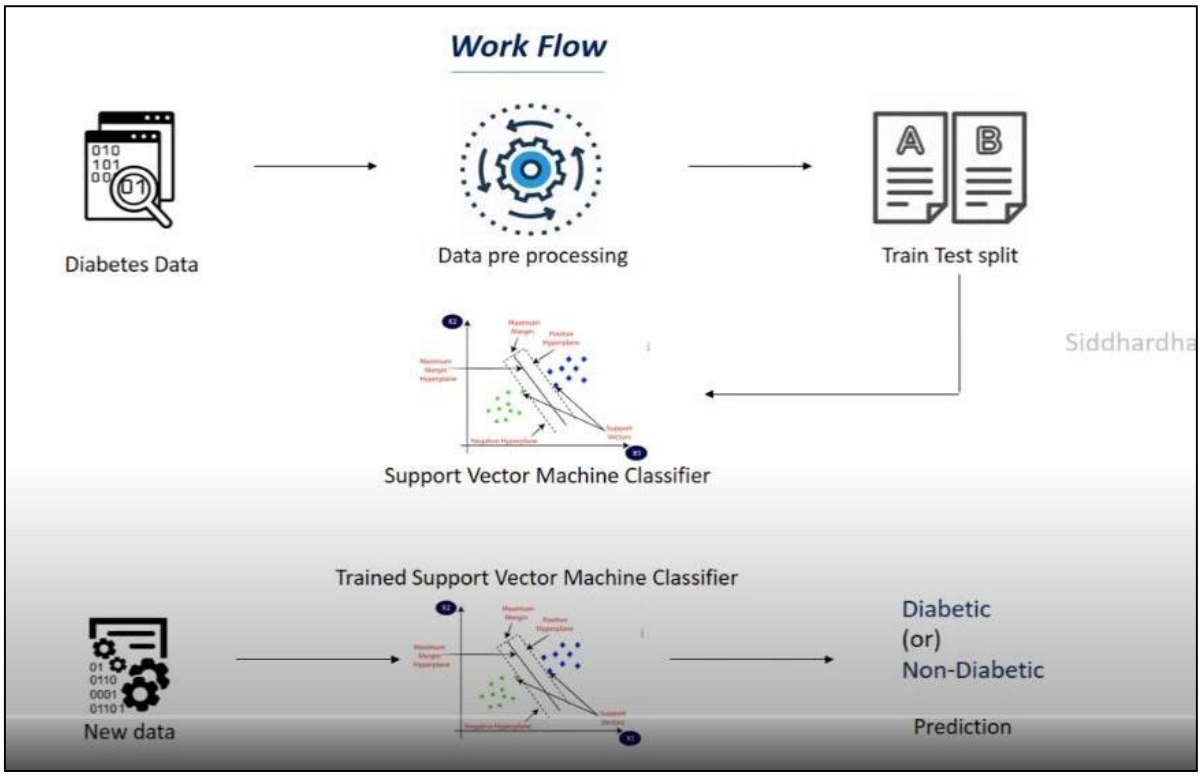
```
[0]
The Person does not have a Heart Disease
```

The work flow of Diabetes disease is as follows:

Diabetes is a chronic disease that occurs either when the pancreas does not produce enough insulin or when the body cannot effectively use the insulin it produces. Insulin is a hormone that regulates blood sugar. Hyperglycemia, or raised blood sugar, is a common effect of uncontrolled diabetes and over time leads to serious damage to many of the body's systems, especially the nerves and blood vessels. IN 2014, 8.5% of adults aged 18 years and older had diabetes. In 2019, diabetes was the direct cause of 1.5 million deaths. To present a more accurate picture of the deaths causes by diabetes, however, deaths due to higher-than-optimal blood glucose through cardiovascular disease, chronic kidney disease and tuberculosis should be added. In 2012 (year of the latest available data), there were another 2.2 million deaths due to high blood glucose.

For this disease we train our model with several medical information such as the blood glucose level and insulin level of patients along with whether the person has diabetic or non-diabetic.

Once we feed the data to our vector machine model what happens is it tries to plot the data in a graph and once it plots the data it tries to find a hyper plane. This hyperplane separates these two data, so once we feed new data in the model it tries to put that particular data in either of these two groups : Diabetic or Non-diabetic.



Diabetes Disease Detection:

Importing Dependencies :

These are the libraries or dependencies that we need.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

Data collection & Processing:

Here, we will load the data from csv file and analysed the data.

```
# loading the diabetes dataset to a pandas DataFrame
diabetes_dataset = pd.read_csv('/content/diabetes.csv')
```

```
pd.read_csv?
```

```
# printing the first 5 rows of the dataset
diabetes_dataset.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
# number of rows and Columns in this dataset
diabetes_dataset.shape
```

```
(768, 9)
```

```
# getting the statistical measures of the data
diabetes_dataset.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

```
diabetes_dataset['Outcome'].value_counts()
```

```
0    500
1    268
Name: Outcome, dtype: int64
```

0 --> Non-Diabetic

1 --> Diabetic

