General Disease Prediction System

Project report in partial fulfilment of the requirement for the award of the degree of

Master of Computer Applications

Submitted By

Manoranjan Panigrahi	Department of Computer Applications
University Roll No. 304202000700004	UEM, Kolkata
Sayan Sen	Department of Computer Applications
University Roll No. 304202000700037	UEM, Kolkata
Debapriya Mukherjee	Department of Computer Applications
University Roll No. 304202000700055	UEM, Kolkata
Roshni Dey.	Department of Computer Applications
University Roll No. 304202000700008	UEM, Kolkata
Swarnali Ghosh	Department of Computer Applications
University Roll No. 304202000700007	UEM, Kolkata

Under the guidance of
PROF. Kaustuv Bhattacharjee
Department of Computer Applications
UEM, Kolkata

UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA
University Area, Plot No. III – B/5, New Town, Action Area – III, Kolkata- 700 156

CERTIFICATE

This is to certify that the project entitled **General Disease Prediction System** submitted by **Manoranjan Panigrahi (University Roll No. 304202000700004),**
Sayan Sen (University Roll No. 304202000700037),
Debapriya Mukherjee (University Roll No. 304202000700055),
Roshni Dey (University Roll No. 304202000700008) and
Swarnali Ghosh (University Roll No. 304202000700007)
Students of UNIVERSITY OF ENGINEERING & MANAGEMENT,
KOLKATA, in fulfilment of requirement for the degree of Master of Computer Applications is a bona fide work carried out by them under the supervision and guidance of Prof. Kaustuv Bhattacharjee. During 3rd Semester of academic session of 2020-2022. The content of this report has not been submitted to any other university or institute for the award of any other degree.
I am glad to inform that the work is entirely original and its performance is found to be quite satisfactory.

Prof. Kaustuv Bhattacharjee. Assistant Professor Department of Computer Applications UEM, Kolkata	Prof. Kaustuv Bhattacharjee Head of the Department Department of Computer Applications UEM,Kolkata

ACKNOWLEDGEMENT

We would like to take this opportunity to thank everyone whose cooperation and encouragement throughout the ongoing course of this project remains invaluable to us.

We are sincerely grateful to our guide Prof. Kaustuv Bhattacharjee. Of the Department of Computer Applications, UEM, Kolkata, for his wisdom, guidance and inspiration that helped us to go through with this project and take it to where it stands now.

We would also like to express our sincere gratitude to Prof. Kaustuv Bhattacharjee, HOD, Department of Computer Applications, UEM, Kolkata and all other departmental faculties for their ever-present assistant and encouragement.

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.

Manoranjan Panigrahi

Enrollment No. 12020007015004

Sayan Sen.

Enrollment No. 12020007015037

Debapriya Mukherjee.

Enrollment No. 12020007015055

Roshni Dey.

Enrollment No. 12020007015008

Swarnali Ghosh.

Enrollment No. 12020007015007

TABLE OF CONTENTS

ABSTRACT

This project aims to develop a web application for General Disease Prediction System. Our project aims to provide a user-friendly platform to cross validate results at 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 three most caused lifestyle diseases like diabetes, heart disease and Parkinson diseases 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. In today’s daytime is the factor and being healthy is also an essential so this idea will help user 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: LITERATURE SURVEY

Here we will elaborate the aspects like the literature survey of the project GDPS. We have analyzed the existing projects in market and took inspiration to make this project. Thus decided to go ahead with the project covering with the problem statement.

Existing Systems:

Diabetes:

According to the World Health Organization, (Who.int, 2019) diabetes has become a leading cause of death in the world. Most of the diabetes patients are in low- and middle-income countries. Many researches have been done specifically using Machine Learning in diagnosing diabetes mellitus, some approaches are discussed including their aim, materials, and methods used, results and conclusion. Kaur and Kumari (Kaur and Kumari, 2018) have discussed “Predictive modelling and Analytics for Diabetes using Machine Learning” in their research paper. The main aim of that research was to find out what is the most accurate predictive model to predict diabetes mellitus among 5 predictive models which are known as “Linear Kernel” and “Radial Basis Function” (RBF), “Multifactor Dimensionality Reduction” (MDR), “k-Nearest Neighbor” (kNN), “Kernel Support Vector Machine” (SVM) and “Artificial Neural Network” (ANN). They have used R data manipulation tool to investigate the diabetes dataset, which is known as Pima Indian Diabetes Dataset, originally owned by the “National institute of diabetes and digestive and kidney diseases”, India. This dataset contains 768 instances classified into two classes; diabetic and nondiabetic. In addition, there are eight different risk factors. They have trained their model with 70% training data and tested with 30% remaining data. Those five different models developed using supervised learning methods mentioned above have been experimented in R programming studio.

Table 1: Accuracy of different Predictive models. (Kaur and Kumari, 2018)

No.	Predictive Model	Accuracy
1	Linear Kernel SVM	0.89
2	Radial Basis Kernel SVM	0.84
3	k-NN	0.88
4	ANN	0.86
5	MDR	0.83

According to the above table, Linear Kernel SVM model is the most accurate model among those five predictive models. From the above research, it can be said that “SVM-linear” is best models to predict diabetes.

<u>Heart:</u>

Before we did the experiments, we did research on how people explored heart disease prediction so that we can broaden our horizons and learn from them. (Beyene & Kamat, 2018) recommended different algorithms like Naive Bayes, Classification Tree, KNN, Logistic Regression, SVM and ANN. The Logistic Regression gives better accuracy compared to other algorithms. (Beyene & Kamat, 2018) suggested Heart Disease Prediction System using Data Mining Techniques.

The summary of the literature review can be seen in Table. Several approaches have been performed on this popular dataset, but the accuracy obtained by all the approaches is more with time computations.

1	Harvard Medical School	2020	Hungarian-Cleveland datasets were used for predicting heart disease using different machine learning classifiers and PCA was used for dimensionality reduction and feature selection
2	Zhang et al.	2018	AdaBoost classifier with PCA combination was used for the feature extraction and the accuracy of the prediction was increased

3	Singh et al.	2018	Heart rate variability was for the detection of coronary artery disease. Fisher method and generalised discriminant analysis with binary classifiers were used for the detection of important features.
4	Chen et al.	2018	A subspace feature clustering was used asa subset of stratified feature clustering and for doing a feature reduction of the clusters formed
5	Yang and Nataliani	2018	A fuzzy clustering method especially fuzzy c-means was used for various feature weighted methods and features were reduced
6	Zhang et al	2017	Support vector machine is used for the classification purpose of the clinical data which is matched

			with the codes of NewYork heart association; further findings are left for other researchers
7	Guidi et al.	2014	Neural networks, SVM, and fuzzy system approach are used and Random Forest is used as a classifier, for the prediction of heart failure by using a clinical decision support system
8	Keogh and Mueen	2012	How to break the curse of dimensionality using PCA, SVM, and other Classifiers and reducefeatures.

Parkinson:

Several approaches allow Parkinson detection, classification, and severity prediction. In Table, we present these studies along with the adopted approaches in each study. In the following, we will introduce a summarization of the up-to-date researches in this field.

