Summer Internship Project report on

**Multi-Disease Prediction Model: Integrating Machine Learning with Flask API**

## BACHELOR OF TECHNOLOGY

in

## CSE DATA SCIENCE

**G.Poojitha Reddy 21E51A6713**

**K.Vaishnavi 21E51A6723**

**N.Saketh Reddy 21E51A6731**

**P.Madhu 21E51A6737**

Under the guidance of

## Mrs. KRISHNA JYOTHI

Associate professor Department of ET, HITAM

## HYDERABAD INSTITUTE OF TECHNOLOGY AND MANAGEMENT

Autonomous, Approved by AICTE, Accredited by NAAC, NBA.

Gowdavelly (Village), Medchal (Mandal), Medchal - Malkajgiri (Dist.), Hyderabad, TS- 501401.

2022–2023

## HYDERABAD INSTITUTE OF TECHNOLOGY AND MANAGEMENT

**(**Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC, NBA – TS 501401**) DEPARTMENT OF EMERGING TECHNOLOGY**



**CERTIFICATE**

This is to certify that the internship project work entitled **“Multi-Disease Prediction Model: Integrating Machine Learning with Flask API”** is

being submitted by **G.Poojitha Reddy, K.Vaishnavi,N.Saketh Reddy,P.Madhu** bearing Roll No.**21E151A6713,21E51A6723,21E51A6731,21E51A6737** in partial fulfilment of the academic requirement, at Hyderabad Institute of Technology and Management, Hyderabad is a record of bonafied work carried out by them under our guidance. The matter contained in this document has not been submitted to any other University or institute.

## Internal Guide Head of Department

#### Mrs KRISHNA JYOTHI Mrs ILA CHANDANA KUMARI

Associate professor Professor & Head

Department of ET, HITAM Department of ET, HITAM

## DECLARATION

We here by declare that the internship project entitled **“Multi-Disease Prediction Model: Integrating Machine Learning with Flask API’’** submitted to **Hyderabad Institute of Technology and Management affiliated to Jawaharlal Nehru Technological University**. Hyderabad (JNTUH) as part of academic requirement, is a result of original research work and done by us. It is further declared that the internship project report or any part thereof has not been previously submitted to any other university or institute.

G.Poojitha Reddy 21E51A6713

K.Vaishnavi 21E51A6723

N.Saketh Reddy 21E51A6731

P.Madhu 21E51A6737

## ABSTRACT

Many of the machine learning models for health care analysis now in use focus on just one disease at a time. Like one analysis is for diabetes, one for heart, and one for brain tumor. A single analysis cannot forecast more than one disease using a same framework. In this paper, a system that uses the Flask Application programming interface (API) to forecast numerous diseases is proposed. This paper proposed a method to examine diabetes, brain tumor, heart disease, and Alzheimer. Later, more disorders might be included, such as covid detection, fever tests, and many others. TensorFlow, Flask API, and machine learning techniques were utilized to implement numerous disease analyses. The model behavior is saved using Python pickling, and the pickle file is loaded when needed using Python unpickling. The significance of this article analysis is that all the factors that contribute to the sickness are considered while analyzing it, making it possible to identify the disease's full range of potential impacts. For instance, many of the existing systems for analyzing diabetes considered a few parameters such as age, sex, BMI, and insulin. But in our proposed system we have considered additional factors like number of pregnancies, Glucose concentration, skin thickness, heart rate/pulse rate, and diabetes pedigree which makes our model more efficient compared to existing model. The final model's behavior will be recorded in a pickle file in Python and the Flask API will be created. The parameters of the disease must be sent together with the disease name when using this API. The relevant model will be called by the Flask API, which then returns the patient's state. The significance of this study is to examine the most diseases possible to keep track of patients' conditions and forewarn them when necessary to reduce the death rate.

|  |  |  |
| --- | --- | --- |
|  | **INDEX** |  |
| **CONTENT**  ABSTRACT |  | **PAGE NO.**  IV |
| LIST OF FIGURES |  | VI |
| **CHAPTER 1: INTRODUCTION** |  | **(07-08)** |
| 1.2: Objective of the project |  | 09 |

**CHAPTER 2: HARDWARE & SOFTWARE REQUIREMENTS (10)**

* 1. [: Hardware requirements 11](file:///C:\Users\gagan\OneDrive\Desktop\sip%20report.docx#_TOC_250005)
  2. [: Software requirements 11](file:///C:\Users\gagan\OneDrive\Desktop\sip%20report.docx#_TOC_250004)

CHAPTER 3: DESIGN & METHODOLOGY (12-28)

* 1. [: Block diagram 14](file:///C:\Users\gagan\OneDrive\Desktop\sip%20report.docx#_TOC_250003)
  2. [: Working 14-17](file:///C:\Users\gagan\OneDrive\Desktop\sip%20report.docx#_TOC_250002)
  3. [: Code 18-26](file:///C:\Users\gagan\OneDrive\Desktop\sip%20report.docx#_TOC_250001)
  4. [: Discussion 27](file:///C:\Users\gagan\OneDrive\Desktop\sip%20report.docx#_TOC_250000)
  5. : Conclussion 28

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **NAME OF THE FIGURE** | **PAGE NO.** |
| Figure 3.1 | Block diagram | 14 |
| Figure 3.2 | Working Diagram of Parkinson’s Disease Prediction |  |
| Figure 3.3 | Working Diagram of Diabetes Prediction |  |
| Figure 3.4 | Working Diagram of Heart Disease Prediction |  |

