# A FRAMEWORK FOR MULTI DISEASE PREDICTION SYSTEM USING RANDOM FOREST ALGORITHM

**A PROJECT REPORT**

**Submitted by**

**ARVINTH S**

**17ITR011**

**BHARANEESHWAR B**

**17ITR014**

**HARI PRASATH K V**

**17ITR033**

***in partial fulfilment of the requirements***

***for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**SCHOOL OF COMMUNICATION AND COMPUTER SCIENCES**

Estd : 1984

**KONGU ENGINEERING COLLEGE**

**(Autonomous)**

**PERUNDURAI, ERODE – 638 060**

**APRIL 2021**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**KONGU ENGINEERING COLLEGE**

**(Autonomous)**

**PERUNDURAI, ERODE – 638 060**

**APRIL 2021**

## BONAFIDE CERTIFICATE

This is to certify that the Project Report entitled **A FRAMEWORK FOR MULTI DISEASE PREDICTION SYSTEM USING RANDOM FOREST ALGORITHM** is the bonafide record of project work done by **ARVINTH S (17ITR011), BHARANEESHWAR B (17ITR014),** and **HARI PRASATH K V (17ITR033)** in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in **INFORMATION TECHNOLOGY** of Anna University, Chennai during the year 2020-2021.

**SUPERVISOR** **HEAD OF THE DEPARTMENT**

**(Signature with seal)**

Date:

Submitted for the end semester viva voce examination held on ____________

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**KONGU ENGINEERING COLLEGE**

**(Autonomous)**

**PERUNDURAI ERODE – 638 060**

**APRIL 2021**

## DECLARATION

We affirm that the Project Report titled **A FRAMEWORK FOR MULTI DISEASE PREDICTION SYSTEM USING RANDOM FOREST ALGORITHM** being submitted in partial fulfilment of the requirements for the award of Bachelor of Technology is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**ARVINTH S**
**(17ITR011)**

**BHARANEESHWAR B**
**(17ITR014)**

**HARI PRASATH K V**
**(17ITR033)**

**Date:**

I certify that the declaration made by the above candidates is true to the best of my knowledge.

Date: Name and Signature of the Supervisor with seal

## ABSTRACT

Health is one of the important factors to be considered by an individual. With the increasing number of diseases and the population, medical practitioners find it hard to diagnose many numbers of diseases and predict whether the individual is suffering from the disease or not, over intensive population growth. Here comes the need for dynamic Health care systems, which are established to meet the health requirements of the population. Such systems are built with technology, health care, and data, that have to be processed in a smart, efficient, and precise course of action. Prediction system is the best technology that can meet the level of expectation in this field. There are many models proposed for single disease identification. However, very little is proposed concerning multiple disease identification. The main aim of the disease prediction model which identifies the multiple disease possibility by analyzing the health record of the patient. We consider the diseases such as Heart disease, Diabetes, and Kidney disease using some of the basic parameters such as Pulse Rate, Cholesterol, Blood Pressure, Heart Rate, etc., and also the risk factors associated with the disease can be found using prediction model with good accuracy and Precision. Many models are created by python pickling method and compared by applying it to multiple data sets and using different performance measures.

## ACKNOWLEDGEMENT

First and foremost, we acknowledge the abundant grace and presence of Almighty throughout different phases of the project and its successful completion.

We wish to express our extreme gratefulness to our beloved Correspondent **Thiru. P. SACHITHANANDAN,** and all the trust members of Kongu Vellalar Institute of Technology Trust for providing all the necessary facilities to complete the project successfully.

We express our deep sense of gratitude to our beloved Principal **Dr. V. BALUSAMY B.E(Hons)., M.Tech., Ph.D.,** for providing us an opportunity to complete the project.

We express our gratitude to **Dr. R. THANGARAJAN M.E., Ph.D.,** Head of the Department, Department of Information Technology, for his valuable suggestions.

We are thankful to our project coordinators **Dr. P. SURESH M.E., Ph.D.,** and **Dr. K. LALITHA M.E., Ph.D.,** for their valuable guidance and support to complete our project successfully.

We are highly indebted to **Dr. R. SHANTHAKUMARI M.E., Ph.D.,** Department of Information Technology, for her valuable supervision and advice for the fruitful completion of the project.

Finally, we extend our love and thanks to our Parents and Friends for their patience in teaching, love, encouragement and support.

# TABLE OF CONTENTS

# LIST OF ABBREVIATIONS

| ABBREVIATION | EXPANSION |
|---|---|
| AB | AdaBoost |
| AI | Artificial Intelligence |
| ANOVA | ANalysis Of VAriance |
| CNN | Convolutional Neural Networks |
| DT | Decision Tree |
| GUI | Graphical User Interface |
| KNN | K- Nearest Neighbor |
| LR | Logistic Regression |
| ML | Machine Learning |
| NB | Naïve Bayes |
| RAM | Random Access Memory |
| RFA | Random Forest Algorithm |
| SVM | Support Vector Machine |

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1 INTRODUCTION

There are many existing systems/models that predicts/identifies a single disease. There are less systems which predict more than one disease and with rise in population the no of disease that are spreading is increasing day by day so, there should be a system that predicts more than one disease in order to provide better treatment for the patient's by analyzing their health records. Predicting more than one disease helps both the patient and the doctor in several ways. This is because Doctors can easily analyze the patient's health records and treat their patients in a better manner. Moreover, patients can also prepare themselves mentally and physically for their treatment. Single disease prediction models can predict and identify only one particular disease and even if the patients are cured completely sometimes, they will have some side effects or some other health issues which if left as it is there is a possibility that it may lead to death or severe health conditions. While working with the analysis of existing systems in the division of health care analysis systems, it is unequivocal that one disease is predicted once. Most of the articles focus on a specific disease once at a time.

If an organization needs to analyze their patient's health report, there arises a need to deploy many models for each disease. The approach which is made use in the existing systems is practicable for one disease but not for more than one disease. The mortality rate is increased since the precise disease is not diagnosed properly. Even though the patient recovered from one disease may also suffer from other diseases.

Few existing models use some specific parameters to analyze the disease which will not look into the possible disease that is caused due to the effect of the previous disease. If we take diabetes, there is a probability of the diseases like hearing loss, heart disease, and dementia. In this model, we consider the diagnosis of heart disease, diabetes, and kidney disease. Also, other diseases like skin-related diseases, COVID-19, and other diseases can be included in the model.

The analysis of the system is designed adaptable in such a way that many diseases can be included subsequently. The appropriate model file of that disease should be loaded to the UI.

When building a new model to predict disease, the programmer has to devise a python pickling technique to carry through the behavior of the model. While using TKinter UI, the developer loads the file that is pickled to retrieve the behavior of the model. If the user needs to analyze the health condition of the patient, either they can predict a specific disease or utilizing the report which comprises of the relevant features considered to diagnose the disease. The main motto is to reduce the increasing mortality rate by alerting the patients beforehand according to their status of health.

### 1.1.1 MACHINE LEARNING

In the area of classification, Machine learning has been achieving the most compromising and the successful results. Machine learning (ML) is the study of computer algorithms that improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks. An algorithm that implements classification, especially in a concrete implementation, is known as a classifier. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category. Recently, Classification has found its zenith because of the advances in the machine learning algorithms. Many algorithms are available for classification which outperforms each other based on the features of the classification. Their accuracy is also very high and shows tremendous results in the field of classification.

## 1.2 OBJECTIVE

The objective is creating a framework established by NumPy, Pandas, Matplotlib, TKinter, dataset, and trained model. The purpose of the system is to minimize the number of human computer interactions, speed up the identification process and improve the usability of the graphical user interface compared to existing manual systems. The system is initially developed using a Python Application for the prediction of multiple diseases based on certain features. To predict and classify a disease, different machine learning algorithms have been tested and retrained.

## 1.3 SCOPE

A unified approach that can predict the chances of having diseases namely Heart disease, Diabetes, Kidney disease, and provides the result as positive or negative, where all features are simply concatenated and fed independently to the algorithm. Prediction of Multiple Diseases are now recently becoming very attention seeking amidst many organizations like Hospitals where the report producing being done very slowly due to large number of patients these days. The main scope of the project is to reduce the mortality rate by alerting the patients.

# CHAPTER 2

# SYSTEM ANALYSIS

## 2.1 LITERATURE REVIEW

1. Akkem, Y.: “Multi Disease Prediction Model by using Machine Learning and Flask API”, In this desk work, the diseases namely cancer, diabetes, Heart disease and diabetic retinopathy are considered for the deployment of prediction model. The final model GUI is designed using Flask API in python and the backend is provided using python pickling technique. For diabetes analysis, logistic regression and for heart disease classification Random Forest algorithm and for cancer detection SVM are used. Diabetes retinopathy analysis contains retina images. The accuracy of all algorithms are found to be less. The user has to input the values of test reports every time to know the result which turns out again as a tedious process.

2. Karthikeyan, H., et. al., “Multi-disease prediction model using improved SVM-radial bias technique in healthcare monitoring system”, In this work, a system was experimented using with reduced set features of Chronic Kidney Disease, Diabetes and Heart Disease dataset using improved SVM-Radial bias kernel method, and also this system has compared with other machine learning techniques such as SVM-Linear, SVM-Polynomial, Random forest and Decision tree in R studio. The performance of all these machine learning algorithms has evaluated with accuracy, misclassification rate, precision, sensitivity and specificity. This needs to improve the accuracy.

3. Classification and Diagnosis of Diabetes: Standards of Medical Care in Diabetes—2020, American Diabetes Association Diabetes Care 2020, 43 (Suppl. 1): S14–S31 | https://doi.org/10.2337/dc20-S002, This article proposed by the American Diabetes Association paves the way to reduce the diabetes issues faced by people. This can be included in daily life style and followed for the prevention of diabetes.

4. Fontecha, J., et. al., “A usability study of a mHealth system for diabetes self-management based on framework analysis and usability problem taxonomy methods”, The article

proposed is prediction system for diabetes only using taxonomical methods. The accuracy of the proposed method in this system seems to be very less and the system flexibility needs to be enhanced.

5. Singh, Y. K., et. al., “Heart Disease Prediction System Using Random Forest, Advances in Computing and Data Sciences”, In this work, the heart disease is predicted using Random Forest algorithm. The highest accuracy achieved by random forest is less due to less features. The accuracy can be incremented and feasibility is improvised.