Sr. no.	Author	SVM	CART	K-NN	NB	BT	RF	FNS	PCA	ANN	EM	ANFIS	SVR	GA	WK	ELM	DNN	CNN	LR
1	Senturk	🟧																	
2	Khour	🟧	🟧	🟧	🟧		🟧												

	y ai al.	🟧	🟧	🟧	🟧		🟧												
3	Nilashi et al.								🟧		🟧	🟧	🟧						
4	Dogantekin													🟧	🟧	🟧			
5	Prince and De Vos						🟧										🟧	🟧	🟧
6	Prashanth et	🟧			🟧	🟧	🟧												
	al.	🟧			🟧	🟧	🟧												
7	Abizade	🟧			🟧	🟧	🟧	🟧											
8	Singh et al.	🟧			🟧	🟧	🟧	🟧											
9	Shetty and Rao	🟧			🟧	🟧	🟧												

10	Ozkan	🟧		🟧	🟧	🟧	🟧		🟧										
11	Nagabhushan	🟧		🟧	🟧	🟧	🟧			🟧									
12	Rovini et al.	🟧			🟧	🟧	🟧	🟧											

Senturk developed an ML method for Parkinson detection diagnosis. In this study, SVM was used for the classification task and presented an overall accuracy of 93.84%.

Several supervised approaches were presented by Khoury et al. for Parkinson detection diagnosis, focusing on gait signals, such as K-NN, NB, SVM, RF, and CART. These approaches were combined with other unsupervised approaches to meet the goal of the study. Among the deployed methods, K-NN, RF, and SVM presented the highest accuracy result.

The presented approach achieved an accuracy of 83.33%. Nilashi, Ibrahim, and Ahani presented a hybrid methodology by using EM, PCA, ANFIS, and SVR techniques. The findings of the study indicated that the presented methodology can detect the severity of the disease accurately.

Avci and Dogantekin presented a new methodology for Parkinson detection detection based on GA, wavelet kernel, and ELM. The findings of the study indicated that the hybrid approach presented better prediction accuracy than other state-of-the-art related approaches.

The outcomes of the study presented efficient performance of the FNS compared to other approaches. Singh et al. presented a new approach for Parkinson detection detection using SVM and presented an overall accuracy of 100%.

Shetty and Rao focused on gait signals in Parkinson detection diagnosis and other neurological disorders using SVM

Ozkan concentrated on vocal indicators by using a hybrid methodology based on PCA with K-NN. The result of the study indicated the robustness of the presented approach with an accuracy of 99.1%.

Pahuja and Nagabhushan assessed the vocal signals of Parkinson detection by using ANN, K- NN, and SVM. The outcomes of the evaluation indicated that ANN presented the most accurate performance with an overall accuracy of 95.89%.

Three ML methods, namely SVM, NB, and RF, were deployed by Rovini et al. and presented an encouraging outcome with a specificity value of 0.967.

CHAPTER – 3: PROBLEM STATEMENT & DISCUSSION

3.1 Problem 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 do not 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.

3.2 Discussion

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 analyze the pattern of presence of diseases in the society. In this project, the disease prediction system will be trained using machine learning.

CHAPTER – 4: PROPOSED SOLUTION & RESULT ANALYSIS

4.1 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.

4.1.1 Requirement Analysis

Functional requirements

Predict disease with the given inputs by user.

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

Non-functional requirements

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

4.1.2 Feasibility Analysis

Technical feasibility

The project is technically feasible as it can be built using the existing available technologies. It is a web-based application that uses Python's 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 datasamples 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.

4.1.3 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.

Workflow of project

Now Let's go through the brief workflow of the project:

i. Firstly, we are taking reliable and current datasets from medical organizations via kaggle (an online community of data scientists and machine learning practitioners).

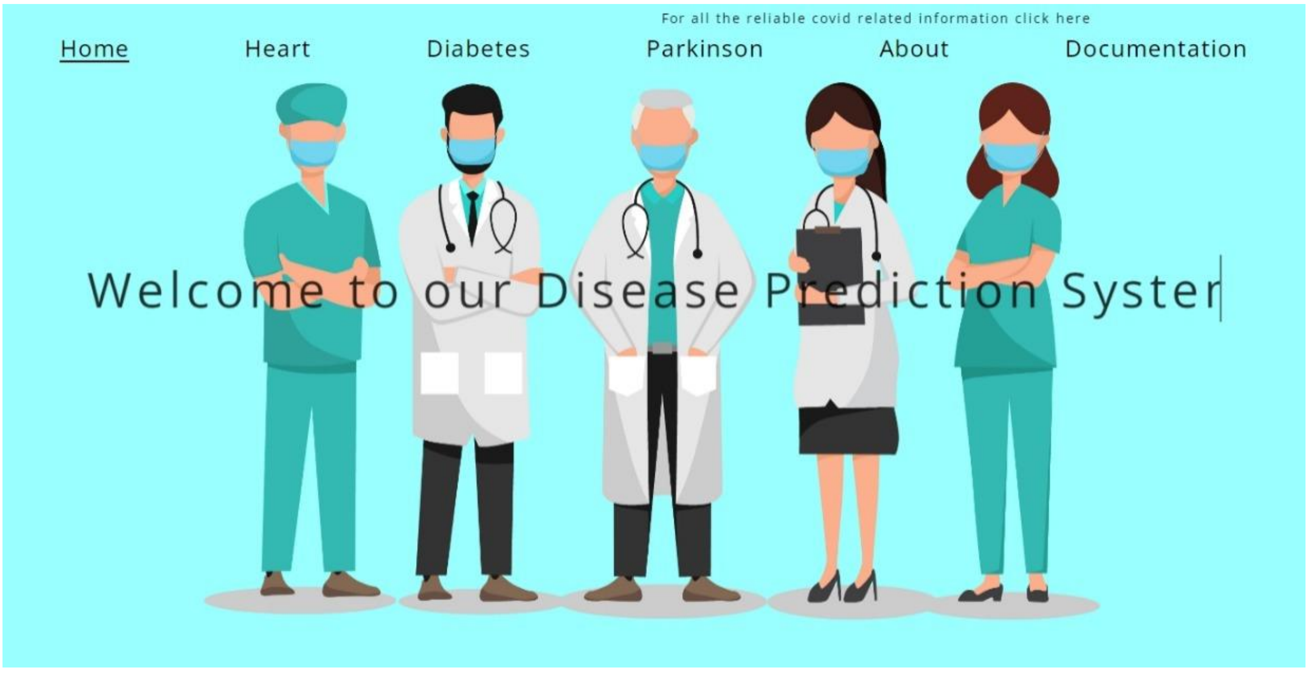
ii. These dataset are used to train the models of individual diseases based on their respective machine learning algorithms.

iii. Once the models are trained and their accuracy are tested, a pickle file would be generated for every individual model.

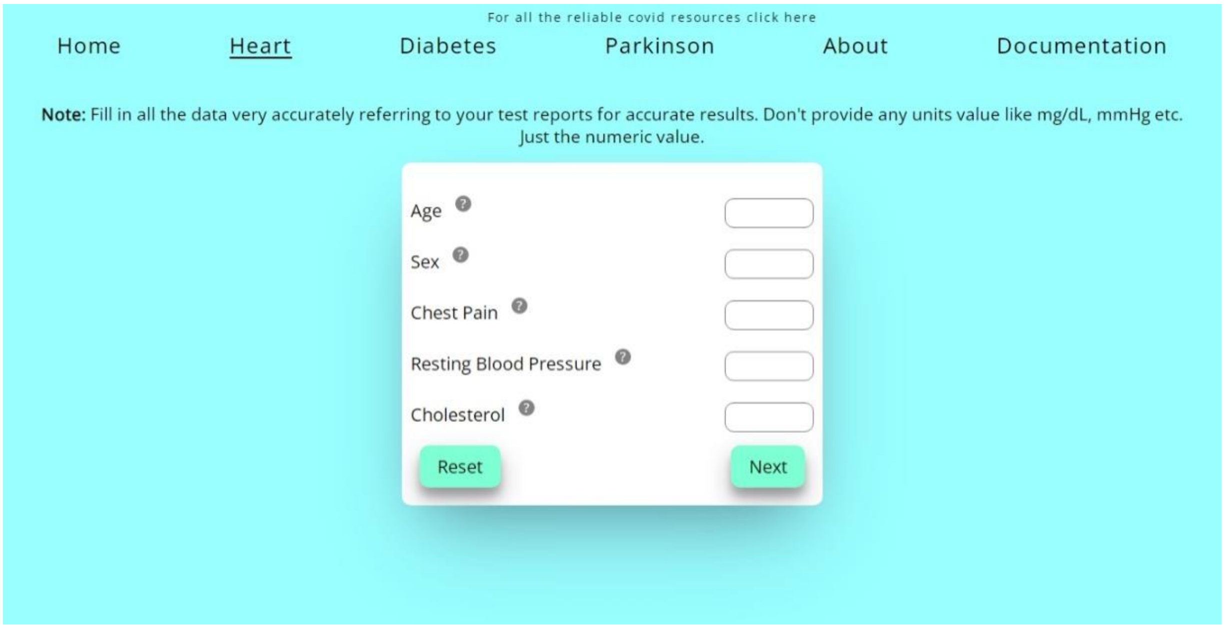
iv. These pickle files later on would be loaded in an centralized python file 'app.py' which is under the flask framework of python architecture. In the app.py file, the predication is occurring from the end users input.