# CHAPTER 1 INTRODUCTION

## CHAPTER 1

### INTRODUCTION

Healthcare is increasingly adopting the artificial intelligence (AI) technologies that are pervasive in modern business and daily life. Artificial intelligence in healthcare has the potential to aid providers in many areas of patient care and operational procedures enabling them to build on current solutions and solve problems more quickly. The majority of AI and healthcare technologies are highly relevant to the healthcare industry, yet hospitals and other healthcare organizations may employ quite different strategies. Precision medicine is the most widely used use of conventional machine learning in the field of artificial intelligence in healthcare. It is a big step forward for many healthcare organizations to be able to forecast which treatment approaches would be most effective with patients based on their characteristics and the treatment framework. And even though some articles on the use of artificial intelligence in healthcare suggest that it can perform just as well as or better than humans at certain procedures, like diagnosing disease, it will be a considerable amount of time before AI in healthcare replaces humans for a wide range of medical tasks. Machine learning and precision medicine applications, which make up the majority of AI in healthcare, require data for training with known outcomes. We call this supervised learning. Many analyses of the health care industry's current systems only took one ailment at a time. For instance, articles are used to analyze diabetes, to analyze brain tumor, and to forecast heart disease. Most articles concentrate on a specific disease. Any institution that wishes to analyze the health reports of its patients must use a variety of models. The method used in the current system is helpful for studying only a certain ailment. Today's mortality has grown as a result of the failure to accurately diagnose diseases. Even after being treated for one disease, a patient may still be afflicted with another. When analyzing the disease, several current methods employed a limited number of parameters. Because of this, it might not be feasible to pinpoint the diseases that will be brought on by the disease's effects. For instance, diabetes increases the risk of developing dementia, hearing loss, retinopathy, neuropathy, and heart disease. The analysis of diabetes, brain tumor, heart disease, and Alzheimer data sets were taken into consideration in this paper. In the future, several other diseases may be added, such as skin conditions, diseases linked to fever, and many more. This adaptable analysis later encompassed several disorders for examination. The model file associated with the analysis of the new disease must be added by the developer when adding any new disease analysis to this API. The developer must set up Python picking to store model behavior when creating new diseases. The developer can load a pickled file to obtain the model behavior while utilizing this Flask API. When a user wishes to analyze a patient's health, they may either forecast a specific disease or, if the report contains factors that are used to predict other diseases, this analysis will provide the most appropriate disease identifications. By forewarning patients depending on their health circumstances, this article aims to stop the mortality ratio from rising day by day. The cost of patient analysis can be decreased since there are several diseases models and forecasts made at one location. The section I explains the Introduction of multiple disease prediction using machine learning. Section II presents the literature review of existing systems and Section III present proposed system architecture and implementation details Section IV presents results and discussion of proposed system. Section V concludes our proposed system. While at the end list of references paper are presented

**1.2 OBJECTIVE OF THE PROJECT**

➢ The analysis accuracy is reduced when the quality of medical data in incomplete.

➢ Moreover, different regions exhibit unique characteristics of certain regional diseases, which may weaken the prediction of disease outbreaks.

➢ However, those existing work mostly considered structured data.

➢ There is no proper methods to handle semi structured and unstructured.

➢ The proposed system will consider both structured and unstructured data.

➢ The analysis accuracy is increased by using Machine Learning algorithm.

* 1. **Scope of the Project**

➢ In this work, our goal is to provide a tool to assist professionals and consumers in finding and choosing disease.

➢ To achieve this goal, we develop an approach that allows a user to query for disease that satisfy a set of conditions based on disease properties, such as disease indications and also takes into account patient profiles.

# CHAPTER 2

**HARDWARE AND SOFTWARE REQUIREMENTS**

## CHAPTER 2

**HARDWARE AND SOFTWARE REQUIREMENTS**

### HARDWARE REQUIREMENTS:

1. **Processor:** A modern multicore processor (e.g., Intel Core i5 or AMD Ryzen 5) for smoother performance.
2. **RAM:** At least 8GB of RAM, but 16GB or more is recommended for seamless multitasking.
3. **Hard Disk:** A solid-state drive (SSD) with at least 256GB of storage for faster load times and efficient project management.
4. **Monitor:** A larger and high-resolution display (e.g., 24" Full HD or higher) for better design and development work.
5. **Keyboard and Mouse:** Standard keyboard and mouse should suffice, but consider investing in ergonomic peripherals for comfort during long work sessions.

### SOFTWARE REQUIREMENTS:

**Python**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

• Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

• Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs

. • Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

• Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features Python's features include –

• Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

• Easy-to-read − Python code is more clearly defined and visible to the eyes.

• Easy-to-maintain − Python's source code is fairly easy-to-maintain.

• A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

• Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

• Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

• Extendable − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

• Databases − Python provides interfaces to all major commercial databases.

• GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

• Scalable − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

• It supports functional and structured programming methods as well as OOP.

• It can be used as a scripting language or can be compiled to byte-code for building large applications.

• It provides very high-level dynamic data types and supports dynamic type checking.

• It supports automatic garbage collection.

• It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java. Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

# CHAPTER 3

**DESIGN AND METHODOLOGY**

## CHAPTER 3

**DESIGN AND METHODOLOGY**

### INTRODUCTION: DESIGN:

EXISTING SYSTEM

Machine can predict diseases but cannot predict the sub types of the diseases caused by occurrence of one disease. It fails to predict all possible conditions of the people. Existing system handles only structured data. The prediction system are broad and ambiguous. In current past, countless disease estimate classifications have been advanced and in procedure. The standing organizations arrange a blend of machine learning algorithms which are judiciously exact in envisaging diseases. However the restraint with the prevailing systems are speckled. First, the prevailing systems are dearer only rich people could pay for to such calculation systems. And also, when it comes to folks, it becomes even higher. Second, the guess systems are non-specific and indefinite so far. So that, a machine can envisage a positive disease but cannot expect the sub types of the diseases and diseases caused by the existence of one bug. For occurrence, if a group of people are foreseen with Diabetes, doubtless some of them might have complex risk for Heart viruses due to the actuality of Diabetes. The remaining schemes fail to foretell all possible surroundings of the tolerant. 3.1

DISADVANTGES OF EXISTING SYSTEM

• Does not analyze the disease

• Less security

• There is no feedback system

PROPOSED SYSTEM

The Proposed system of multiple disease prediction using machine learning is that we have used algorithms and all other various tools to build a system which predicts the disease of the patient using the symptoms and by taking those symptoms we are comparing with the system‟s dataset that is previously available. By taking those datasets and comparing with the patient‟s disease we will predict the accurate percentage disease of the patient. The dataset and symptoms go to the prediction model of the system where the data is pre-processed for the future references and then the feature selection is done by the user where he will enter/select the various symptoms. Then the classification of those data is done with the help of machine learning algorithms such as Logistic regression. Then the data goes in the recommendation model, there it shows the risk analysis that is involved in the system and it also provides the probability estimation of the system such that it shows the various probability like how the system behaves when there are n number of predictions are done and it also does the recommendations for the patients from their final result and also from their symptoms like it can show what to use and what not to use from the given datasets and the final results. It predicts probable diseases by mining data sets such as Covid-19, Chronic Kidney disease and heart Disease. To the best of our knowledge in the area of medical big data analytics none of the existing work focused on both data types. 4.1 ADVANTAGES OF PROPOSED SYSTEM