6. Devika, R., et. al., “Comparative Study of Classifier for Chronic Kidney Disease prediction using Naive Bayes, KNN and Random Forest”, In this administrative work, the comparison of algorithms is done for Kidney disease prediction using the kidney dataset from Kaggle repository. The accuracy is high for random forest algorithm and is less. The accuracy can be enhanced and also the flexibility needs some enhancements.

## 2.2 SUMMARY

The previous existing systems namely the Multi-Disease Predictors, Heart Disease Prediction, Kidney Disease Prediction and Diabetes prediction for a patient are less accurate and have some drawbacks. They have good performance but the drawbacks of predicting each disease separately for which the user have to give input of the patient’s report every time they need to check the disease. In order to improve the functionality and flexibility of the prediction system, a single interface with which the user can generate reports as positive or negative is built with the help of TKinter and backend using pickle files. By increasing the number of rows in the data set the prediction rate can be increased. If the report value in testing data is significantly less than that of a training set, it misses the prediction. So, high-quality and more training data sets covering multiple scale levels are required. The proposed method is not robust to noise.

# CHAPTER 3

# PROBLEM DEFINITION

## 3.1 EXISTING SYSTEM

The aim of the existing systems is to reduce the mortality rate by identifying the disease based on their health records. In this system, they have proposed a model that predicts whether a person has a disease or not using machine learning algorithms, deep learning algorithms and Flask API. The model analyzes the following diseases: Heart Disease, Diabetes, Diabetes Retinopathy and Breast Cancer. The importance of this article analysis is while analyzing the diseases all the parameters which causes the disease is included so it possible to detect the maximum effects which the disease will cause. The model uses the maximum parameters in order to identify/predict the disease more accurately. The model uses the following machine learning algorithms like Naïve Bayes, Random Forest, Logistic Regression, Decision Tree, and SVM. Each dataset is divided into two for Training and Testing. The training and testing are done using the above machine learning algorithms. The algorithm with best accuracy is traced out. On that basis, for diabetes analysis, heart disease, breast cancer and diabetes retinopathy, Logistic regression resulted 92% accuracy, Random forest yield 95% accuracy, Support vector machine yield 96% accuracy and tensorflow CNN produced 91% accuracy. All these trained models are pickled so that they can be loaded and used for real-time prediction in future. The final prediction model uses an API for the user to enter the patient's data (basically the parameters obtained from blood test, ECG, etc.,) to predict whether the patients is suffering from the disease or not. In that API, User is first allowed to select the disease they want to predict with their health records. After this, they can enter their parameters and finally prediction result will be displayed. If any user wants to proceed with another such process, they can continue with selection of other disease predictions.

## 3.2 PROBLEM STATEMENT

The Problem description for the Multi-Disease Prediction System is manual observation and interpretation of data may cause errors. These errors may cost a life. There are systems that are provided using machine learning but they are for only one disease. So, we are going for machine learning based multi-disease prediction system. Also, predicting disease individually takes more time than usual and becomes more tedious process. Since the previous methods detect each disease separately which creates a chance of not detecting other disease resulting in mortality.

## 3.3 PROPOSED SYSTEM

In this article, we have applied some machine learning algorithms to predict three diseases namely Heart Disease, Kidney Disease and Diabetes (Multi disease Prediction) with three different datasets. We have used the following algorithms for analysis: Naïve Bayes, KNN, Decision Tree, Adaboost, Random Forest, Logistic Regression, and SVM. In future several diseases can be added to this model. With the model trained, performance measures, namely, Accuracy, Precision, Recall, Time, F1-score and R-squared value. The user can save his/her time by using a single model to predict the disease instead of using more no of models. By using multi disease, the mortality rate can also be decreased. The aim of the project is to predict the presence of 3 diseases mentioned above, with one set of input data.

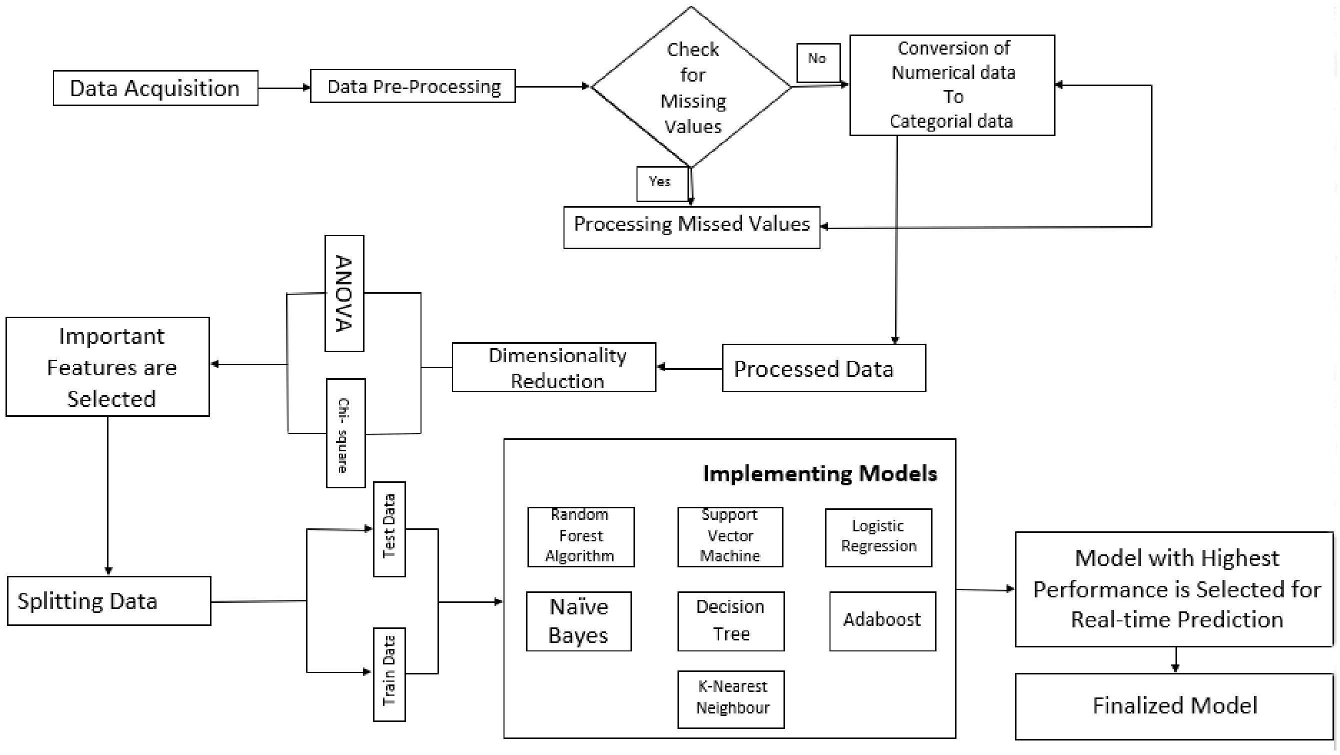


Figure. 3a Proposed System for Multiple Disease Prediction

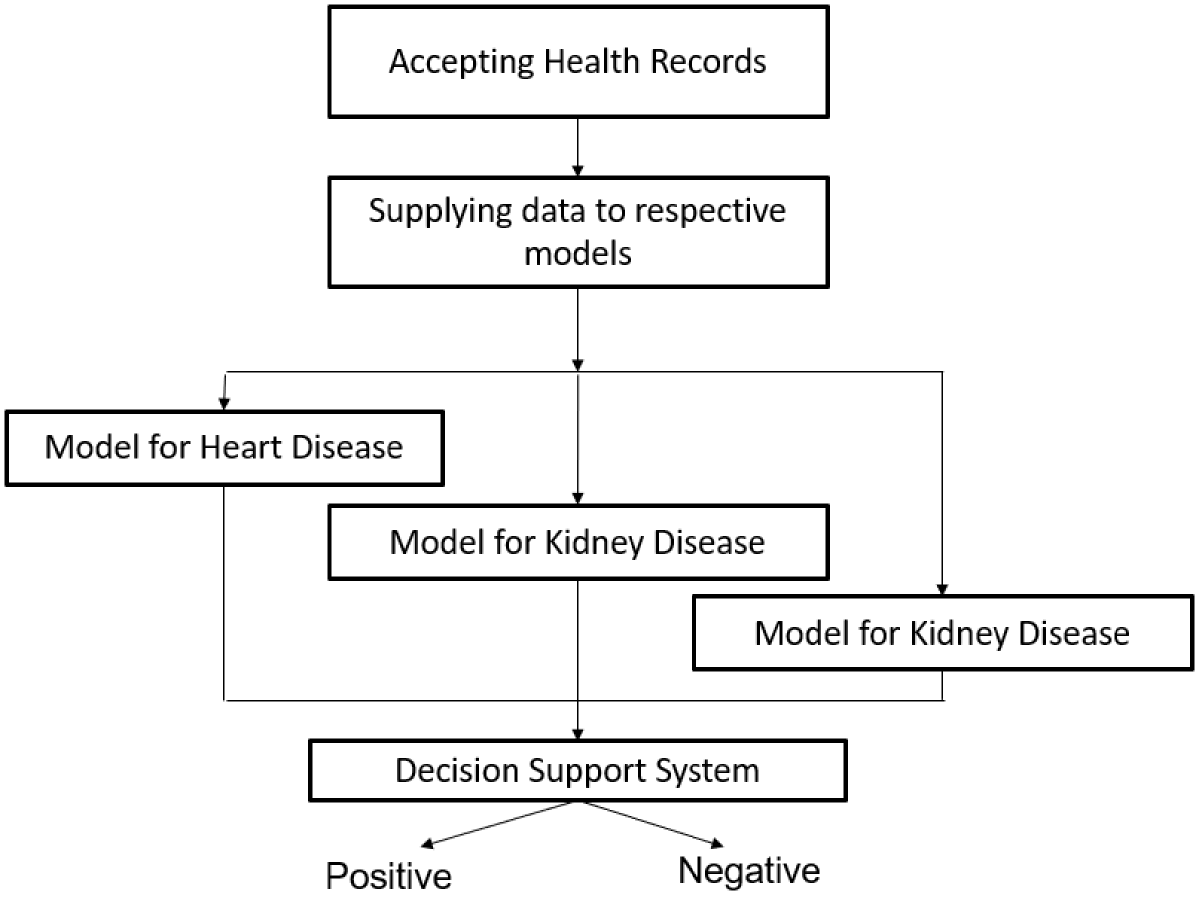


Figure. 3b Integration of Multiple Dataset in Final Model

Interpreting the results, we conclude with using Random Forest Algorithm for proposing the predictive system. The proposed model is depicted in Fig. 3b.