v. Now, the project is being deployed in the web using Heroku (Heroku is a cloud platform as a service supporting programming languages). Heroku gives us a domain as per our choice.

vi. Now let us talk about the layout of the project. We are using HTML which is acting like the backbone of the website and CSS for Formatting, Styling, and creating a more user friendly Interface.


For all the reliable covid related information click here
Home
Heart
Diabetes
Parkinson
About
Documentation
Welcome to our Disease Prediction Syster

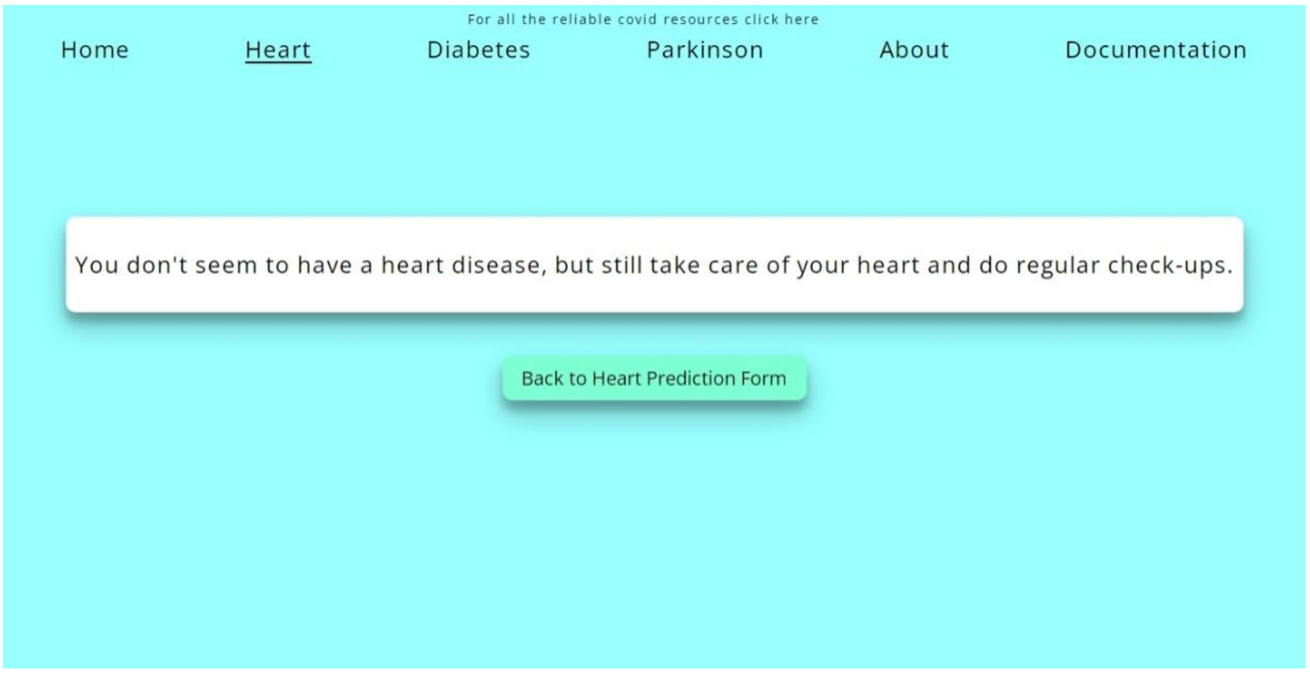
vii. The user are given are multi-step form for input parameters of diseases, which will result in an output. We are using JavaScript for building the form and for processing the input data from users.



Note: Fill in all the data very accurately referring to your test reports for accurate results. Don't provide any units value like mg/dL, mmHg etc. Just the numeric value.

Age	
Sex	
Chest Pain	
Resting Blood Pressure	
Cholesterol	

viii. Lastly, a precaution statement is provided to the user as the end output.


For all the reliable covid resources click here
Home
Heart
Diabetes
Parkinson
About
Documentation
Back to Heart Prediction Form

You don't seem to have a heart disease, but still take care of your heart and do regular check-ups.

4.2 RESULT ANALYSIS

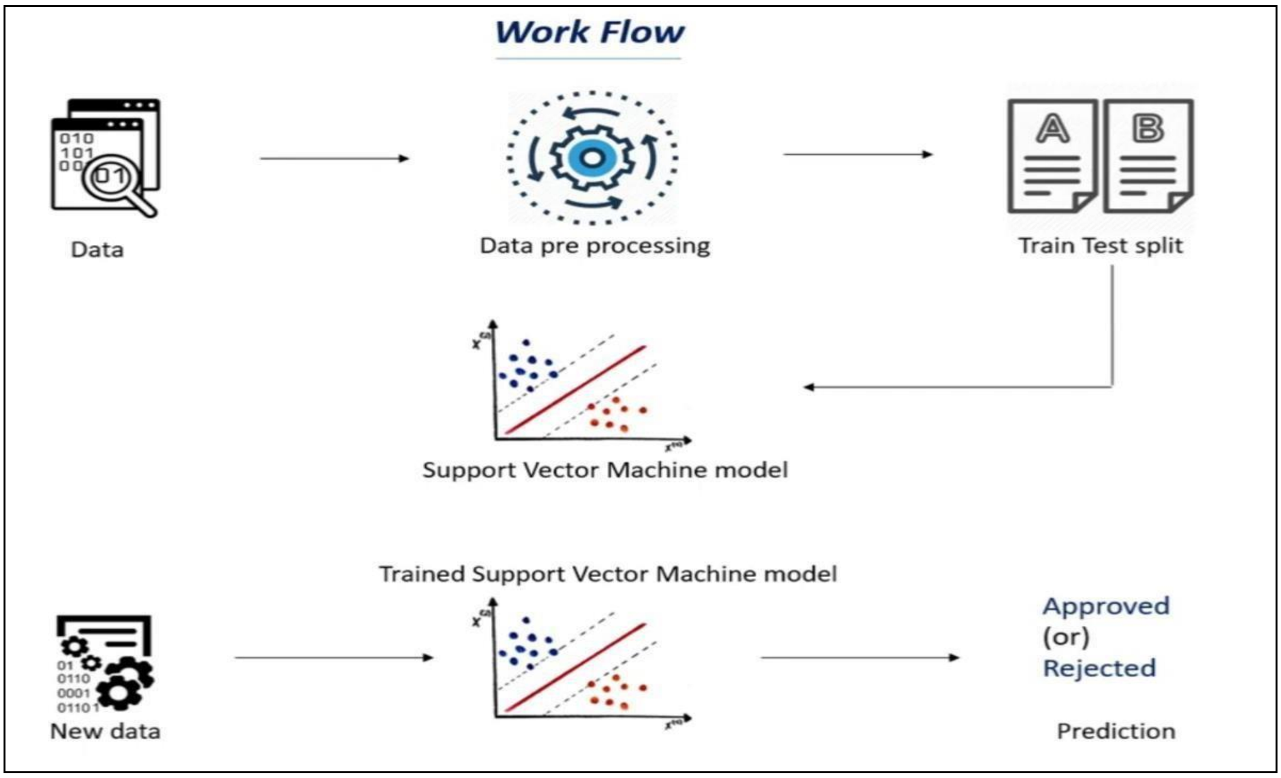
The system will initially be fed data from different sources, the data will then be pre-processed before further process is carried out, this is done to get clean data from the raw initial data, as the raw data would be noisy, or flawed. This data will be processed using algorithms, the system and will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