```
diabetes_dataset.groupby('Outcome').mean()
```

Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	0.429734	31.190000
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	0.550500	37.067164

```
# separating the data and labels
X = diabetes_dataset.drop(columns = 'Outcome', axis=1)
Y = diabetes dataset['Outcome']
```

```
print(X)
```

```
     Pregnancies  Glucose  BloodPressure  ...   BMI  DiabetesPedigreeFunction  Age
0              6      148             72  ...  33.6                     0.627   50
1              1       85             66  ...  26.6                     0.351   31
2              8      183             64  ...  23.3                     0.672   32
3              1       89             66  ...  28.1                     0.167   21
4              0      137             40  ...  43.1                     2.288   33
..           ...      ...            ...  ...   ...                       ...  ...
763           10      101             76  ...  32.9                     0.171   63
764            2      122             70  ...  36.8                     0.340   27
765            5      121             72  ...  26.2                     0.245   30
766            1      126             60  ...  30.1                     0.349   47
767            1       93             70  ...  30.4                     0.315   23

[768 rows x 8 columns]
```

```
print(Y)
```

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

Data Standardization:

We need to standardize the data in a normal form and to do that we use data stadarization technique.

```
scaler = StandardScaler()
```

```
scaler.fit(X)
```

```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

```
standardized_data = scaler.transform(X)
```

```
print(standardized_data)
```

```
[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
  -0.87137393]]
```

```
X = standardized_data
Y = diabetes_dataset['Outcome']
```

```
print(X)
print(Y)
```

```
[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
  -0.87137393]]
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

Splitting the data into training data and Test data:

We are splitting the data into training data and testing data.

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2, stratify=Y, random_state=2)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(768, 8) (614, 8) (154, 8)
```

Model Training (SVM):

This step is training the SVM model with training data.

```
classifier = svm.SVC(kernel='linear')
```

```
#training the support vector Machine Classifier
classifier.fit(X_train, Y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Model Evaluation:

This is about model evaluation where accuracy score of model will be found.

```
# accuracy score on the training data
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

```
print('Accuracy score of the training data : ', training_data_accuracy)
```

Prediction:

Here, the prediction is printed in the form of a list. The name of the list is prediction.

```
input_data = (5,166,72,19,175,25.8,0.587,51)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

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

# standardize the input data
std_data = scaler.transform(input_data_reshaped)
print(std_data)

prediction = classifier.predict(std_data)
print(prediction)

if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

```
[[ 0.3429808   1.41167241  0.14964075 -0.09637905  0.82661621 -0.78595734
   0.34768723  1.51108316]]
[1]
The person is diabetic
```

CHAPTER – 7: CONCLUSION & FUTURE SCOPE

Conclusion

This project aims to predict the disease on the basis of the inputs given by the user. The project is designed in such a way that the system takes input from the user and provides a report of user's condition i.e. predict disease's prevalence in body. Average prediction accuracy probability of heart disease is 77%, diabetes is 81% and Parkinson is 88%. Disease Predictor was successfully implemented using flask framework.

Future Scope

i) As the current system covers only two diseases the plan is to include disease of higher fatality, like various cancers, so that early prediction and treatment could be done.

ii) This project has not implemented recommendation of medications to the user. So, medication recommendation can be implemented in the project.

iii) History about the disease for a user can be kept as a log and recommendation can be implemented for medications.

CHAPTER – 8: BIBLIOGRAPHY

1. Prediction of heart disease and classifiers' sensitivity analysis - PubMed (nih.gov)
2. http://refhub.elsevier.com/S2352-9148(21)00180-5/sref26
3. http://refhub.elsevier.com/S2352-9148(21)00180-5/sref18
4. Qawqzeh Y. K., Bajahzar A. S., Jemmali M., Otoom M. M., Thaljaoui A. Classification of diabetes using photoplethysmogram (PPG) waveform analysis: logistic regression ing. *BioMed Research International* . 2020;2020:6. doi: 10.1155/2020/3764653.3764653 [PMC free article] [PubMed] [CrossRef] [Google Scholar]
5. Pethunachiyar G. A. Classification of diabetes patients using kernel based support vector machines. Proceeding of the 2020 International Conference on Computer Communication and Informatics (ICCCI); January 2020; Coimbatore, India. IEEE; pp. 1–4. [Google Scholar]
6. Ahuja R., Sharma S. C., Ali M. A diabetic disease prediction model based on classification algorithms. *Annals of Emerging Technologies in Computing* . 2019;3(3):44-52.doi: 10.33166/aetic.2019.03.005. [CrossRef] [Google Scholar]
7. Mohapatra S. K., Swain J. K., Mohanty M. N. Detection of diabetes using multilayer perceptron. Proceeding of the International Conference on Intelligent Computing and Applications; December 2019; Ghaziabad, India. Springer; pp. 109–116. [CrossRef] [Google Scholar]
8. https://www.kaggle.com/code/arkalodh/diabetes-prediction-easy/data
9. https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset
10. https://www.educba.com/svm-algorithm/
11. https://online.stat.psu.edu/stat462/node/207/