# CHAPTER 4

# SYSTEM REQUIREMENTS

## 4.1 HARDWARE REQUIREMENTS

| | |
|---|---|
| Processor | : Intel Core i5 |
| Speed | : 2.7 GHz |
| RAM | : 8 GB |
| Hard Disk | : 50 GB |

## 4.2 SOFTWARE REQUIREMENTS

| | |
|---|---|
| Operating System | : Windows 10, 64-bit |
| Programming Languages | : Python 3 |
| IDE | : Jupyter Notebook |
| Framework | : NumPy, Pandas, Matplotlib, TKinter |

## 4.3 SOFTWARE DESCRIPTION

### 4.3.1 PYTHON 3

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input never causes a segmentation fault. Instead, when the interpreter discovers an error; it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

### 4.3.2 JUPYTER NOTEBOOK

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through "Download As" in the web interface, via the nbconvert library or "jupyter nbconvert" command line interface in a shell. To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

Jupyter Notebook provides a browser-based REPL built upon a number of popular open-source libraries:

- IPython
- ØMQ
- Tornado (web server)
- jQuery
- Bootstrap (front-end framework)
- MathJax

The Notebook interface was added to IPython in the 0.12 release (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is similar to the notebook interface of other programs such as Maple, Mathematica, and SageMath, a computational interface style that originated with Mathematica in the 1980s. According to The Atlantic, Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

### 4.3.3 NUMPY

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

### 4.3.4 PANDAS

pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself. Wes McKinney started building what would become pandas at AQR Capital while he was a researcher there from 2007 to 2010. Pandas is mainly used for data analysis. Pandas allows importing data from various file formats such as comma-separated values, JSON, SQL, Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features.

### 4.3.5 MATPLOTLIB

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter. Since then, it has an active development community and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012 and was further joined by Thomas Caswell.

Matplotlib 2.0.x supports Python versions 2.7 through 3.6. Python 3 support started with Matplotlib 1.2. Matplotlib 1.4 is the last version to support Python 2.6. Matplotlib has pledged not to support Python 2 past 2020 by signing the Python 3 Statement.

### 4.3.6 TKINTER

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python. The name Tkinte*r* comes from Tk interface. Tkinter was written by Fredrik Lundh. Tkinter is free software released under a Python license. It has advantages over the Command Line Interface(CLI) where users interact with computers by writing commands using keyboard only and whose usage is more difficult than GUI. Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps,

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

# CHAPTER 5

# SYSTEM DESIGN

## 5.1 PROPOSED METHODOLOGY

In the health-care sector, the latest research proposes a method for forecasting various diseases. The method has dealt with Heart Disease, Diabetes, and Kidney Disease from the UCI dataset. Standards of Medical Datasets considered in Heart Disease, Diabetes and Kidney Disease respectively, as a research step. Fig. 3a. explains the Multiple disease prediction system.

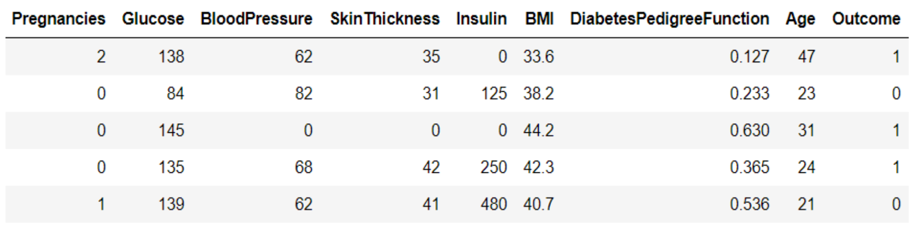
## 5.2 MODULE DESCRIPTION

The proposed predictive system constitutes of three sub modules as, Data pre-processing module, Feature selection module and Predictive system module.

### 5.2.1 DATASET PRE-PROCESSING

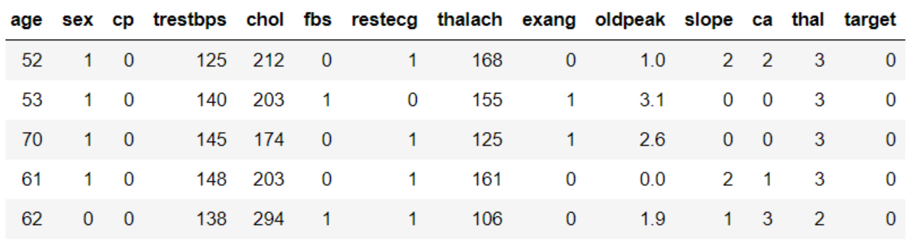
#### 5.2.1.1 DATASET ACQUISITION

Data set acquisition is the first step in building disease prediction model. For constructing a multi disease prediction model, 3 diseases in particular, three different datasets are acquired from UCI machine learning repository. All three datasets differed with respect to the dimensions, characteristics and the type of attributes. In this project, we have used three different datasets for three diseases. The datasets are Heart disease dataset, Kidney Disease and Diabetes dataset. The Diabetes dataset consist of 9 columns with outcome as the dependent variable. The Diabetes dataset is given Fig. 5a. The Heart Disease dataset consist of 14 columns with target as dependent variable. The heart disease dataset is shown in Fig. 5b. The kidney Disease dataset consist of 26 columns with classification as the dependent variable. The kidney disease dataset is shown in Fig. 5c. Train and test sets are prepared in accordance with industry requirements. The data is split into 80 percent for training and 20 percent for testing using the Scikit learn train test split process.



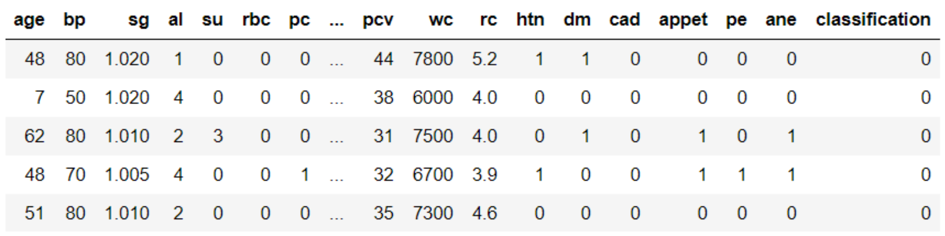
| Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|
| 2 | 138 | 62 | 35 | 0 | 33.6 | 0.127 | 47 | 1 |
| 0 | 84 | 82 | 31 | 125 | 38.2 | 0.233 | 23 | 0 |
| 0 | 145 | 0 | 0 | 0 | 44.2 | 0.630 | 31 | 1 |
| 0 | 135 | 68 | 42 | 250 | 42.3 | 0.365 | 24 | 1 |
| 1 | 139 | 62 | 41 | 480 | 40.7 | 0.536 | 21 | 0 |

Figure. 5a First Five Rows of Diabetes Dataset



| age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 52 | 1 | 0 | 125 | 212 | 0 | 1 | 168 | 0 | 1.0 | 2 | 2 | 3 | 0 |
| 53 | 1 | 0 | 140 | 203 | 1 | 0 | 155 | 1 | 3.1 | 0 | 0 | 3 | 0 |
| 70 | 1 | 0 | 145 | 174 | 0 | 1 | 125 | 1 | 2.6 | 0 | 0 | 3 | 0 |
| 61 | 1 | 0 | 148 | 203 | 0 | 1 | 161 | 0 | 0.0 | 2 | 1 | 3 | 0 |
| 62 | 0 | 0 | 138 | 294 | 1 | 1 | 106 | 0 | 1.9 | 1 | 3 | 2 | 0 |

Figure. 5b First Five Rows of Heart Disease Dataset



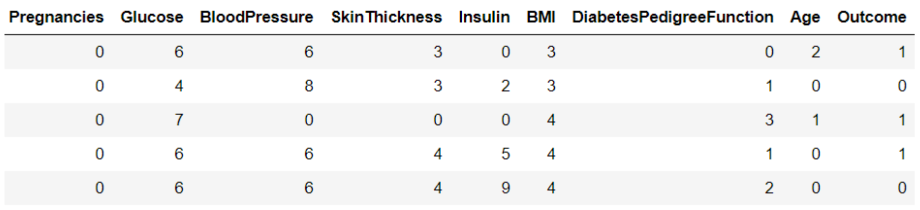
| age | bp | sg | al | su | rbc | pc | ... | pcv | wc | rc | htn | dm | cad | appet | pe | ane | classification |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 48 | 80 | 1.020 | 1 | 0 | 0 | 0 | ... | 44 | 7800 | 5.2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 7 | 50 | 1.020 | 4 | 0 | 0 | 0 | ... | 38 | 6000 | 4.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 62 | 80 | 1.010 | 2 | 3 | 0 | 0 | ... | 31 | 7500 | 4.0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 48 | 70 | 1.005 | 4 | 0 | 0 | 1 | ... | 32 | 6700 | 3.9 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 51 | 80 | 1.010 | 2 | 0 | 0 | 0 | ... | 35 | 7300 | 4.6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure. 5c First Five Rows of Kidney Disease Dataset

After the dataset acquisition, the Data Pre-Processing is done. Data preprocessing is essential in data analytics because it removes unnecessary and noisy data. Unwanted data and missing data are often collected by data sources. The data pre-processing is done to process the missing values. Numerical missing values are filled using statistical measures i.e., mean of each attribute. Categorical missing values are filled using mode value of the classified attribute.

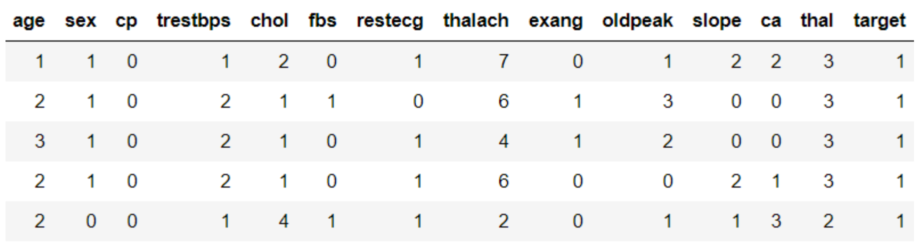
### 5.1.2.2 CONVERSION OF NUMERICAL DATA TO CATEGORICAL DATA

The dataset is converted from numerical to categorical data in order to obtain balanced and normalized dataset. The code snap for the conversion of numerical to categorical data for the diabetes, heart disease, and kidney disease dataset are as picturized in Fig. 5d., 5e., and 5f., severally.