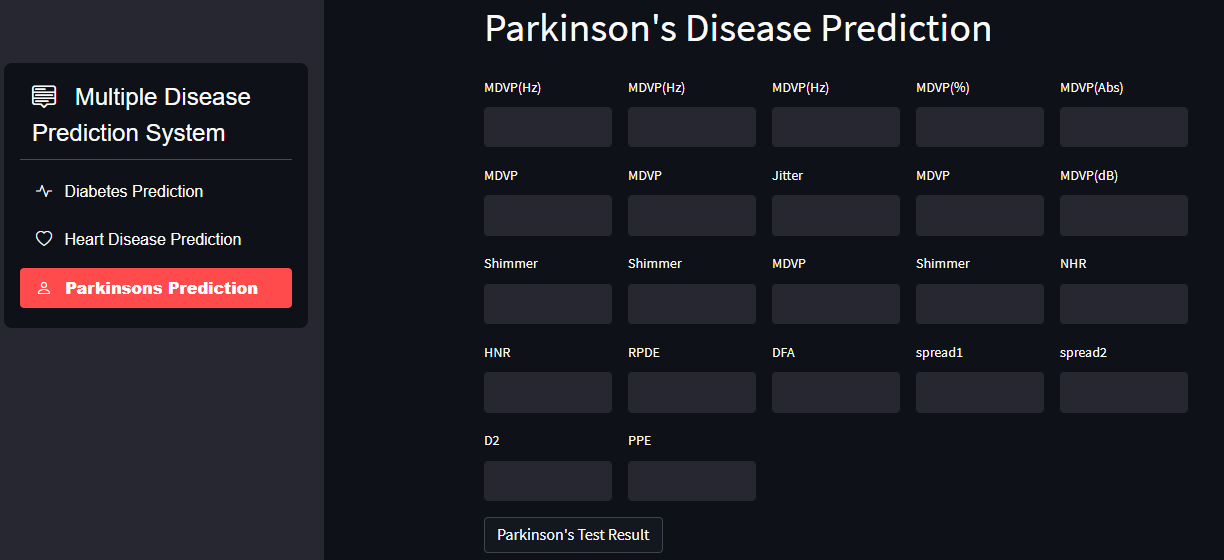
• Easily analyze the disease

• High Accuracy

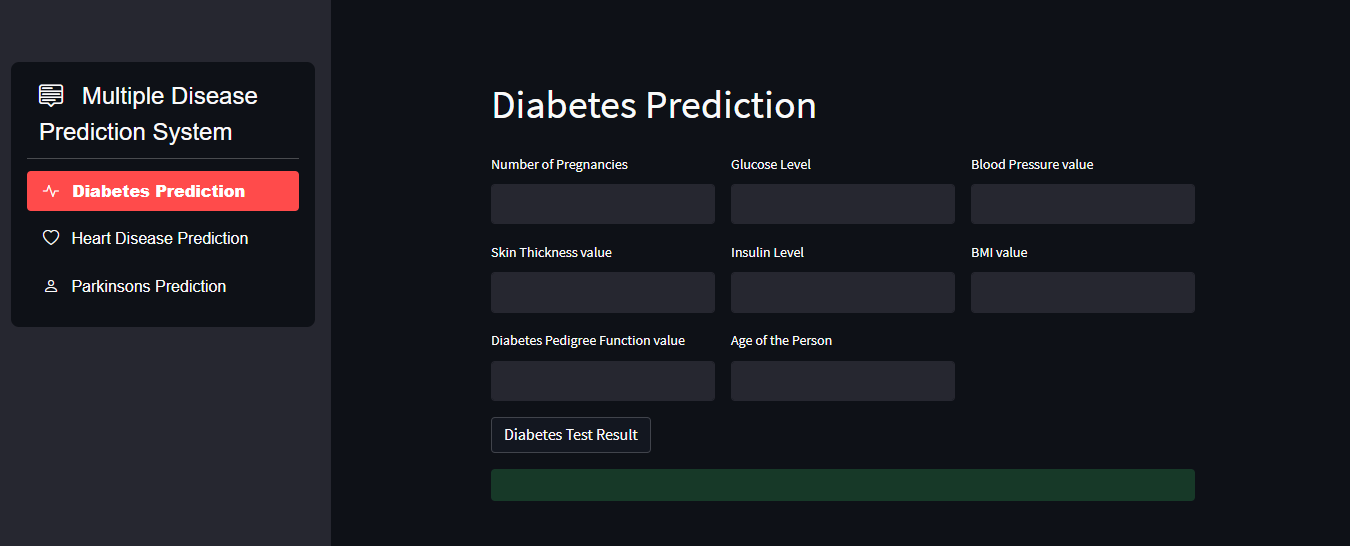
### BLOCK DIAGRAM

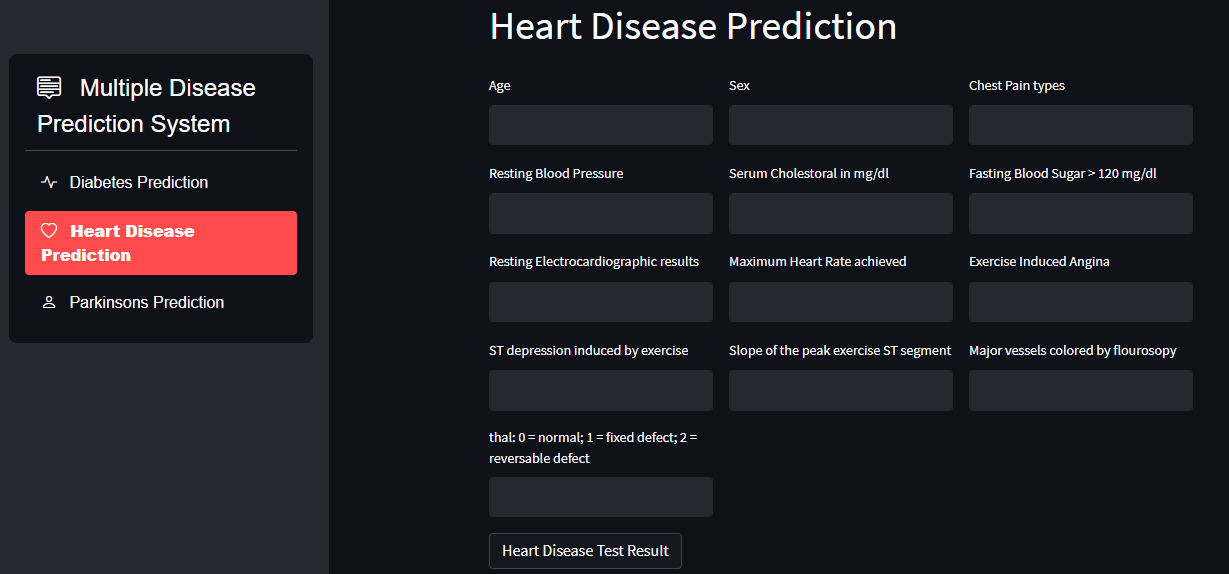
### 

### Multiple Disease Prediction System



**Working Diagram of Parkinson’s Disease Prediction**



**Working Diagram of diabetes Prediction** 

**Working Diagram of heart Disease Prediction**

**3.2 Code:**

**Multiple disease prediction system - diabetes**

Importing the Dependencies

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn import svm

from sklearn.metrics import accuracy\_score

Data Collection and Analysis

PIMA Diabetes Dataset

# loading the diabetes dataset to a pandas DataFrame

diabetes\_dataset = pd.read\_csv('diabetes.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()

Outcome

0 500

1 268

Name: count, dtype: int64

0 --> Non-Diabetic

1 --> Diabetic

diabetes\_dataset.groupby('Outcome').mean()

|  | **Pregnancies** | **Glucose** | **BloodPressure** | **SkinThickness** | **Insulin** | **BMI** | **DiabetesPedigreeFunction** | **Age** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Outcome** |  |  |  |  |  |  |  |  |
| **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 SkinThickness Insulin BMI \

0 6 148 72 35 0 33.6

1 1 85 66 29 0 26.6

2 8 183 64 0 0 23.3

3 1 89 66 23 94 28.1

4 0 137 40 35 168 43.1

.. ... ... ... ... ... ...