All right, let us discuss about the model creation and the workflow of project in a more detailed manner.

The workflow of Parkinson's disease is as follows:

Parkinson's disease a progressive nervous system disorder that affects movement leading to shaking, stiffness and difficulty with walking balance. It begins gradually and get worse overtime. It affects mostly male than female and more than 50-60 years of age.

We are building a system using machine-learning algorithm SVM to diagnose this. We collected the Parkinson's data (about the patient who have Parkinson's and who doesn't have Parkinson's) which we need to train our machine learning model. Our machine-learning model can understand the pattern of the data given by us about Parkinson's disease and understand what are the symptoms can be found in Parkinson's patient and what are the symptoms that are common in non-parkinson's patient. After having the dataset we need to process this data, as we cannot feed the raw data to our model. Hence, we need to split our data into train and test model. We use training data in our machine-learning model to train or make understand the model about the data. For this, we are going to use SVM (Support Vector Machine Model). Here training data is used to train the model and testing data is used for evaluating our model. So after evaluation we will have a trained SVM model whereby provided data it can detect whether a person has Parkinson or not. Therefore, we have followed this workflow.



Now, let us discuss the source code, we are dividing the source into different phases to make it more understandable.

1. Data pre-processing and analysis :

This phase involves of preparing a suitable dataset from the raw data gathered also for analysis of the data.

Here, the python open source library pandas is used for importing the dataset in the form of a dataframe.

```
import pandas as pd
```

```
 # Loading the data from dataset to a Pandas DataFrame
parkinsons_df = pd.read_csv('E:/DATASETS/parkinsons.csv')
parkinsons_df.head()
```

Now, we have to observe few of the properties of the dataframe for analysis like,

```
# number of rows and columns in the dataframe
parkinsons_df.shape
```

```
# getting more information about the dataset
parkinsons_df.info()
```

```
# getting some statistical measures about the data
parkinsons_df.describe()
```

Now, we observe the data distribution of the target variable i.e. the output

```
# distribution of target Variable
parkinsons_df['status'].value_counts()
```

```
1    147
0     48
Name: status, dtype: int64
```

1 --> Parkinson's Positive

0 --> Healthy

```
# grouping the data based on the target variable
parkinsons_df.groupby('status').mean()
```

Now, Data is being separated into two variables: features and target

```
X = parkinsons_df.drop(columns=['name','status'], axis=1)
Y = parkinsons_df['status']
```

In addition, the values are printed,

```
     MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  \
0        119.992       157.302        74.997         0.00784
1        122.400       148.650       113.819         0.00968
2        116.682       131.111       111.555         0.01050
3        116.676       137.871       111.366         0.00997
4        116.014       141.781       110.655         0.01284
..           ...           ...           ...             ...
190      174.188       230.978        94.261         0.00459
191      209.516       253.017        89.488         0.00564
192      174.688       240.005        74.287         0.01360
193      198.764       396.961        74.904         0.00740
194      214.289       260.277        77.973         0.00567

     MDVP:Jitter(Abs)  MDVP:RAP  MDVP:PPQ  Jitter:DDP  MDVP:Shimmer  \
0             0.00007   0.00370   0.00554     0.01109       0.04374
1             0.00008   0.00465   0.00696     0.01394       0.06134
2             0.00009   0.00544   0.00781     0.01633       0.05233
3             0.00009   0.00502   0.00698     0.01505       0.05492
4             0.00011   0.00655   0.00908     0.01966       0.06425
..                ...       ...       ...         ...           ...
190           0.00003   0.00263   0.00259     0.00790       0.04087
191           0.00003   0.00331   0.00292     0.00994       0.02751
192           0.00008   0.00624   0.00564     0.01873       0.02308
193           0.00004   0.00370   0.00390     0.01109       0.02296
194           0.00003   0.00295   0.00317     0.00885       0.01884
```

2. Splitting the data into trainee data and Test data:

Now, the data is split into two halves before it is being fitted in the ML model.
Here, the open source library sklearn of python

```
from sklearn.model_selection import train_test_split
```

Now, train_test_split function is used, it is a function in Sklearn model selection for splitting data arrays into two subsets: training data and testing data.

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

Separating data into training and testing sets is an important part of evaluating data mining models.

While training data is necessary to teach an ML algorithm, testing data, as the name suggests, helps you to validate the progress of the algorithm's training and adjust or optimize it for improved results.

3. Data Standardization:

Data Stadardization is performed through StandardScaler module. It removes the mean and scales each variable to unit varience.

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
```

```
scaler.fit(X_train)
```

Here, the ‘X_train’ features training data is being fitted in StandardScaler() for performing transformation of the training data.

```
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

transform() will transform our data such that its distribution will have a mean value 0 and standard deviation of 1

4. Model Training (SVM):

Finally the model is being trained under the Support Vector Machine (SVM) ML algorithm.

Importing the svm module from sklearn library.

```
from sklearn import svm
```

In our case, we are going kernel with linear classification.

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

Now, we have to fit the features training data and target training data

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

Now, our Parkinson's model is successfully created.

5. Model Evaluation:

Once, model is created the accuracy of the model is checked. We use accurancy_score module for this activity.

```
from sklearn.metrics import accuracy_score
```

Now, accuracy of both training and testing data is checked.

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

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

```
Accuracy score of training data :  0.8846153846153846
```

```
# accuracy score on testing data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
```

```
print('Accuracy score of test data : ', test_data_accuracy)
```

```
Accuracy score of test data :  0.8717948717948718
```

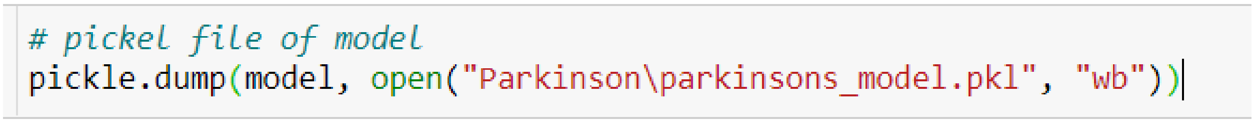
So, we can observe that training data accuracy is 88% accurate and Testing data accuracy is 87% accurate.

6. Pickle file creation:

Pickle file is used for python object serialization into character stream. The pickle

```
import pickle
```

is used to avoid the recompilation of the source code of the models in every iteration.