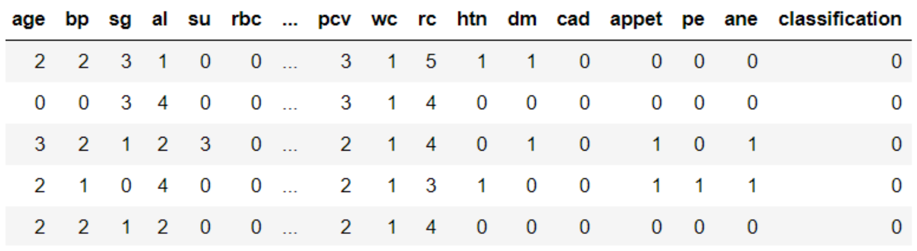
| Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|
| 0 | 6 | 6 | 3 | 0 | 3 | 0 | 2 | 1 |
| 0 | 4 | 8 | 3 | 2 | 3 | 1 | 0 | 0 |
| 0 | 7 | 0 | 0 | 0 | 4 | 3 | 1 | 1 |
| 0 | 6 | 6 | 4 | 5 | 4 | 1 | 0 | 1 |
| 0 | 6 | 6 | 4 | 9 | 4 | 2 | 0 | 0 |

Figure. 5d First Five Rows of Diabetes Dataset after Pre-processing and Conversion



| age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 2 | 0 | 1 | 7 | 0 | 1 | 2 | 2 | 3 | 1 |
| 2 | 1 | 0 | 2 | 1 | 1 | 0 | 6 | 1 | 3 | 0 | 0 | 3 | 1 |
| 3 | 1 | 0 | 2 | 1 | 0 | 1 | 4 | 1 | 2 | 0 | 0 | 3 | 1 |
| 2 | 1 | 0 | 2 | 1 | 0 | 1 | 6 | 0 | 0 | 2 | 1 | 3 | 1 |
| 2 | 0 | 0 | 1 | 4 | 1 | 1 | 2 | 0 | 1 | 1 | 3 | 2 | 1 |

Figure. 5e First Five Rows of Heart Disease Dataset after Pre-processing and Conversion



| age | bp | sg | al | su | rbc | ... | pcv | wc | rc | htn | dm | cad | appet | pe | ane | classification |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 3 | 1 | 0 | 0 | ... | 3 | 1 | 5 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 3 | 4 | 0 | 0 | ... | 3 | 1 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 2 | 1 | 2 | 3 | 0 | ... | 2 | 1 | 4 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 2 | 1 | 0 | 4 | 0 | 0 | ... | 2 | 1 | 3 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 2 | 2 | 1 | 2 | 0 | 0 | ... | 2 | 1 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure. 5f First Five Rows of Kidney Disease Dataset after Pre-processing and Conversion

### 5.2.2 FEATURE SELECTION

It is a technique of identifying the most important features in a dataset. The data source generated data with a large number of attributes. Few of the features in the dataset will not impact the performance of the prediction system. Only necessary features will consider for the Decision-making system, it will reduce the computational complexity. The following algorithms are applied on the pre-processed data to trace out the best features from the dataset.

1. **Chi-Squared**: It is a feature selection algorithm that provides the best features from the dataset. This is applied only for categorical data. It is used to test the independence of two events.
2. **ANOVA:** It stands for ANalysis Of VAriance. It is a feature selection algorithm that helps to gain information about the dependent and independent variables.

Refer Fig. A 2.9. that depicts the feature selection output for all three datasets. The Heart disease dataset features with considerable score value and common to both the scripts results are selected for classification namely: Exang, Cp, Ca, Oldpeak and Thalach. The Diabetes dataset features selected for classification are: Glucose, Age, Pregnancies, Insulin and BMI. The Kidney disease dataset features selected for the classification are: Hemo, sg, htn, dm, al and bgr.

### 5.2.3 PREDICTION SYSTEM

Among these three datasets, only kidney disease dataset had missing values. These are processed with mode values for numerical and categorical attributes respectively. All the numerical features are then converted to categorical. Thus, all the datasets are done with pre-processing and normalization.

With proper pre-processing been done, the datasets are now fit to train the machine learning algorithms. For dimensionality reduction, these pre-processed datasets are applied to Chi-Squared and ANOVA feature selection algorithms. The best attributes that are ideally close to its intrinsic dimensions are obtained with greater scores. Only those attributes with top scores and common to both the feature selection algorithms' results are retained for proceeding with algorithms' performance analysis.

### 5.2.3.1 CLASSIFICATION MODELS

The algorithms applied are 5 base classifiers (SVM, KNN, Decision Tree, Naïve Bayes and Logistic Regression), and 2 Ensemble techniques (Adaboost, Random Forest).

**Support Vector Machine:** SVM is an algorithm which is used mainly as a classifier and also for regression and outlier detection.

**K Nearest Neighbors:** KNN is a supervised machine learning technique that is used for classification. Based on the K nearest neighbor's and the distance between them the new input class will belong to.

**Decision Tree:** It is a supervised machine learning technique which is used for classification problems. The classifier is a tree structure that consist of internal nodes, branch and leaf nodes. The internal nodes represent the features of the dataset, branches represent the decision rules and leaf nodes represent the outcome.

**Naïve-Bayes:** It is a classifier algorithm which uses Bayes theorem to predict the probability of different classes based on various parameters.

**Logistic Regression:** It is a classification algorithm used to predict the probability of the target variable.

**Adaboost:** Adaboost is an ensemble technique. It is used for binary classification and it enhances the performance of any machine learning algorithm.

**Random Forest:** The Random Forest algorithm is a classification algorithm that is similar to decision tree. It adds randomness to the model. It searches for the best features from the random subset of features. A random forest is made up of a large number of independent decision trees that function together as an ensemble. Each tree in the random forest generates a class prediction, and the class that has the highest ballots becomes the prediction of our model. The basic idea behind random forest is a clear but effective one.

Initially, Random forest, SVM, KNN, Decision Tree, Naïve Bayes, Logistic Regression and Adaboost algorithms are trained with the 80% of each disease dataset. After testing the trained model with other 20% of each dataset, all the previously mentioned performance measures are calculated. All the performance measures are calculated and tabulated for all these 7 algorithms for all 3 datasets individually. Final model for disease prediction is recommended, with comparatively best performance measures tabulated. All these stages are represented diagrammatically.

#### 5.2.3.2 PYTHON PICKLING TECHNIQUE FOR RANDOM FOREST ALGORITHM

After the model is built, the model is saved in binary form into a pickle file which can later be loaded while analyzing real-time data for prediction of disease. The pickling code is done for one disease considered as diabetes and similarly, for heart disease and kidney disease models.

#### 5.2.3.3 PICKLE FILE LOADING TO USE THE MODEL

The pickle files are loaded and deployed in the GUI. It is used to predict multiple disease and provide combined results.

After tracing out the best algorithm as Random forest algorithm for each dataset, those algorithms are now trained with the respective entire dataset. These trained models are saved individually as pickle files. Once the user enters the health records, these pickled files are loaded (un-pickling) and the input values are supplied to respective loaded models for prediction. All three models predict the disease with given records of specific fields. Their prediction results are shown in the same module. Interpreting the results, we conclude with using Random Forest Algorithm for proposing the predictive system.

# CHAPTER 6

# RESULTS AND DISCUSSION

For comparing the performance of various algorithms on all 3 datasets, the following measures are calculated as shown in Fig. 6a, Fig. 6b, and Fig. 6c, Refer Fig. 6d, 6e, and 6f, for graphical representation of the performance analysis.

1. Accuracy
2. R-Squared
3. Time of Execution (in Secs)
4. Precision
5. Recall
6. F1-Score

Figure 6a. Performance Analysis of Diabetes Prediction Model

| **ALGORITHMS** | **ACCURACY (%)** | **TIME OF EXECUTION (sec)** | **R-SQUARED (%)** | **PRECISION (%)** | **RECALL (%)** | **F1-SCORE (%)** |
|---|---|---|---|---|---|---|
| **RFA** | 92.3 | 0.37 | 0.64 | 94 | 91 | 92 |
| **SVM** | 79.75 | 0.127 | 0.08 | 78 | 76 | 76 |
| **LR** | 79 | 0.048 | 0.047 | 76 | 74 | 75 |
| **KNN** | 80.5 | 0.134 | -1.19 | 46 | 46 | 46 |
| **NB** | 79 | 0.019 | 0.046 | 77 | 76 | 77 |
| **AB** | 82 | 0.61 | 0.182 | 80 | 80 | 80 |
| **DT** | 80 | 0.012 | -0.046 | 75 | 69 | 72 |

Figure 6b. Performance Analysis of Heart Disease Prediction Model

| **ALGORITHMS** | **ACCURACY (%)** | **TIME OF EXECUTION (sec)** | **R-SQUARED (%)** | **PRECISION (%)** | **RECALL (%)** | **F1-SCORE (%)** |
|---|---|---|---|---|---|---|
| **RFA** | 98.05 | 0.14 | 92 | 98 | 98 | 98 |
| **SVM** | 84.74 | 0.033 | 38 | 85 | 84 | 85 |
| **LR** | 86.36 | 0.036 | 45 | 87 | 86 | 86 |
| **KNN** | 82.47 | 0.087 | 45 | 87 | 86 | 86 |
| **NB** | 83.44 | 0.014 | 33 | 84 | 83 | 83 |
| **AB** | 85.39 | 0.49 | 41 | 86 | 85 | 86 |
| **DT** | 91.56 | 0.012 | 66 | 92 | 92 | 92 |