763 10 101 76 48 180 32.9

764 2 122 70 27 0 36.8

765 5 121 72 23 112 26.2

766 1 126 60 0 0 30.1

767 1 93 70 31 0 30.4

DiabetesPedigreeFunction Age

0 0.627 50

1 0.351 31

2 0.672 32

3 0.167 21

4 2.288 33

.. ... ...

763 0.171 63

764 0.340 27

765 0.245 30

766 0.349 47

767 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

Train Test Split

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)

Training the Model

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

#training the support vector Machine Classifier

classifier.fit(X\_train, Y\_train)

SVC

SVC(kernel='linear')

Model Evaluation

Accuracy Score

# 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 : 0.7833876221498371

# 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 : 0.7727272727272727

Making a Predictive System

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)

​

prediction = classifier.predict(input\_data\_reshaped)

print(prediction)

​

if (prediction[0] == 0):

print('The person is not diabetic')

else:

print('The person is diabetic')

[1]

The person is diabetic

Saving the trained model

import pickle

filename = 'diabetes\_model.sav'

pickle.dump(classifier, open(filename, 'wb'))

# loading the saved model

loaded\_model = pickle.load(open('diabetes\_model.sav', 'rb'))

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)

​

prediction = loaded\_model.predict(input\_data\_reshaped)

print(prediction)

​

if (prediction[0] == 0):

print('The person is not diabetic')

else:

print('The person is diabetic')

[1]

The person is diabetic

for column in X.columns:

print(column)

Pregnancies

Glucose

BloodPressure

SkinThickness

Insulin

BMI

DiabetesPedigreeFunction

Age

**Multiple disease prediction system – heart:**

Importing the Dependencies

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 and Processing

# loading the csv data to a Pandas DataFrame

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

memory usage: 33.3 KB

# checking for missing values

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 |
| **max** | 77.000000 | 1.000000 | 3.000000 | 200.000000 | 564.000000 | 1.000000 | 2.000000 | 202.000000 | 1.000000 | 6.200000 | 2.000000 | 4.000000 | 3.000000 | 1.000000 |

# checking the distribution of Target Variable

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

target

1 165

0 138

Name: count, dtype: int64

1 --> Defective Heart

0 --> Healthy Heart

Splitting the Features and Target

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

Y = heart\_data['target']

print(X)

age sex cp trestbps chol fbs restecg thalach exang oldpeak \

0 63 1 3 145 233 1 0 150 0 2.3

1 37 1 2 130 250 0 1 187 0 3.5

2 41 0 1 130 204 0 0 172 0 1.4

3 56 1 1 120 236 0 1 178 0 0.8

4 57 0 0 120 354 0 1 163 1 0.6

.. ... ... .. ... ... ... ... ... ... ...

298 57 0 0 140 241 0 1 123 1 0.2

299 45 1 3 110 264 0 1 132 0 1.2

300 68 1 0 144 193 1 1 141 0 3.4

301 57 1 0 130 131 0 1 115 1 1.2

302 57 0 1 130 236 0 0 174 0 0.0

slope ca thal

0 0 0 1

1 0 0 2

2 2 0 2

3 2 0 2

4 2 0 2

.. ... .. ...

298 1 0 3

299 1 0 3

300 1 2 3

301 1 1 3

302 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 & Test 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

model = LogisticRegression()

# training the LogisticRegression model with Training data

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

LogisticRegression

LogisticRegression()

Model Evaluation

Accuracy Score

# accuracy on training data

X\_train\_prediction = model.predict(X\_train)

print('Accuracy on Training data : ', training\_data\_accuracy)

Accuracy on Training data : 0.8512396694214877

# 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 : 0.819672131147541

Building a Predictive System

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

Saving the trained model

import pickle

filename = 'heart\_disease\_model.sav'

pickle.dump(model, open(filename, 'wb'))

# loading the saved model

loaded\_model = pickle.load(open('heart\_disease\_model.sav', 'rb'))

for column in X.columns:

print(column)

age

sex

cp

trestbps

chol

fbs

restecg

thalach

exang

oldpeak

slope

ca

thal

**Multiple disease prediction system – Parkinsons:**

Importing the Dependencies

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn import svm

from sklearn.metrics import accuracy\_score

Data Collection & Analysis

# loading the data from csv fil

parkinsons\_data = pd.read\_csv('parkinsons.csv')

# printing the first 5 rows of the dataframe

parkinsons\_data.head()

|  | **name** | **MDVP:Fo(Hz)** | **MDVP:Fhi(Hz)** | **MDVP:Flo(Hz)** | **MDVP:Jitter(%)** | **MDVP:Jitter(Abs)** | **MDVP:RAP** | **MDVP:PPQ** | **Jitter:DDP** | **MDVP:Shimmer** | **...** | **Shimmer:DDA** | **NHR** | **HNR** | **status** | **RPDE** | **DFA** | **spread1** | **spread2** | **D2** | **PPE** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | phon\_R01\_S01\_1 | 119.992 | 157.302 | 74.997 | 0.00784 | 0.00007 | 0.00370 | 0.00554 | 0.01109 | 0.04374 | ... | 0.06545 | 0.02211 | 21.033 | 1 | 0.414783 | 0.815285 | -4.813031 | 0.266482 | 2.301442 | 0.284654 |
| **1** | phon\_R01\_S01\_2 | 122.400 | 148.650 | 113.819 | 0.00968 | 0.00008 | 0.00465 | 0.00696 | 0.01394 | 0.06134 | ... | 0.09403 | 0.01929 | 19.085 | 1 | 0.458359 | 0.819521 | -4.075192 | 0.335590 | 2.486855 | 0.368674 |
| **2** | phon\_R01\_S01\_3 | 116.682 | 131.111 | 111.555 | 0.01050 | 0.00009 | 0.00544 | 0.00781 | 0.01633 | 0.05233 | ... | 0.08270 | 0.01309 | 20.651 | 1 | 0.429895 | 0.825288 | -4.443179 | 0.311173 | 2.342259 | 0.332634 |
| **3** | phon\_R01\_S01\_4 | 116.676 | 137.871 | 111.366 | 0.00997 | 0.00009 | 0.00502 | 0.00698 | 0.01505 | 0.05492 | ... | 0.08771 | 0.01353 | 20.644 | 1 | 0.434969 | 0.819235 | -4.117501 | 0.334147 | 2.405554 | 0.368975 |
| **4** | phon\_R01\_S01\_5 | 116.014 | 141.781 | 110.655 | 0.01284 | 0.00011 | 0.00655 | 0.00908 | 0.01966 | 0.06425 | ... | 0.10470 | 0.01767 | 19.649 | 1 | 0.417356 | 0.823484 | -3.747787 | 0.234513 | 2.332180 | 0.410335 |