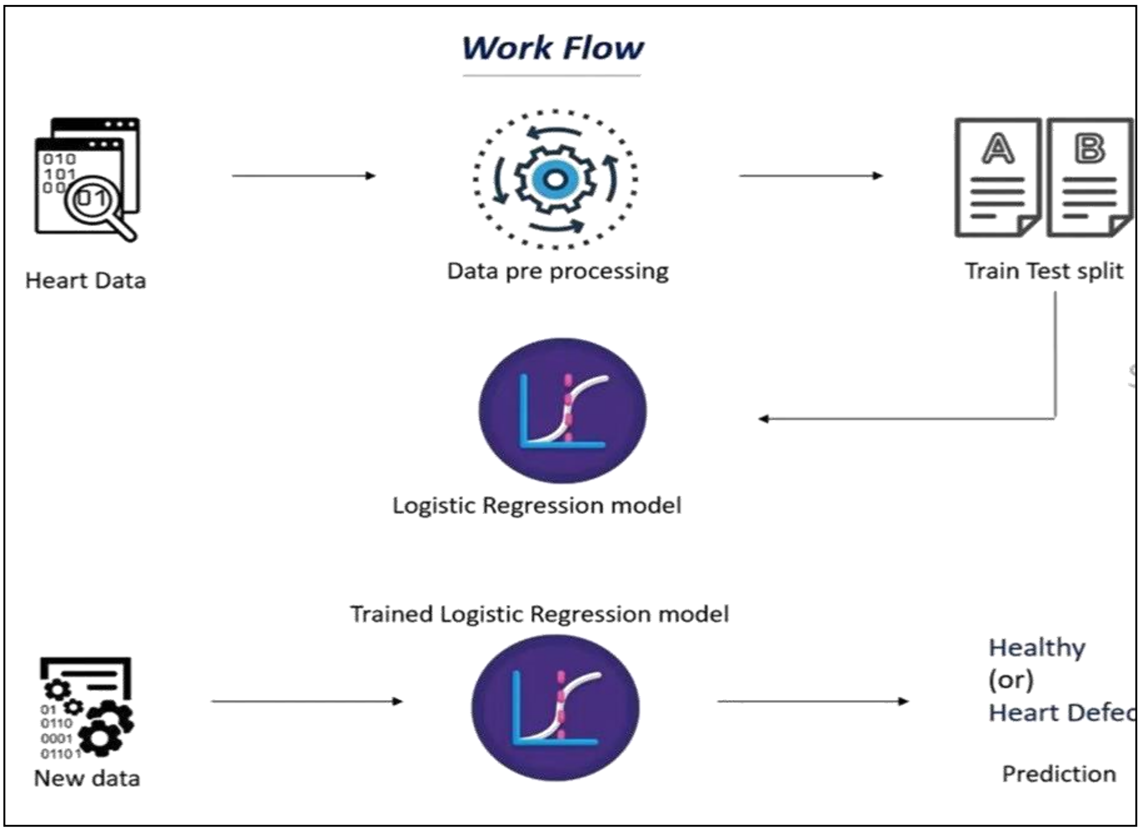


```
# pickel file of model
pickle.dump(model, open("Parkinson\parkinsons_model.pkl", "wb"))
```

The workflow 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 of65.

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.



Now, let us discuss the source code, we are dividing the source into different phases to make it more understandable.

1. **Data pre-processing and analysis :**

This phase involves of preparing a suitable dataset from the raw data gathered also analysis of the data.

Here, the python open source library pandas is used for importing the dataset in the form of a dataframe.

```
import pandas as pd
```

```
# loading the csv data to a Pandas DataFrame
heart_data = pd.read_csv('E:/DATASETS/heart_cleveland.csv')
heart_data.head()
```

Now, we observe the data distribution of the target variable i.e the output.

```
# splitting the feature and target
X = heart_data.drop(columns='target', axis=1)
Y = heart_data['target']
```

Here we are splitting the Features and Target

2. <u>Splitting the data into training data and Test data:</u>

Now, train_test_split function is used, it is a function in Sklearn model selection for splitting data arrays into two subsets: training data and testing data.

```
# splitting the data into training 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)
```

3. <u>Model Training:</u>

Finally, we are creating ML model using Logistic Regression algorithm.

```
from sklearn.linear_model import LogisticRegression
```

Now, we have to fit the features training data and target training data.

```
# Loading the LogisticRegression algorithm into a model variable
model = LogisticRegression(max_iter=6000)

# training the algorithm with X and Y training data and saving the result in a mainmodel var
mainmod = model.fit(X_train, Y_train)
```

4. <u>Model Evaluation:</u>

Once, model is created the accuracy of the model is checked. We use accurancy_score module for both training and testing data.

```
#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)
```

```
Accuracy on Training data :  0.869198312236287
```

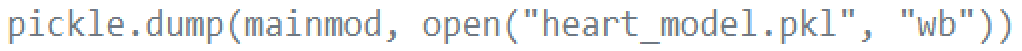
```
# accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
print('Accuracy on Test data : ', test_data_accuracy)
```

```
Accuracy on Test data :  0.9
```

Therefore, we can conclude that training data accuracy is 86% and testing data is 90%.

5. **Pickle file creation:**



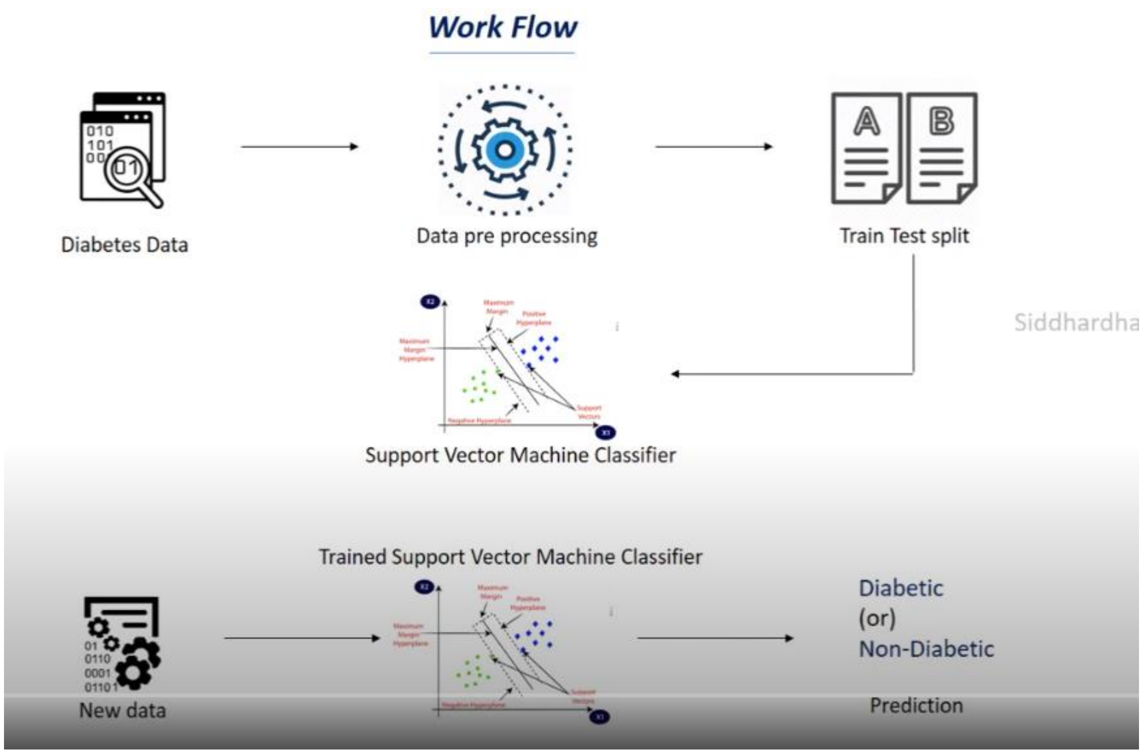
```
pickle.dump(mainmod, open("heart_model.pkl", "wb"))
```

pickle file is used for python object serialization into character stream. The pickle is used to avoid the recompilation of the source code of the models in every iteration.