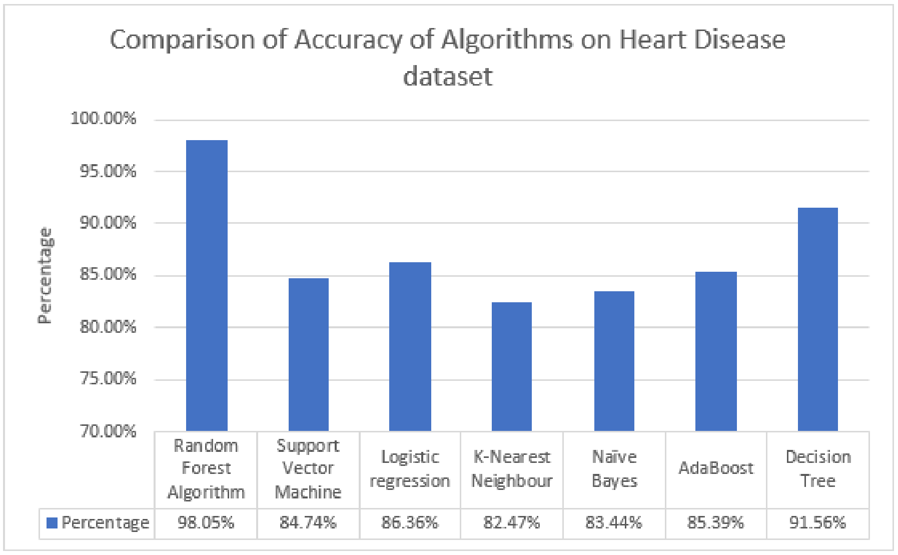
Figure 6c. Performance Analysis of Kidney Disease Prediction Model

| **ALGORITHMS** | **ACCURACY (%)** | **TIME OF EXECUTION (sec)** | **R-SQUARED (%)** | **PRECISION (%)** | **RECALL (%)** | **F1-SCORE (%)** |
|---|---|---|---|---|---|---|
| **RFA** | 99.17 | 0.13 | 0.96 | 100 | 98 | 99 |
| **SVM** | 98.17 | 0.019 | 0.96 | 99 | 99 | 99 |
| **LR** | 98.33 | 0.045 | 0.93 | 98 | 98 | 98 |
| **KNN** | 98.34 | 0.042 | 0.93 | 98 | 98 | 98 |
| **NB** | 91.67 | 0.016 | 0.625 | 90 | 94 | 92 |
| **AB** | 99.17 | 0.54 | 0.96 | 99 | 100 | 99 |
| **DT** | 95 | 0.011 | 0.775 | 94 | 97 | 96 |

## 6.1 COMPARISON OF DIFFERENT PREDICTION SYSTEM FOR MULTIPLE DISEASE

The classification algorithm applied to the best features of the dataset after feature selection is done has produced the results are depicted in Fig. 6d, 6e, and 6f. The Fig. 6d, 6e, and 6f show the results on accuracy, time, R-squared values. The confusion matrix tabulation is illustrated in Fig 6a, Fig. 6b, and Fig. 6c. show the various results obtained by applying the classification algorithm on the 3 different datasets namely heart disease, Kidney disease and diabetes. From these tabulations, we can conclude that Random Forest Algorithm has resulted with best performance measures analysis.

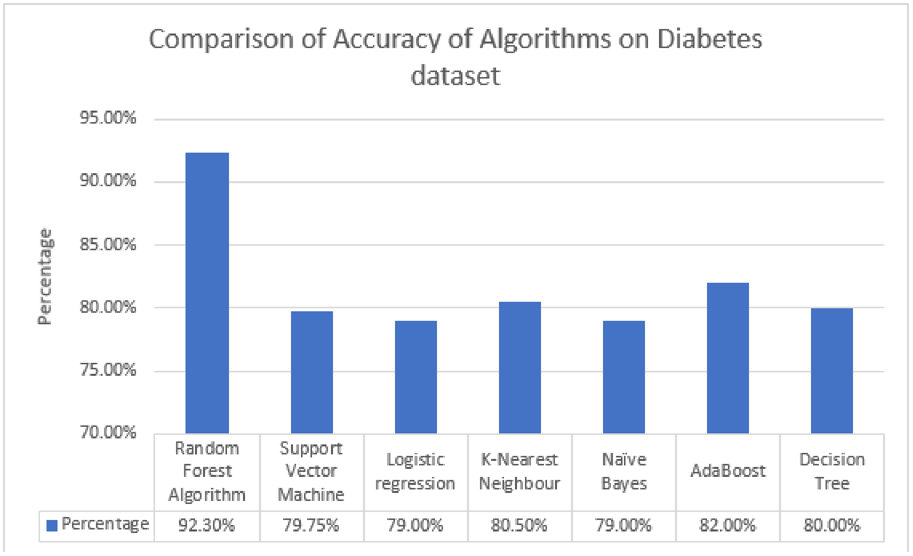
For Heart disease: It has an accuracy of 98.05%, precision as 100%, Recall as 96%, F1-Score as 98%.



| | Random Forest Algorithm | Support Vector Machine | Logistic regression | K-Nearest Neighbour | Naïve Bayes | AdaBoost | Decision Tree |
|---|---|---|---|---|---|---|---|
| ■ Percentage | 98.05% | 84.74% | 86.36% | 82.47% | 83.44% | 85.39% | 91.56% |

Figure 6d. Comparison of Accuracy of Algorithms on Heart Disease Dataset

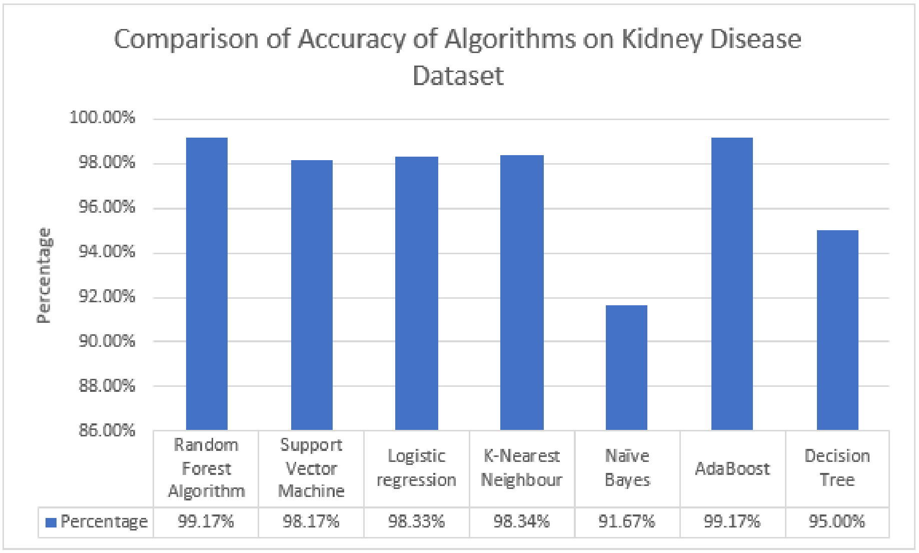
For Diabetes: It has an accuracy of 92.30%, precision as 94%, Recall as 91%, F1-Score as 92%.



| | Random Forest Algorithm | Support Vector Machine | Logistic regression | K-Nearest Neighbour | Naïve Bayes | AdaBoost | Decision Tree |
|---|---|---|---|---|---|---|---|
| ■ Percentage | 92.30% | 79.75% | 79.00% | 80.50% | 79.00% | 82.00% | 80.00% |

Figure 6e. Comparison of Accuracy of Algorithms on Diabetes Dataset

For Kidney Disease: It has an accuracy of 99.17%, precision as 100%, Recall as 99%, F1-Score as 99%.



| | Random Forest Algorithm | Support Vector Machine | Logistic regression | K-Nearest Neighbour | Naïve Bayes | AdaBoost | Decision Tree |
|---|---|---|---|---|---|---|---|
| ■ Percentage | 99.17% | 98.17% | 98.33% | 98.34% | 91.67% | 99.17% | 95.00% |

Figure 6f. Comparison of Accuracy of Algorithms on Kidney Dataset

Hence, Random Forest Algorithm is having highest accuracy among all 3 datasets and thus it is prescribed for building the prediction model. Thereby, all 3 datasets are trained with Random Forest Algorithm and pickled separately for final prediction. In the GUI part, this pickled file will be loaded and used for prediction.

# CHAPTER 8

# CONCLUSION AND FUTURE WORK

Multiple disease prediction model is the system in which the users are allowed to input their health records that they have obtained from various prescribed tests' reports. Disease predictions with this model can be considered as an intermediate step between taking up the prescribed health check-up test and consultation with the medical practitioners. Thereupon, not all the patients will have a need to go for consultation. Compared with the existing system, the model that has been proposed is more flexible and efficient, as the prediction of all three diseases is done with single set of inputs and the users are given a choice, either to predict with the features they are most interested in or with the system recommended features. The benefit of multiple disease prediction model is that it can predict the occurrence probability of various diseases in advance, thereby reducing the mortality ratio. The proposed model has overcome one of the demerits of the existing model, which is prediction can be done with single set of input data with higher flexibility and it is fixed with 3 types of diseases.

In future, the developers can add any type and number of diseases with better scalability on datasets. If the dataset, which is going to be collected in the future, has records of patients from various birth places all over the world, then its trained model will be more efficient than the proposed one. Moreover, the larger the dataset, more will be the benefit to be gained. As a result of that, the model may be recommended to more users irrespective of geographical locations and its conditions.