5 rows × 24 columns

# number of rows and columns in the dataframe

parkinsons\_data.shape

(195, 24)

# getting more information about the dataset

parkinsons\_data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 195 entries, 0 to 194

Data columns (total 24 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 name 195 non-null object

1 MDVP:Fo(Hz) 195 non-null float64

2 MDVP:Fhi(Hz) 195 non-null float64

3 MDVP:Flo(Hz) 195 non-null float64

4 MDVP:Jitter(%) 195 non-null float64

5 MDVP:Jitter(Abs) 195 non-null float64

6 MDVP:RAP 195 non-null float64

7 MDVP:PPQ 195 non-null float64

8 Jitter:DDP 195 non-null float64

9 MDVP:Shimmer 195 non-null float64

10 MDVP:Shimmer(dB) 195 non-null float64

11 Shimmer:APQ3 195 non-null float64

12 Shimmer:APQ5 195 non-null float64

13 MDVP:APQ 195 non-null float64

14 Shimmer:DDA 195 non-null float64

15 NHR 195 non-null float64

16 HNR 195 non-null float64

17 status 195 non-null int64

18 RPDE 195 non-null float64

19 DFA 195 non-null float64

20 spread1 195 non-null float64

21 spread2 195 non-null float64

22 D2 195 non-null float64

23 PPE 195 non-null float64

dtypes: float64(22), int64(1), object(1)

memory usage: 36.7+ KB

# checking for missing values in each column

parkinsons\_data.isnull().sum()

name 0

MDVP:Fo(Hz) 0

MDVP:Fhi(Hz) 0

MDVP:Flo(Hz) 0

MDVP:Jitter(%) 0

MDVP:Jitter(Abs) 0

MDVP:RAP 0

MDVP:PPQ 0

Jitter:DDP 0

MDVP:Shimmer 0

MDVP:Shimmer(dB) 0

Shimmer:APQ3 0

Shimmer:APQ5 0

MDVP:APQ 0

Shimmer:DDA 0

NHR 0

HNR 0

status 0

RPDE 0

DFA 0

spread1 0

spread2 0

D2 0

PPE 0

dtype: int64

# getting some statistical measures about the data

parkinsons\_data.describe()

|  | **MDVP:Fo(Hz)** | **MDVP:Fhi(Hz)** | **MDVP:Flo(Hz)** | **MDVP:Jitter(%)** | **MDVP:Jitter(Abs)** | **MDVP:RAP** | **MDVP:PPQ** | **Jitter:DDP** | **MDVP:Shimmer** | **MDVP:Shimmer(dB)** | **...** | **Shimmer:DDA** | **NHR** | **HNR** | **status** | **RPDE** | **DFA** | **spread1** | **spread2** | **D2** | **PPE** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | ... | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 |
| **mean** | 154.228641 | 197.104918 | 116.324631 | 0.006220 | 0.000044 | 0.003306 | 0.003446 | 0.009920 | 0.029709 | 0.282251 | ... | 0.046993 | 0.024847 | 21.885974 | 0.753846 | 0.498536 | 0.718099 | -5.684397 | 0.226510 | 2.381826 | 0.206552 |
| **std** | 41.390065 | 91.491548 | 43.521413 | 0.004848 | 0.000035 | 0.002968 | 0.002759 | 0.008903 | 0.018857 | 0.194877 | ... | 0.030459 | 0.040418 | 4.425764 | 0.431878 | 0.103942 | 0.055336 | 1.090208 | 0.083406 | 0.382799 | 0.090119 |
| **min** | 88.333000 | 102.145000 | 65.476000 | 0.001680 | 0.000007 | 0.000680 | 0.000920 | 0.002040 | 0.009540 | 0.085000 | ... | 0.013640 | 0.000650 | 8.441000 | 0.000000 | 0.256570 | 0.574282 | -7.964984 | 0.006274 | 1.423287 | 0.044539 |
| **25%** | 117.572000 | 134.862500 | 84.291000 | 0.003460 | 0.000020 | 0.001660 | 0.001860 | 0.004985 | 0.016505 | 0.148500 | ... | 0.024735 | 0.005925 | 19.198000 | 1.000000 | 0.421306 | 0.674758 | -6.450096 | 0.174351 | 2.099125 | 0.137451 |
| **50%** | 148.790000 | 175.829000 | 104.315000 | 0.004940 | 0.000030 | 0.002500 | 0.002690 | 0.007490 | 0.022970 | 0.221000 | ... | 0.038360 | 0.011660 | 22.085000 | 1.000000 | 0.495954 | 0.722254 | -5.720868 | 0.218885 | 2.361532 | 0.194052 |
| **75%** | 182.769000 | 224.205500 | 140.018500 | 0.007365 | 0.000060 | 0.003835 | 0.003955 | 0.011505 | 0.037885 | 0.350000 | ... | 0.060795 | 0.025640 | 25.075500 | 1.000000 | 0.587562 | 0.761881 | -5.046192 | 0.279234 | 2.636456 | 0.252980 |
| **max** | 260.105000 | 592.030000 | 239.170000 | 0.033160 | 0.000260 | 0.021440 | 0.019580 | 0.064330 | 0.119080 | 1.302000 | ... | 0.169420 | 0.314820 | 33.047000 | 1.000000 | 0.685151 | 0.825288 | -2.434031 | 0.450493 | 3.671155 | 0.527367 |

8 rows × 23 columns

# distribution of target Variable

parkinsons\_data['status'].value\_counts()

status

1 147

0 48

Name: count, dtype: int64

1 --> Parkinson's Positive

0 --> Healthy

print(parkinsons\_data['status'].unique())

​

[1 0]

numeric\_data = parkinsons\_data.select\_dtypes(include='number')

grouped\_mean = numeric\_data.groupby('status').mean()

​

Data Pre-Processing

Separating the features & Target

X = parkinsons\_data.drop(columns=['name','status'], axis=1)

Y = parkinsons\_data['status']

print(X)

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

MDVP:Shimmer(dB) ... MDVP:APQ Shimmer:DDA NHR HNR RPDE \

0 0.426 ... 0.02971 0.06545 0.02211 21.033 0.414783

1 0.626 ... 0.04368 0.09403 0.01929 19.085 0.458359

2 0.482 ... 0.03590 0.08270 0.01309 20.651 0.429895

3 0.517 ... 0.03772 0.08771 0.01353 20.644 0.434969

4 0.584 ... 0.04465 0.10470 0.01767 19.649 0.417356

.. ... ... ... ... ... ... ...