The work flow of Diabetes disease is as follows:

Diabetes is a chronic disease that occurs 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 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 hyperactive plane. This hyperplane separates these two data, so once we feed new data in the model it tries to put that particular data in of these two groups: either Diabetic or Non-diabetic.



Now, let us discuss the source code, we are dividing the source into different phases to make it more understandable.

1. <u>**Data pre-processing and analysis :**</u>

This phase involves of preparing a suitable dataset from the raw data gathered also for analysis of the data.

Here, the python open source library pandas is used for importing the dataset in the form of a dataframe.

```
import pandas as pd
```

```
#loading the diabetes dataset to a pandas DataFrame
diabetes_df = pd.read_csv('Diabetes\Diabetes.csv')
```

Now, we observe the data distribution of the target variable i.e. the output

```
# distribution of target Variable
diabetes_df['Outcome'].value_counts()
```

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

0 ---> Non-Diabetic

1 ---> Diabetic

Now, Data is being separated into two variables: features and target

```
# Data Pre-Processing
X = diabetes_df.drop(columns='Outcome', axis=1)
Y = diabetes_df['Outcome']
```

2. <u>Splitting the data into trainee data and Test data:</u>

Now, the data is split into two halves before it is being fitted in the ML model.
Here, the open source library sklearn of python

```
from sklearn.model_selection import train_test_split
```

Now, train_test_split function is used, It is a function in Sklearn model selection for splitting data arrays into two subsets: training data and testing data.

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

3. <u>Data Standardization:</u>

Data Stadardization is performed through StandardScaler module. It basically removes the mean and scales each variable to unit varience.

```
from sklearn.preprocessing import StandardScaler
```

```
scaler.fit(X_train)
```

Here, the ‘X_train’ features training data is being fitted in StandardScaler() for performing transformation of the training data.

```
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

transform() will transform our data such that its distribution will have a mean value 0 and standard deviation of 1

4. <u>Model Training (SVM):</u>

Finally the model is being trained under the Support Vector Machine (SVM) ML algorithm.

Importing the svm module from sklearn library.

```
from sklearn import svm
```

In our case we are going kernel with linear classification.

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

Now, we have to fit the features training data and target training data

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

```
SVC(kernel='linear')
```

Now, our Diabetes's model is successfully created.

5. <u>Model Evaluation:</u>

Once, model is created the accuracy of the model is checked. We use accurancy_score module for this activity.

```
from sklearn.metrics import accuracy_score
```

Now, accuracy of both training and testing data is checked.

```
# 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)
```

```
Accuracy score of the training data :  0.7734375
```

```
# accuracy score on the test data
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
print('Accuracy score of the test data : ', test_data_accuracy)
```

```
Accuracy score of the test data :  0.7708333333333334
```

Therefore, we can observe that training data accuracy is 77% and testing data accuracy is 77% accurate.

6. <u>Pickle file creation:</u>

Pickle is used for python object serialization into character stream. It is used to avoid the recompilation of the source code of the models in every iteration.

```
import pickle
```

```
 # pickel file of model
pickle.dump(classifier, open("Diabetes\diabetes_model.pkl", "wb"))
```

Now, these three individual files are loaded in python file in flask framework.

Creation of app.py Flask Framework:

Importing the dependencies required

```
import pickle
from flask import Flask, render_template, request
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd
```

```
# instantiating the class Flask with app obj,
app = Flask(__name__)
```

Here, the pickle files are loaded,

```
file = open("Heart\heart_model.pkl", "rb")
model = pickle.load(file)

file_diabetes = open("Diabetes\diabetes_model.pkl", "rb")
model_diabetes = pickle.load(file_diabetes)
diabetes_df = pd.read_csv('Diabetes\Diabetes.csv')
diabetes_X = diabetes_df.drop(columns='Outcome', axis=1)
diabetes_Y = diabetes_df['Outcome']

file_parkinson = open("Parkinson\parkinsons_model.pkl", "rb")
model_parkinson = pickle.load(file_parkinson)
parkinsons_df = pd.read_csv('Parkinson\parkinsons.csv')
parkinsons_X = parkinsons_df.drop(columns=['name', 'status'], axis=1)
parkinsons_Y = parkinsons_df['status']
```

Now, for display each html page we need use @app decorator which is used to map URL.

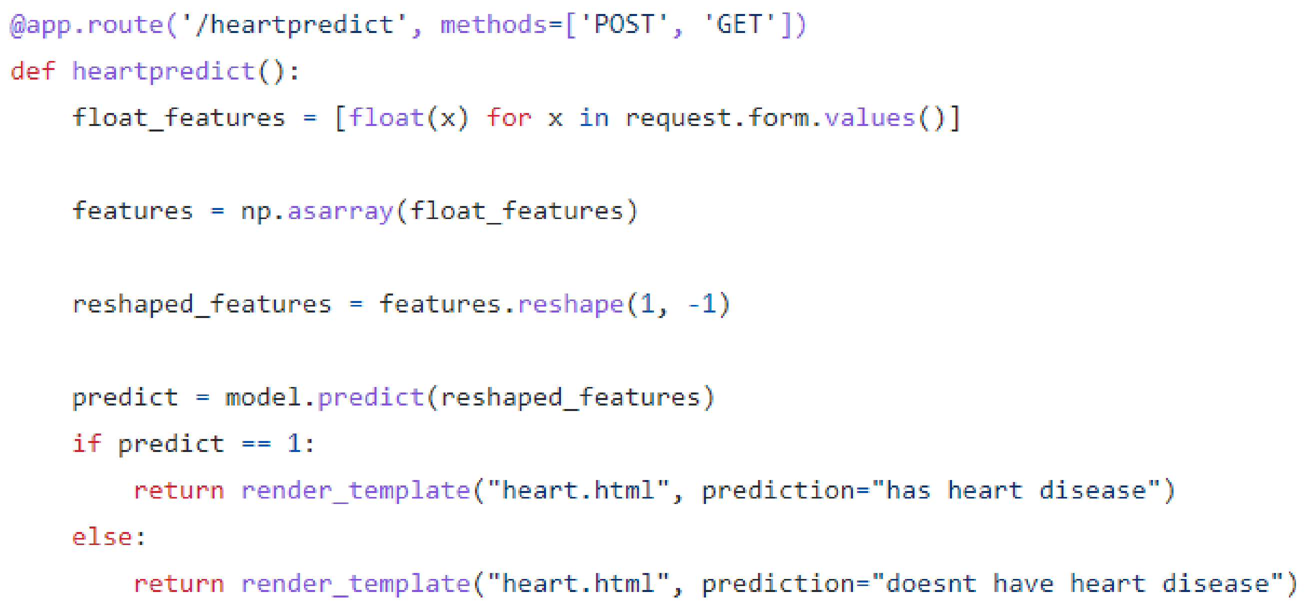
```
@app.route('/')
def home():
    return render_template("index.html")

@app.route('/index.html')
def index():
    return render_template("index.html")

@app.route('/heart.html')
def heart():
    return render_template("heart.html")

@app.route('/diabetes.html')
def diabetes():
    return render_template("diabetes.html")

@app.route('/parkinson.html')
def parkinson():
    return render_template("parkinson.html")

@app.route('/about.html')
def about():
    return render_template("about.html")