# APPENDIX-1

## CODING

### DIABETES DATASET PRE-PROCESSING CODE

```
import numpy as np
import pandas as pd
df=pd.read_excel('Diabetes1.xlsx')
df[:5]
min(df['Pregnancies']),max(df['Pregnancies'])
d=df['Pregnancies']
df['Pregnancies']=np.where(d<3,0,np.where(d<6,1,np.where(d<9,2,np.where(d<12,3,np.where(d<1
5,4,5)))))
df[:5]
op=pd.ExcelWriter('Diab_Categ.xlsx')
df.to_excel(op)
op.save()
import pandas as pd
import numpy as np
df=pd.read_excel('Diab_Categ.xlsx')
for a in df.columns:
    print(df[a].value_counts())
```

### HEART DATASET PRE-PROCESSING CODE

```
import pandas as pd
import numpy as np
import pandas as pd
import numpy as np
df=pd.read_excel('heart.xlsx')
df[:5]
c=df['chol']
df['chol']=np.where(c<166,0,np.where(c<206,1,np.where(c<246,2,np.where(c<286,3,np.where(c<3
26,4,np.where(c<366,5,np.where(c<406,6,np.where(c<446,7,np.where(c<486,8,np.where(c<526,9,
10))))))))))
df['chol'].value_counts()
df=pd.read_excel('Heart_Categ.xlsx')
```

### KIDNEY DISEASE DATASET PRE-PROCESSING CODE

```
import pandas as pd
import numpy as np
import pandas as pd
import numpy as np
df=pd.read_excel('kidney.xlsx')
df[:5]
c=df['s=na']
df['na']=np.where(c<166,0,np.where(c<206,1,np.where(c<246,2,np.where(c<286,3,np.where(c<326
,4,np.where(c<366,5,np.where(c<406,6,np.where(c<446,7,np.where(c<486,8,np.where(c<526,9,10)
)))))))))
```

```
df['na'].value_counts()
df=pd.read_excel('Kidney_Categ.xlsx')
```

**FEATURE SELECTION**

```
import pickle as pk
import pandas as pd
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
data = pd.read_excel("Heart_Categ.xlsx")
X = data.iloc[:,0:13]
Y = data.iloc[:,-1]

features = SelectKBest(score_func=chi2, k=13)
fit = features.fit(X,Y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)

featureScores1 = pd.concat([dfcolumns,dfscores],axis=1)
featureScores1.columns = ['Features','Score']
print(featureScores1.nlargest(13,'Score'))
pk.dump(fit,open('HC.pickle','wb'))

import pandas as pd
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
data = pd.read_excel("Diab_Categ.xlsx")
X = data.iloc[:,0:9]
Y = data.iloc[:,-1]

features = SelectKBest(score_func=chi2, k=9)
fit = features.fit(X,Y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)

featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Features','Score']
print(featureScores.nlargest(9,'Score'))

plt.figure(1, figsize=(14, 5))
plt.title("Feature importance")
plt.bar(featureScores['Features'],featureScores['Score'])
plt.xlabel('Features')
plt.ylabel('Scores')
plt.show()

from numpy import set_printoptions
from sklearn.feature_selection import f_classif

features = SelectKBest(score_func=f_classif, k=9)
fit = features.fit(X, Y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
```

```
featureScores.columns = ['Features','Score']
print(featureScores.nlargest(9,'Score'))

plt.figure(1, figsize=(14, 5))
plt.title("Feature importance")
plt.bar(featureScores['Features'],featureScores['Score'])
plt.xlabel('Features')
plt.ylabel('Scores')
plt.show()

import pandas as pd
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
data = pd.read_excel("Kid_Categ.xlsx")
X = data.iloc[:,1:25]
Y = data.iloc[:,-1]

features = SelectKBest(score_func=chi2, k=24)
fit = features.fit(X,Y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)

featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Features','Score']
print(featureScores.nlargest(24,'Score'))

plt.figure(1, figsize=(14, 5))
plt.title("Feature importance")
plt.bar(featureScores['Features'],featureScores['Score'])
plt.xlabel('Features')
plt.ylabel('Scores')
plt.show()

from numpy import set_printoptions
from sklearn.feature_selection import f_classif

features = SelectKBest(score_func=f_classif, k=24)
fit = features.fit(X, Y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Features','Score']
print(featureScores.nlargest(24,'Score'))

plt.figure(1, figsize=(14, 5))
plt.title("Feature importance")
plt.bar(featureScores['Features'],featureScores['Score'])
plt.xlabel('Features')
plt.ylabel('Scores')
plt.show()
```

**CREATING PICKLING FILES**

```
import pickle
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier

d1=pd.read_excel('Diab_Categ.xlsx')
d2=pd.read_excel('Heart_Categ.xlsx')
d3=pd.read_excel('Kid_Categ.xlsx')

#Diabetes
X=d1[['Glucose','Age','Pregnancies','BMI','Insulin']]
Y = d1.iloc[:,-1]
clf=RandomForestClassifier(n_estimators=10)
clf.fit(X,Y)
pickle.dump(clf, open ('DiabPred.pickle', 'wb'))

# Heart
X=d2[['oldpeak','ca','cp','exang','thalach']]
Y = d2.iloc[:,-1]
clf=RandomForestClassifier(n_estimators=47)
clf.fit(X,Y)
pickle.dump(clf, open ('HeartPred.pickle', 'wb'))

# Kidney
X=d3[['Hemo', 'sg', 'htn', 'dm', 'al', 'bgr']]
Y = d3.iloc[:,-1]
clf=RandomForestClassifier(n_estimators=47)
clf.fit(X,Y)
pickle.dump(clf, open ('KidPred.pickle', 'wb'))
```

**DIABETES CODE**

```
import pandas as pd
data=pd.read_excel('Diab_Categ.xlsx')
data[:5]

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sn
from sklearn.metrics import r2_score
from sklearn import metrics
import time
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
```

```
X=data.iloc[:,1:9]
Y=data.iloc[:,-1]
X=data[['Glucose','Age','Pregnancies','BMI','Insulin']]
Y=data.iloc[:,-1]

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
import time
s=time.time()
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,random_state=50)
clf=RandomForestClassifier(n_estimators=10)
clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print('Time = ',time.time()-s)
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sn

cm = confusion_matrix(y_test, y_pred)
conf_matrix = pd.DataFrame(data = cm,  columns = ['FN', 'FP'],  index =['TN', 'TP'])
plt.figure(figsize = (8, 5))
sn.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Greens")
plt.show()
print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))

from sklearn import svm
import time
s=time.time()
x_train, x_test, y_train, y_test = train_test_split(X,Y, test_size=0.2,random_state=10)
ml = svm.SVC(kernel='linear')
ml.fit(x_train, y_train)
y_pred = ml.predict(x_test)
print("Accuracy = ",ml.score(x_test,y_test))
print('Time = ',time.time()-s)
print('R2 = ',r2_score(y_test, y_pred))
print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))

from sklearn.linear_model import LogisticRegression
from sklearn import metrics
s=time.time()
lr = LogisticRegression()
x_train, x_test, y_train, y_test = train_test_split(X,Y, test_size=0.2,random_state=10)
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)
print('Accuracy : ',metrics.accuracy_score(y_test, y_pred))
print('Time : ',time.time()-s)
print('R2 = ',r2_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
conf_matrix = pd.DataFrame(data = cm,  columns = ['FN', 'FP'],  index =['TN', 'TP'])
plt.figure(figsize = (8, 5))
sn.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Greens")
plt.show()

print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))

print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))

from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,random_state=10)
s=time.time()
classifier = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1), n_estimators=200)
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
print('Accuracy : ',metrics.accuracy_score(y_test, y_pred))
print('Test : ',time.time()-s)
print('R2 = ',r2_score(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
conf_matrix = pd.DataFrame(data = cm,  columns = ['FN', 'FP'],  index =['TN', 'TP'])
plt.figure(figsize = (8, 5))
sn.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Greens")
plt.show()

print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))

from sklearn.tree import DecisionTreeClassifier
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,random_state=100)
s=time.time()
dtree_model = DecisionTreeClassifier(max_depth = 2).fit(x_train, y_train)
y_pred = dtree_model.predict(x_test)
print('Accuracy : ',metrics.accuracy_score(y_test, y_pred))
print('Test : ',time.time()-s)
print('R2 = ',r2_score(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
conf_matrix = pd.DataFrame(data = cm,  columns = ['FN', 'FP'],  index =['TN', 'TP'])
plt.figure(figsize = (8, 5))
sn.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Greens")
plt.show()
m1 = LogisticRegression(penalty = 'l2', random_state = 20)
estimators.append(('logistic1', m1))
m2 = LogisticRegression(penalty = 'l2', random_state = 10)
estimators.append(('logistic2', m2))
m3 = LogisticRegression(penalty = 'l2', random_state = 50)
```

```
estimators.append(('logistic3', m3))
m4 = LogisticRegression(penalty = 'l2', random_state = 100)
estimators.append(('logistic4', m4))
m5 = LogisticRegression(penalty = 'l2', random_state = 5)
estimators.append(('logistic5', m5))

m6 = svm.SVC(kernel='linear')
estimators.append(('svm1', m6))
m7 = GaussianNB()
estimators.append(('NB1', m7))
ensemble = VotingClassifier(estimators)
ensemble.fit(x_train, y_train)
y_pred = ensemble.predict(x_test)
kfold = model_selection.KFold(n_splits=10, random_state=1)
results = model_selection.cross_val_score(ensemble, x_train, y_train, cv=kfold)
print('Accuracy : ',results.mean())
print('Time : ',time.time()-s)
print('R2 = ',r2_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
conf_matrix = pd.DataFrame(data = cm,  columns = ['FN', 'FP'], index =['TN', 'TP'])
plt.figure(figsize = (8, 5))
sn.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Greens")
plt.show()

print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))

x_train,x_test,y_train, y_test = train_test_split(X,Y,test_size = 0.2 , random_state = 5 )
s=time.time()
classifier = Sequential()
classifier.add(Dense(activation = "relu", input_dim = 8,
                units = 8, kernel_initializer = "uniform"))
classifier.add(Dense(activation = "relu", units = 10,
                kernel_initializer = "uniform"))
classifier.add(Dense(activation = "sigmoid", units = 1,
                kernel_initializer = "uniform"))
classifier.compile(optimizer = 'adam' , loss = 'binary_crossentropy',
               metrics = ['accuracy'] )
classifier.fit(x_train , y_train , batch_size = 10 ,epochs = 10 )
y_pred = classifier.predict(x_test)
y_pred = (y_pred > 0.5)
end=time.time()
print('Time = ',end-s)
print('R2 = ',r2_score(y_test, y_pred))
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

**HEART DISEASE CODE**