190 0.405 ... 0.02745 0.07008 0.02764 19.517 0.448439

191 0.263 ... 0.01879 0.04812 0.01810 19.147 0.431674

192 0.256 ... 0.01667 0.03804 0.10715 17.883 0.407567

193 0.241 ... 0.01588 0.03794 0.07223 19.020 0.451221

194 0.190 ... 0.01373 0.03078 0.04398 21.209 0.462803

DFA spread1 spread2 D2 PPE

0 0.815285 -4.813031 0.266482 2.301442 0.284654

1 0.819521 -4.075192 0.335590 2.486855 0.368674

2 0.825288 -4.443179 0.311173 2.342259 0.332634

3 0.819235 -4.117501 0.334147 2.405554 0.368975

4 0.823484 -3.747787 0.234513 2.332180 0.410335

.. ... ... ... ... ...

190 0.657899 -6.538586 0.121952 2.657476 0.133050

191 0.683244 -6.195325 0.129303 2.784312 0.168895

192 0.655683 -6.787197 0.158453 2.679772 0.131728

193 0.643956 -6.744577 0.207454 2.138608 0.123306

194 0.664357 -5.724056 0.190667 2.555477 0.148569

[195 rows x 22 columns]

print(Y)

0 1

1 1

2 1

3 1

4 1

..

190 0

191 0

192 0

193 0

194 0

Name: status, Length: 195, dtype: int64

Splitting the data to training data & Test data

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

print(X.shape, X\_train.shape, X\_test.shape)

(195, 22) (156, 22) (39, 22)

Model Training

Support Vector Machine Model

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

# training the SVM model with training data

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

SVC

SVC(kernel='linear')

Model Evaluation

Accuracy Score

# 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 : 0.8717948717948718

# accuracy score on training 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

Building a Predictive System

input\_data = (197.07600,206.89600,192.05500,0.00289,0.00001,0.00166,0.00168,0.00498,0.01098,0.09700,0.00563,0.00680,0.00802,0.01689,0.00339,26.77500,0.422229,0.741367,-7.348300,0.177551,1.743867,0.085569)

​

# changing input data to a numpy array

input\_data\_as\_numpy\_array = np.asarray(input\_data)

​

# reshape the numpy array

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 Parkinsons Disease")

​

else:

print("The Person has Parkinsons")

​

[0]

The Person does not have Parkinsons Disease

Saving the trained model

import pickle

filename = 'parkinsons\_model.sav'

pickle.dump(model, open(filename, 'wb'))

# loading the saved model

loaded\_model = pickle.load(open('parkinsons\_model.sav', 'rb'))

for column in X.columns:

print(column)

MDVP:Fo(Hz)

MDVP:Fhi(Hz)

MDVP:Flo(Hz)

MDVP:Jitter(%)

MDVP:Jitter(Abs)

MDVP:RAP

MDVP:PPQ

Jitter:DDP

MDVP:Shimmer

MDVP:Shimmer(dB)

Shimmer:APQ3

Shimmer:APQ5

MDVP:APQ

Shimmer:DDA

NHR

HNR

RPDE

DFA

spread1

spread2

D2

PPE

**Multiple Disease Prediction [Deploying]:**

import pickle

#import joblib

import streamlit as st

from streamlit\_option\_menu import option\_menu

import os

#Loading the saved models

file\_path = os.path.join(os.path.abspath(os.getcwd()), 'Saved Model', 'diabetes\_model.sav')

diabetes\_model = pickle.load(open(file\_path, 'rb'))

diabetes\_model=pickle.load(open('C:/Users/madhu/OneDrive/Desktop\ML/Saved Models/diabetes\_model.sav','rb'))

heart\_disease\_model=pickle.load(open('C:/Users/madhu/OneDrive/Desktop/ML/Saved Model/heart\_disease\_model.sav','rb'))

parkisons\_model=pickle.load(open('C:/Users/madhu/OneDrive/Desktop/ML/Saved Model/parkisons\_model.sav','rb'))

# sidebar for navigation

with st.sidebar:

selected = option\_menu('Multiple Disease Prediction System',

['Diabetes Prediction',

'Heart Disease Prediction',

'Parkinsons Prediction'],

icons=['activity','heart','person'],

default\_index=0)

# Diabetes Prediction Page

if (selected == 'Diabetes Prediction'):

# page title

st.title('Diabetes Prediction using ML')

# getting the input data from the user

col1, col2, col3 = st.columns(3)

with col1:

Pregnancies = st.text\_input('Number of Pregnancies')

with col2:

Glucose = st.text\_input('Glucose Level')

with col3:

BloodPressure = st.text\_input('Blood Pressure value')

with col1:

SkinThickness = st.text\_input('Skin Thickness value')

with col2:

Insulin = st.text\_input('Insulin Level')

with col3:

BMI = st.text\_input('BMI value')

with col1:

DiabetesPedigreeFunction = st.text\_input('Diabetes Pedigree Function value')

with col2:

Age = st.text\_input('Age of the Person')

# code for Prediction

diab\_diagnosis = ''

# creating a button for Prediction

if st.button('Diabetes Test Result'):

diab\_prediction = diabetes\_model.predict([[Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age]])

if (diab\_prediction[0] == 1):

diab\_diagnosis = 'The person is diabetic'

else:

diab\_diagnosis = 'The person is not diabetic'

st.success(diab\_diagnosis)

# Heart Disease Prediction Page

if (selected == 'Heart Disease Prediction'):

# page title

st.title('Heart Disease Prediction using ML')

col1, col2, col3 = st.columns(3)

with col1:

age = st.text\_input('Age')

with col2:

sex = st.text\_input('Sex')

with col3:

cp = st.text\_input('Chest Pain types')

with col1:

trestbps = st.text\_input('Resting Blood Pressure')

with col2:

chol = st.text\_input('Serum Cholestoral in mg/dl')

with col3:

fbs = st.text\_input('Fasting Blood Sugar > 120 mg/dl')

with col1:

restecg = st.text\_input('Resting Electrocardiographic results')

with col2:

thalach = st.text\_input('Maximum Heart Rate achieved')

with col3:

exang = st.text\_input('Exercise Induced Angina')

with col1:

oldpeak = st.text\_input('ST depression induced by exercise')

with col2:

slope = st.text\_input('Slope of the peak exercise ST segment')

with col3:

ca = st.text\_input('Major vessels colored by flourosopy')

with col1:

thal = st.text\_input('thal: 0 = normal; 1 = fixed defect; 2 = reversable defect')