@app.route('/contact.html')
def contact():
    return render_template("contact.html")
```

Now, we are building the predictive logic system where input data are given by end user from website.

We will be using numpy, an open source library that will be used to calculate the output from the user input.

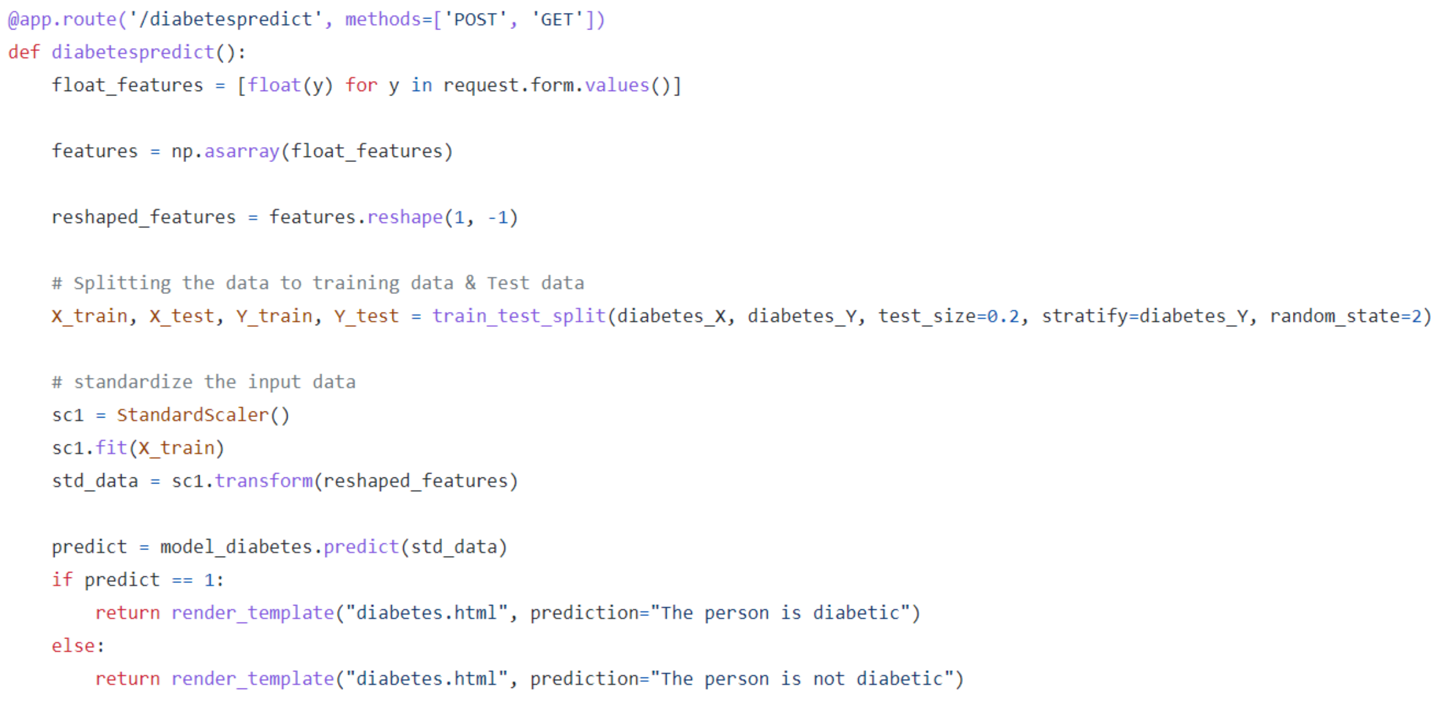
```
import numpy as np
```

Heart Disease:



```
@app.route('/heartpredict', methods=['POST', 'GET'])
def heartpredict():
    float_features = [float(x) for x in request.form.values()]

    features = np.asarray(float_features)

    reshaped_features = features.reshape(1, -1)

    predict = model.predict(reshaped_features)
    if predict == 1:
        return render_template("heart.html", prediction="has heart disease")
    else:
        return render_template("heart.html", prediction="doesnt have heart disease")
```

Diabetes:



```
@app.route('/diabetespredict', methods=['POST', 'GET'])
def diabetespredict():
    float_features = [float(y) for y in request.form.values()]

    features = np.asarray(float_features)

    reshaped_features = features.reshape(1, -1)

    # Splitting the data to training data & Test data
    X_train, X_test, Y_train, Y_test = train_test_split(diabetes_X, diabetes_Y, test_size=0.2, stratify=diabetes_Y, random_state=2)

    # standardize the input data
    sc1 = StandardScaler()
    sc1.fit(X_train)
    std_data = sc1.transform(reshaped_features)

    predict = model_diabetes.predict(std_data)
    if predict == 1:
        return render_template("diabetes.html", prediction="The person is diabetic")
    else:
        return render_template("diabetes.html", prediction="The person is not diabetic")
```

Parkinson’s disease:

```
@app.route('/parkinsonpredict', methods=['POST', 'GET'])
def parkinsonpredict():
    a = 0
    float_features = [float(a) for a in request.form.values()]

    features = np.asarray(float_features)

    reshaped_features = features.reshape(1, -1)

    # Splitting the data to training data & Test data
    P_X_train, X_test, Y_train, Y_test = train_test_split(parkinsons_X, parkinsons_Y, test_size=0.2, random_state=2)

    # standardize the input data
    sc2 = StandardScaler()
    sc2.fit(P_X_train)
    P_std_data = sc2.transform(reshaped_features)