```
import pandas as pd
data = pd.read_excel("Heart_Categ.xlsx")
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
```

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,random_state=46)
clf=RandomForestClassifier(n_estimators=47)
clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
end=time.time()
print('Time = ',end-start)
print('R2 = ',r2_score(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
conf_matrix = pd.DataFrame(data = cm, columns = ['FN', 'FP'],  index =['TN', 'TP'])
plt.figure(figsize = (8, 5))
sn.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Greens")
plt.show()
print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))

x_train, x_test, y_train, y_test = train_test_split(X,Y, test_size=0.3,random_state=100)
start=time.time()
ml = svm.SVC(kernel='linear')
ml.fit(x_train, y_train)
y_pred = ml.predict(x_test)
print(ml.score(x_test,y_test))
print('Time = ',time.time()-start)
print('R2 = ',r2_score(y_test, y_pred))

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,random_state=100)
estimators = []
s=time.time()
m1 = svm.SVC(kernel='linear')
estimators.append(('svm', m1))
m2 = GaussianNB()
estimators.append(('NB', m2))
m3 =  KNeighborsClassifier(n_neighbors = 2)
estimators.append(('knn1', m3))

ensemble = VotingClassifier(estimators)
ensemble.fit(x_train, y_train)
y_pred = ensemble.predict(x_test)
print('Accuracy : ',metrics.accuracy_score(y_test, y_pred))
print('Time : ',time.time()-s)
print('R2 = ',r2_score(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
conf_matrix = pd.DataFrame(data = cm,  columns = ['FN', 'FP'],  index =['TN', 'TP'])
plt.figure(figsize = (8, 5))
sn.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Greens")
plt.show()
print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))

x_train,x_test,y_train, y_test = train_test_split(X,Y,test_size = 0.3 , random_state = 10 )
s=time.time()
```

```
classifier = Sequential()
classifier.add(Dense(activation = "relu", input_dim = 5,
                    units = 8, kernel_initializer = "uniform"))
classifier.fit(x_train , y_train , batch_size = 8  ,epochs = 5 )
y_pred = classifier.predict(x_test)
y_pred = (y_pred > 0.5)
end=time.time()
print('Time = ',end-s)
print('R2 = ',r2_score(y_test, y_pred))
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

conf_matrix = pd.DataFrame(data = cm,  columns = ['FN', 'FP'],  index =['TN', 'TP'])
plt.figure(figsize = (8, 5))
sn.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Greens")
plt.show()
print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))
```

**KIDNEY DISEASE CODE**

```
import pandas as pd
data=pd.read_excel('Kid_Categ.xlsx')
data[:5]

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from sklearn import metrics
import time
from sklearn import metrics
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3)
clf=RandomForestClassifier(n_estimators=50)
clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print('Time = ',time.time()-s)
print('R2 = ',r2_score(y_test, y_pred))

from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sn

cm = confusion_matrix(y_test, y_pred)
conf_matrix = pd.DataFrame(data = cm,
                          columns = ['FN', 'FP'],
                          index =['TN', 'TP'])
plt.figure(figsize = (8, 5))
sn.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Greens")
plt.show()

print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))
```

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
s=time.time()
lr = LogisticRegression()

x_train, x_test, y_train, y_test = train_test_split(X,Y, test_size=0.3,random_state=100)
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)

print('Accuracy : ',metrics.accuracy_score(y_test, y_pred))
print('Time : ',time.time()-s)
print('R2 = ',r2_score(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
conf_matrix = pd.DataFrame(data = cm,
                        columns = ['FN', 'FP'],
                        index =['TN', 'TP'])
plt.figure(figsize = (8, 5))
sn.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Greens")
plt.show()

print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))

classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
print('Accuracy : ',metrics.accuracy_score(y_test, y_pred))
print('Test : ',time.time()-s)
print('R2 = ',r2_score(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
conf_matrix = pd.DataFrame(data = cm,
                        columns = ['FN', 'FP'],
                        index =['TN', 'TP'])
plt.figure(figsize = (8, 5))
sn.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Greens")
plt.show()

print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))
```

**GRAPHICAL USER INTERFACE (GUI)**

```
import pickle as p
import pandas as pd
import numpy as np
from tkinter import *
import tkinter as tk
gui = Tk()
gui['background']='#85fff8'
gui.geometry('1100x700')
head = Label(gui, text = "Multi Disease Prediction", bg='#85ffff',font=("Serif", 32, 'bold')).place(x
= 230,y = 40)
```

```
#Prediction
def Pred():
    win = Tk()
    win.geometry('700x500') # set the root dimensions
    win['background']='#85fff0'

    diab=p.load(open('DiabPred.pickle','rb'))
    #Column 1
    Label(win,text="Pregnancies",bg='#85fff0').place(x=10,y=30)
    P_entrybox=ttk.Entry(win,width=16,textvariable=tk.IntVar())
    P_entrybox.place(x=100,y=30)

    heart=p.load(open('HeartPred.pickle','rb'))
    #Column 1
    Label(win,text="Exang",bg='#85fff0').place(x=230,y=30)
    lbh1=Listbox(win,width=7, height=2)
    lbh1.insert(1,'No')
    lbh1.insert(2,'Yes')
    lbh1.place(x=280,y=30)
    h1=0
    def selecth1():
        global h1
        c=lbh1.curselection()[0]
        if c==1: h1=1
    btn = Button(win, text = "select",command=selecth1).place(x=350,y=30)

    kid=p.load(open('KidPred.pickle','rb'))
    #Column 1
    Label(win,text="Hemoglobin",bg='#85fff0').place(x=450,y=30)
    k1=tk.IntVar()
    k1_entrybox=ttk.Entry(win,width=16,textvariable=k1)
    k1_entrybox.place(x=550,y=30)

    def Submit():
        df=pd.DataFrame()
        d=int(P_entrybox.get())
        p=np.where(d<3,0,np.where(d<6,1,np.where(d<9,2,np.where(d<12,3,np.where(d<15,4,5)))))
        d=int(A_entrybox.get())

a=np.where(d<30,0,np.where(d<40,1,np.where(d<50,2,np.where(d<60,3,np.where(d<70,4,5)))))
        d=int(G_entrybox.get())

g=np.where(d<20,0,np.where(d<40,1,np.where(d<60,2,np.where(d<80,3,np.where(d<100,4,np.whe
re(d<120,5,np.where(d<140,6,np.where(d<160,7,np.where(d<180,8,9)))))))))
        d=int(B_entrybox.get())
        h3=int(h3_entrybox.get())
        h4=int(h4_entrybox.get())
        t=int(h5_entrybox.get())

h5=np.where(t<84,0,np.where(t<97,1,np.where(t<110,2,np.where(t<123,3,np.where(t<136,4,np.wh
ere(t<149,5,np.where(t<162,6,np.where(t<175,7,np.where(t<188,8,9)))))))))

        d=int(k1_entrybox.get())
        k1=np.where(d<6,0,np.where(d<9,1,np.where(d<12,2,np.where(d<15,3,4))))
        d=int(k2_entrybox.get())

k2=np.where(d<=1.005,0,np.where(d<=1.010,1,np.where(d<=1.015,2,np.where(d<=1.020,3,4))))
        d=int(k6_entrybox.get())

k6=np.where(d<70,0,np.where(d<140,1,np.where(d<210,2,np.where(d<280,3,np.where(d<350,4,np
.where(d<420,5,6))))))

        d1={"Pregnancies":[p],"Age":[a],"Glucose":[g],"BMI":[b],"Insulin":[i]}
        d2={'exang':[h1],'cp':[h2],'ca':[h3],'oldpeak':[h4],'thalach':[h5]}
        d3={'hemo':[k1],'sg':[k2],'htn':[k3],'dm':[k4],'al':[k5],'bgr':[k6]}
        df1=pd.DataFrame(d1)
        df2=pd.DataFrame(d2)
        df3=pd.DataFrame(d3)

    def Clear():
        P_entrybox.delete(0, 'end')
        I_entrybox.delete(0, 'end')
        h3_entrybox.delete(0, 'end')
        k1_entrybox.delete(0, 'end')

    DB=ttk.Button(win,text="Submit",command=Submit).place(x=300,y=300)
    clr=ttk.Button(win,text="Clear",command=Clear).place(x=300,y=320)

lb=Listbox(gui,width=20, height=3)
lb.insert(1,'Heart Disease')
lb.insert(2,'Diabetes')
lb.insert(3,'Kidney Disease')
lb.place(x=100,y=150)
d=0
def View():
    global d
    d=lb.curselection()[0]
    if d==0:
        I_H()
    elif d==1:
        I_D()
    else: I_K()

Button(gui, text = " View ",command=View).place(x=100,y=200)

#Algorithms
Label(gui, text = "Algorithms", bg='#85ffff',font=("Serif", 20)).place(x = 740,y = 100)

def lr():
    global algo
    if algo[0]==0: algo[0]=1
    else: algo[0]=0
    print(algo)
def nb():
    global algo
    if algo[2]==0: algo[2]=1
    else: algo[2]=0
    print(algo)
def svm():
    global algo
    if algo[3]==0: algo[3]=1
    else: algo[3]=0
    print(algo)
def rf():
    global algo
    if algo[5]==0: algo[5]=1
    else: algo[5]=0
    print(algo)
def ab():
    global algo
    if algo[6]==0: algo[6]=1
    else: algo[6]=0
    print(algo)
algo=[0]*6
al_names=["Logistic Regression","Decision Tree","NB","Support Vector Machine","Nearest
Neighbour","Random Forest","Adaboost"]
Checkbutton(gui, text = "Random Forest",activebackground="black" ,font=("Serif", 12),
activeforeground="white",bg="#85ffff",bd=5,variable = IntVar(),onvalue = 1, offvalue =
0,command = rf).place(x=740,y=300)
Checkbutton(gui, text = "Adaboost",activebackground="black" ,font=("Serif", 12),
activeforeground="white",bg="#85ffff",bd=5,variable = IntVar(),onvalue = 1, offvalue =
0,command = ab).place(x=740,y=330)

def Submit():
    fs=Tk()
    fs['background']='black'
    fs.geometry('1500x1500')
    fs.pack_propagate(True)
    #Label(vis, text = "Features Selected", bg='black',font=("Serif", 20),fg='white').place(x = 50,y =
50)

 d={0:'Heart disease',1:'Diabetes',2:'Kidney disease'}
    for i in range(3):

        if dataset_selection[i]==1:
            if i==0:
                data = pd.read_excel("Heart_Categ.xlsx")
                X = data.iloc[:,0:13]
                Y = data.iloc[:,-1]
            else:
                data = pd.read_excel("kid_Categ.xlsx")
                X = data.iloc[:,1:25]
                Y = data.iloc[:,-1]
                key=24
            if feature_selection[0]==1:
                features = SelectKBest(score_func=chi2, k=key)
                fit = features.fit(X,Y)
            if cnt==1:
                a,b=0,400
            elif cnt==2:
                a,b=450,0
            elif cnt==3:
                a,b=450,400
            elif cnt==4:
                a,b=900,0
            elif cnt==5:
                a,b=900,400