# code for Prediction

heart\_diagnosis = ''

# creating a button for Prediction

if st.button('Heart Disease Test Result'):

heart\_prediction = heart\_disease\_model.predict([[age, sex, cp, trestbps, chol, fbs, restecg,thalach,exang,oldpeak,slope,ca,thal]])

if (heart\_prediction[0] == 1):

heart\_diagnosis = 'The person is having heart disease'

else:

heart\_diagnosis = 'The person does not have any heart disease'

st.success(heart\_diagnosis)

# Parkinson's Prediction Page

if (selected == "Parkinsons Prediction"):

# page title

st.title("Parkinson's Disease Prediction using ML")

col1, col2, col3, col4, col5 = st.columns(5)

with col1:

fo = st.text\_input('MDVP:Fo(Hz)')

with col2:

fhi = st.text\_input('MDVP:Fhi(Hz)')

with col3:

flo = st.text\_input('MDVP:Flo(Hz)')

with col4:

Jitter\_percent = st.text\_input('MDVP:Jitter(%)')

with col5:

Jitter\_Abs = st.text\_input('MDVP:Jitter(Abs)')

with col1:

RAP = st.text\_input('MDVP:RAP')

with col2:

PPQ = st.text\_input('MDVP:PPQ')

with col3:

DDP = st.text\_input('Jitter:DDP')

with col4:

Shimmer = st.text\_input('MDVP:Shimmer')

with col5:

Shimmer\_dB = st.text\_input('MDVP:Shimmer(dB)')

with col1:

APQ3 = st.text\_input('Shimmer:APQ3')

with col2:

APQ5 = st.text\_input('Shimmer:APQ5')

with col3:

APQ = st.text\_input('MDVP:APQ')

with col4:

DDA = st.text\_input('Shimmer:DDA')

with col5:

NHR = st.text\_input('NHR')

with col1:

HNR = st.text\_input('HNR')

with col2:

RPDE = st.text\_input('RPDE')

with col3:

DFA = st.text\_input('DFA')

with col4:

spread1 = st.text\_input('spread1')

with col5:

spread2 = st.text\_input('spread2')

with col1:

D2 = st.text\_input('D2')

with col2:

PPE = st.text\_input('PPE')

# code for Prediction

parkinsons\_diagnosis = ''

# creating a button for Prediction

if st.button("Parkinson's Test Result"):

parkinsons\_prediction = parkisons\_model.predict([[fo, fhi, flo, Jitter\_percent, Jitter\_Abs, RAP, PPQ,DDP,Shimmer,Shimmer\_dB,APQ3,APQ5,APQ,DDA,NHR,HNR,RPDE,DFA,spread1,spread2,D2,PPE]])

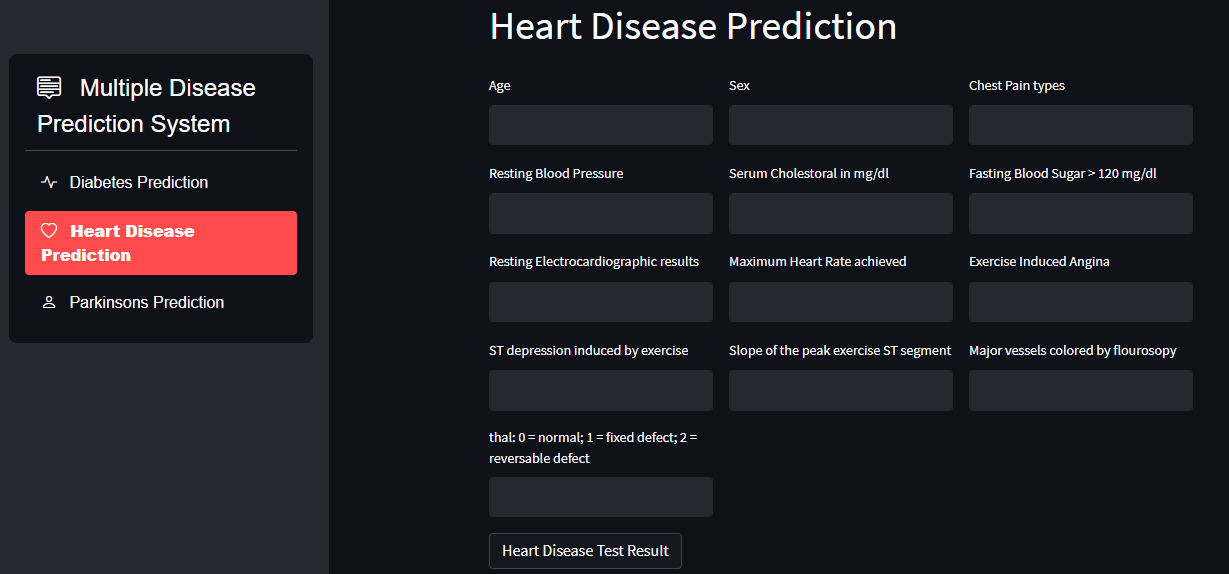
if (parkinsons\_prediction[0] == 1):

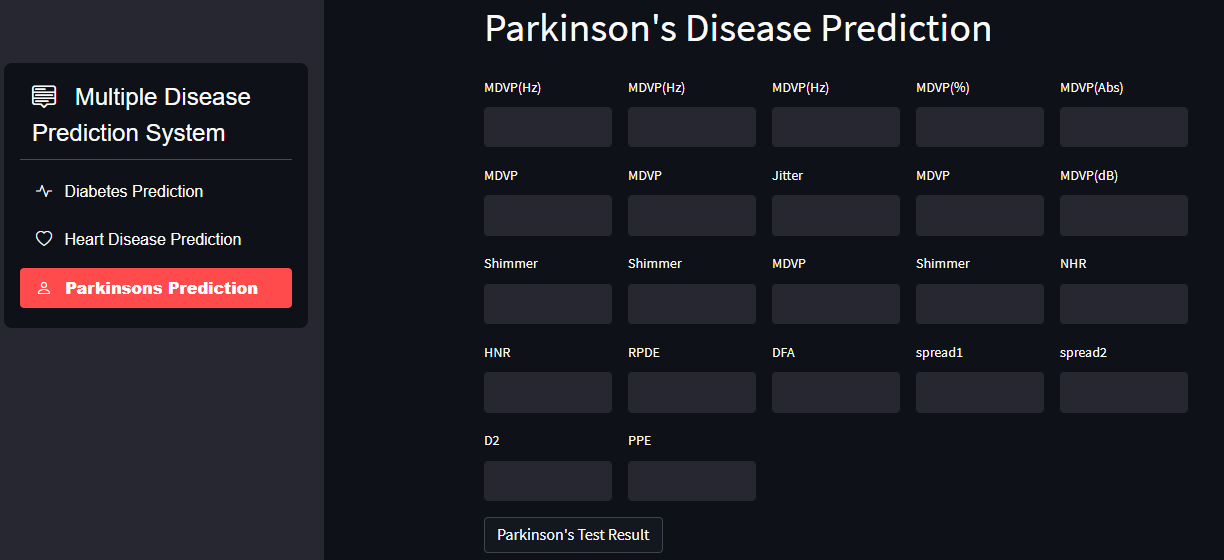
parkinsons\_diagnosis = "The person has Parkinson's disease"