    predict = model_parkinson.predict(P_std_data)
    if predict == 1:
        return render_template("parkinson.html", prediction="The Person has Parkinsons DiseaseT")
    else:
        return render_template("parkinson.html", prediction="The Person does not have Parkinsons Disease")
```

Now, we have to run the current app

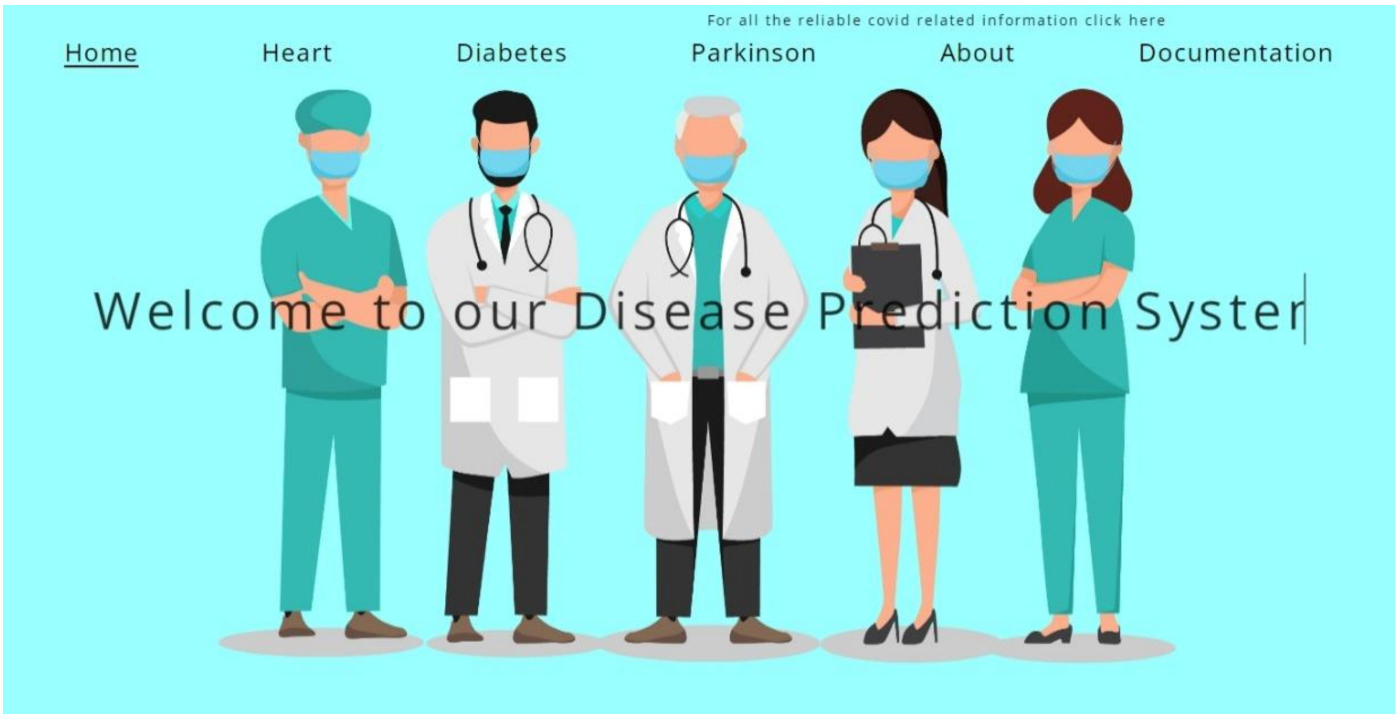
```
if __name__ == "__main__":
    app.run(debug=True)
```

The project backend part is completed now. The next step involves deploying the project in cloud-based platform i.e. Heroku and for deployment and Version-Control, we are using Git.

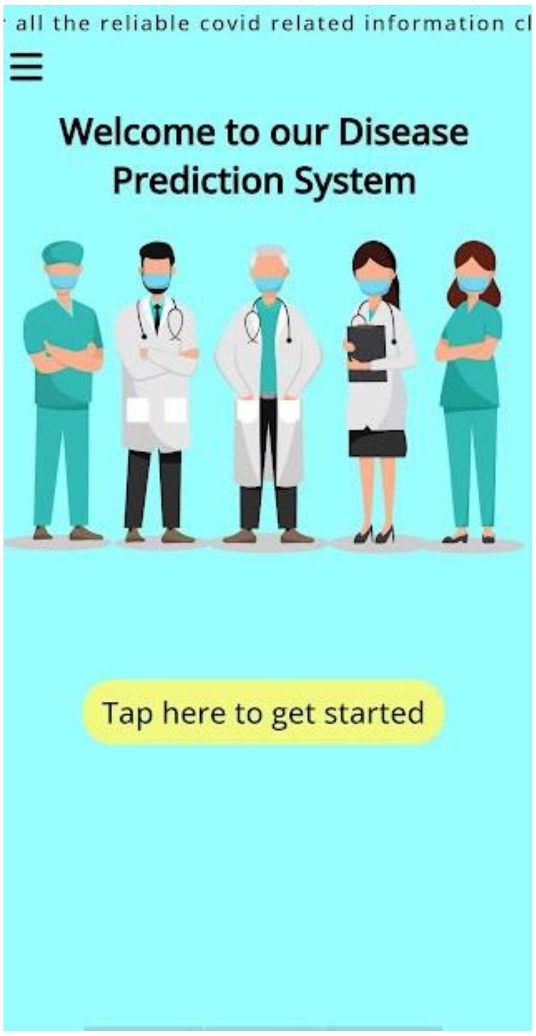
Our Domain: https://disease-predictions-app.herokuapp.com/

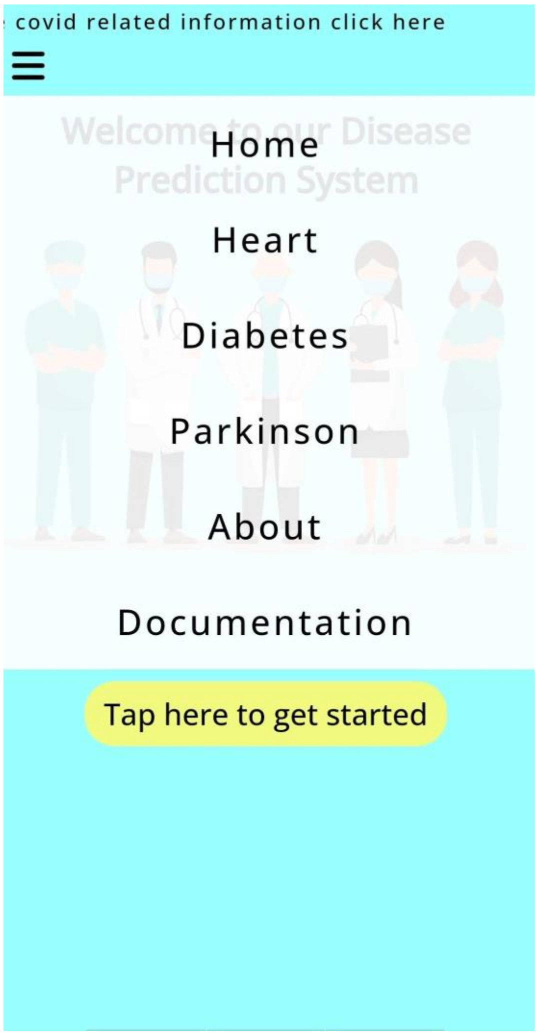
Website Layout:

Desktop/pc:


For all the reliable covid related information click here
Home
Heart
Diabetes
Parkinson
About
Documentation
Welcome to our Disease Prediction Syster

Mobile


all the reliable covid related information cl
Welcome to our Disease Prediction System
Tap here to get started



CHAPTER – 5: CONCLUSION & FUTURE SCOPE

Conclusion

This project aims to predict the disease based on 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 three 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) User Medical report related to these diseases can be stored and further recommendations can be made.

CHAPTER – 6: BIBLIOGRAPHY

1. https://www.neurologyindia.com/article.asp?issn=0028-3886;year=2018;volume=66;issue=7;spage=26;epage=35;aulast=Radhakrishnan
2. https://www.thelancet.com/journals/langlo/article/PIIS2214-109X(21)00214-X/fulltext
3. https://www.frontiersin.org/articles/10.3389/fneur.2020.00524/full
4. https://www.researchgate.net/publication/288324436_Global_Prevalence_and_Therapeutic_Strategies_for_Parkinson's_Disease
5. https://www.irjet.net/archives/V5/i3/IRJET-V5I3930.pdf
6. https://www.kaggle.com/nidaguler/parkinsons-data-set
7. https://colab.research.google.com/drive/1PTbvUuqlOoCIr4uH0wsuVAsr88dtjb1c#scrollTo=Ojx1z6YnIKlT&uniqifier=2
8. https://www.mayoclinic.org/diseases-conditions/diabetes/diagnosis-treatment/drc-20371451#:~:text=A%20fasting%20blood%20sugar%20level,separate%20tests%2C%20you%20have%20diabetes
9. https://www.who.int/news-room/fact-sheets/detail/diabetes
10. https://www.healthline.com/health/diabetes
11. https://www.dropbox.com/s/uh7o7uyeghqkhoy/diabetes.csv?dl=0
12. https://colab.research.google.com/drive/1oxnhMTlomJ4HVhPuowpPFyMt1mwuOuQo?usp=sharing#scrollTo=-71UtHzNVWjB
13. https://colab.research.google.com/drive/1FYGPRSEGvd0urNlZmRJHx-gq6ANn3IpX?usp=sharing
14. https://statisticalhorizons.com/whats-so-special-about-logit
15. https://searchbusinessanalytics.techtarget.com/definition/logistic-regression
16. https://www.learningtheory.org/learning-has-just-started-an-interview-with-prof-vladimir-vapnik/