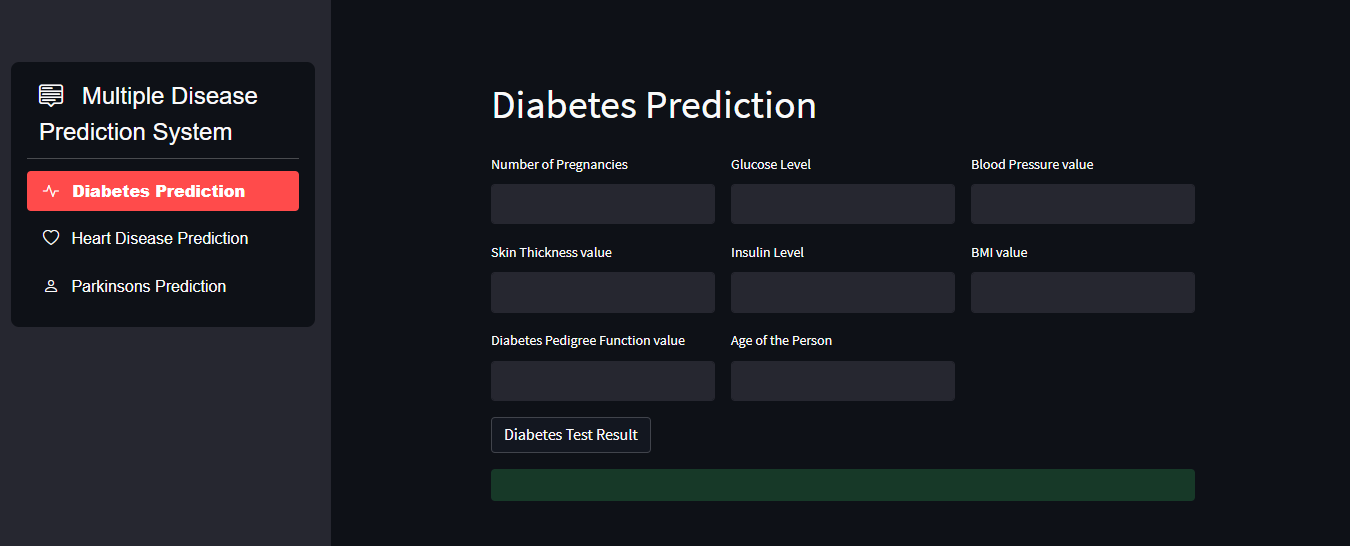
else:

parkinsons\_diagnosis = "The person does not have Parkinson's disease"

st.success(parkinsons\_diagnosis)







### DISCUSSION

1. Experimental Setup

All the experimental cases are developed in Python in a congested environment using Anaconda tools. The competing classification approach and various feature extraction techniques are also used, and the system is configured with an Intel Core i5-6200U processor running at 2.30 GHz and 8GB of RAM.

1. Dataset

The disease dataset are downloaded from Github, kaagle and other ML website. And as per industry standards train set and test are prepared. By using Scikit learn train, test, split method to split the data as 70 % for training and 30 % for testing are divided. Example of Diabetic Disease: Diabetes feature train, diabetes feature test, Diabetes label train, diabetes label test=train test split (diabetes features, diabetes label, test size=0.3, train size=0.7)

1. Evaluation Methods

For the experiment's performance assessment. First, we identify True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). True positive refers to the number of cases successfully forecasted as necessary, false positive refers to the number of instances mistakenly predicted as required, and so on. The following are the four measurements that may be obtained: accuracy, precision, recall, and F1-measure.

Accuracy = (TP+TN)/(TP+TN+FP+FN)

Precision = TP/(TP+FP)

Recall = TP/(TP+FN) F1

Score = TN/(TN+FP)

1. Result

When the patient adds the disease-specific parameter, it will indicate whether the patient has the ailment in question. The parameters will display the necessary value range, and if the value is outside of that range, is invalid, or is empty, a warning message will appear, advising the user to input a proper value.

### CONCLUSION

### We used standard machine learning algorithms to categorize patient data since, in the modern medical world, medical data is expanding greatly and must be processed to provide precise disease predictions based on symptoms. By providing the input of patient records, which aid in understanding the degree of disease risk prediction, we were able to produce an accurate general disease risk prediction. Because of this technique, disease and risk prediction may be accomplished with little effort and expense. We compare the outcomes of the different algorithms in terms of accuracy and processing time. The accuracy of the CNN algorithm is higher than that of the other machine learning algorithm, and CNN's processing time is lower than that of others. So, in terms of accuracy and timing, CNN is superior to other machine learning algorithm. In the future, we'll include more diseases and forecast the likelihood that a patient would develop each one.

### REFERENCE

[1] M. Chen, Y. Hao, K. Hwang, L. Wang and L. Wang, "Disease Prediction by Machine Learning Over Big Data From Healthcare Communities," in IEEE Access, vol. 5, pp. 8869-8879, 2017, doi: 10.1109/ACCESS.2017.2694446.

[2] Qian, Buyue & Wang, Xiang & Cao, Nan & Li, Hongfei & Jiang, Yu-Gang. (2014). A relative similarity based method for interactive patient risk prediction. Data Mining and Knowledge Discovery. 29. 10.1007/s10618-014-0379-5.

[3] M. Chen, Y. Ma, Y. Li, D. Wu, Y. Zhang and C. -H. Youn, "Wearable 2.0: Enabling Human-Cloud Integration in Next Generation Healthcare Systems," in IEEE Communications Magazine, vol. 55, no. 1, pp. 54-61, January 2017, doi: 10.1109/MCOM.2017.1600410CM.

[4] Y. Zhang, M. Qiu, C. -W. Tsai, M. M. Hassan and A. Alamri, "Health-CPS: Healthcare Cyber-Physical System Assisted by Cloud and Big Data," in IEEE Systems Journal, vol. 11, no. 1, pp. 88-95, March 2017, doi: 10.1109/JSYST.2015.2460747.