    fea.mainloop()
    fs.mainloop()
Button(gui,text=' Submit ', font=("Serif", 12),command=Submit).place(x=500,y=400)
gui.mainloop()
```

# APPENDIX-2

## SCREENSHOTS

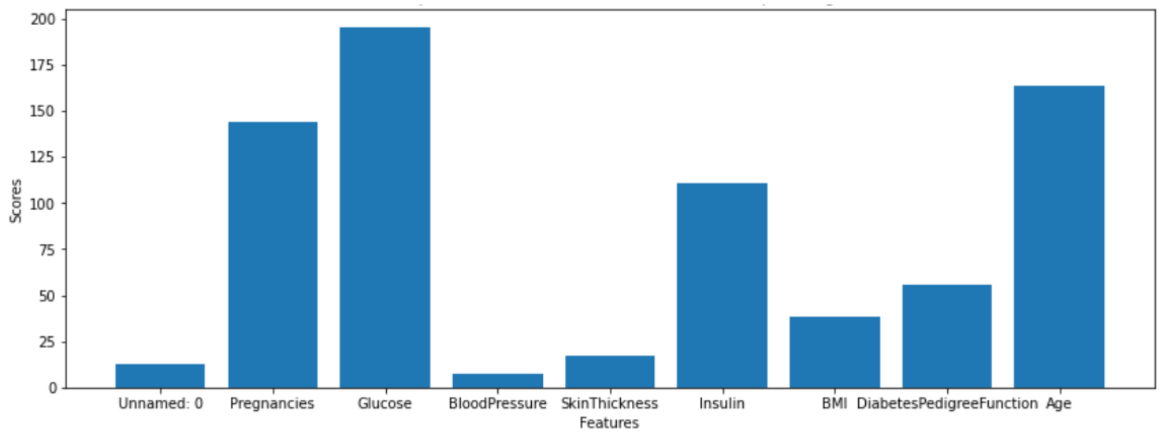


Figure. A 2.1 Feature Importance in Diabetes Dataset With Chi-Square Algorithm

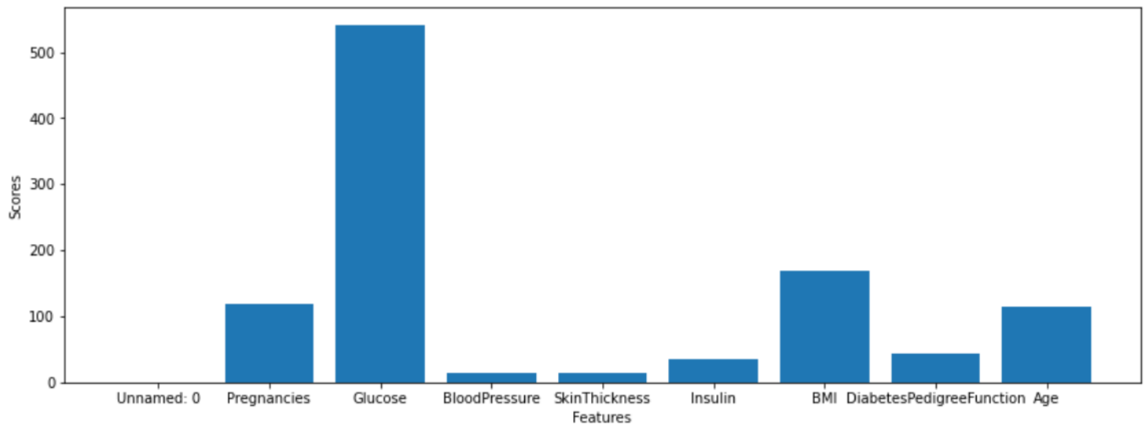


Figure. A 2.2 Feature Importance in Diabetes Dataset With ANOVA Algorithm

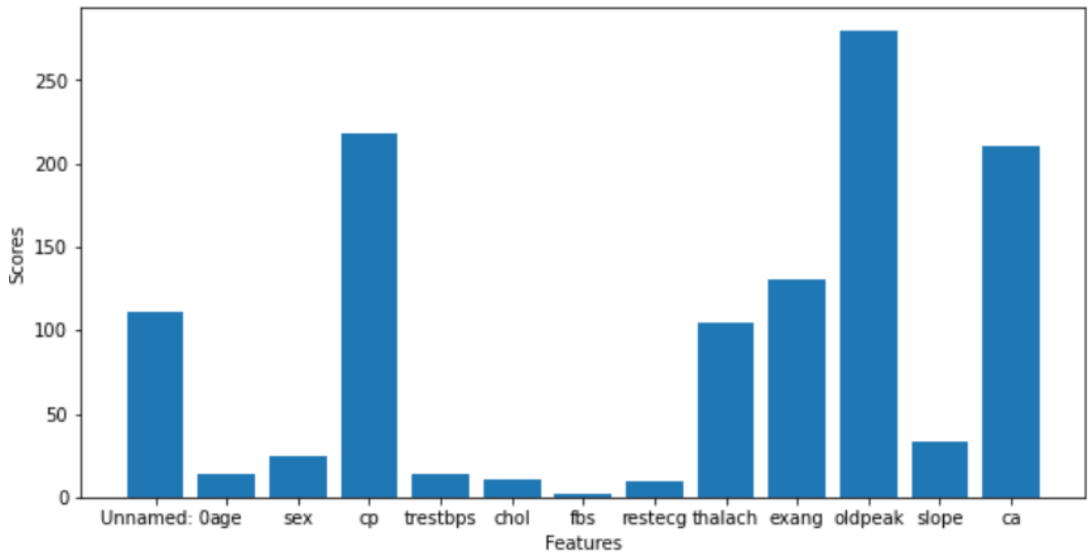


Figure. A 2.3 Feature Importance in Heart Disease Dataset With Chi-Square Algorithm

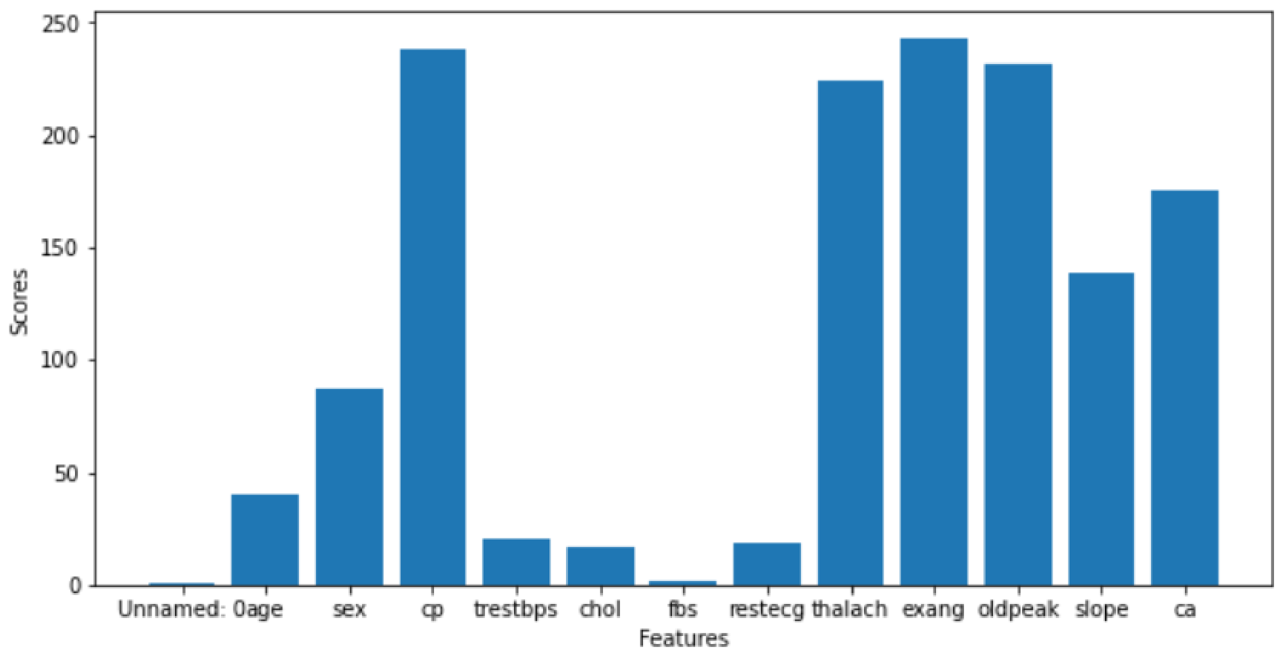


Figure. A 2.4 Feature Importance in Heart Disease Dataset With ANOVA Algorithm

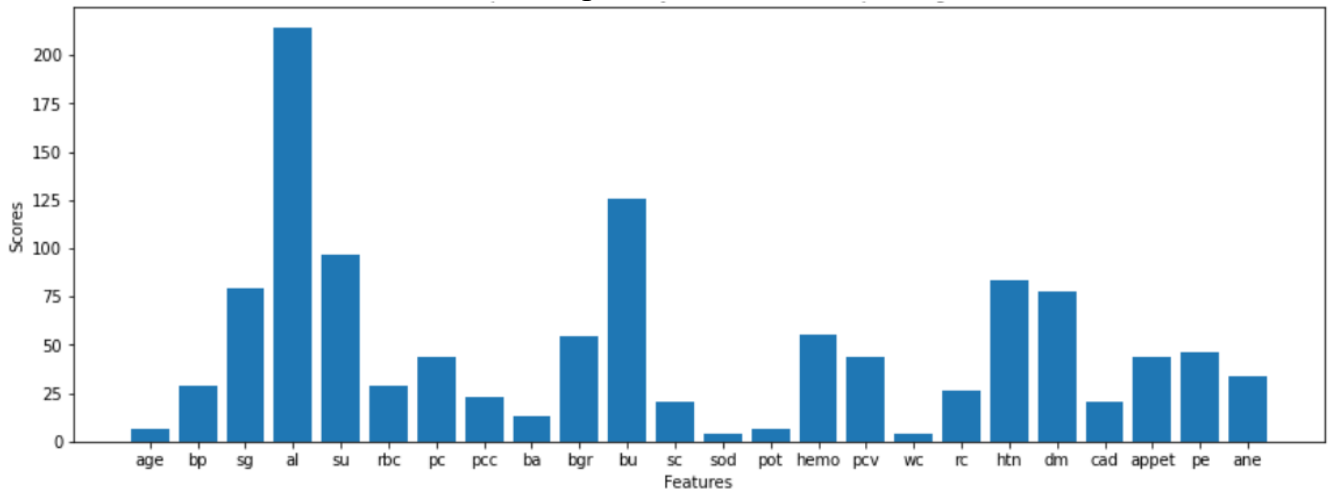


Figure. A 2.5 Feature Importance in Kidney Disease Dataset With Chi-Square Algorithm

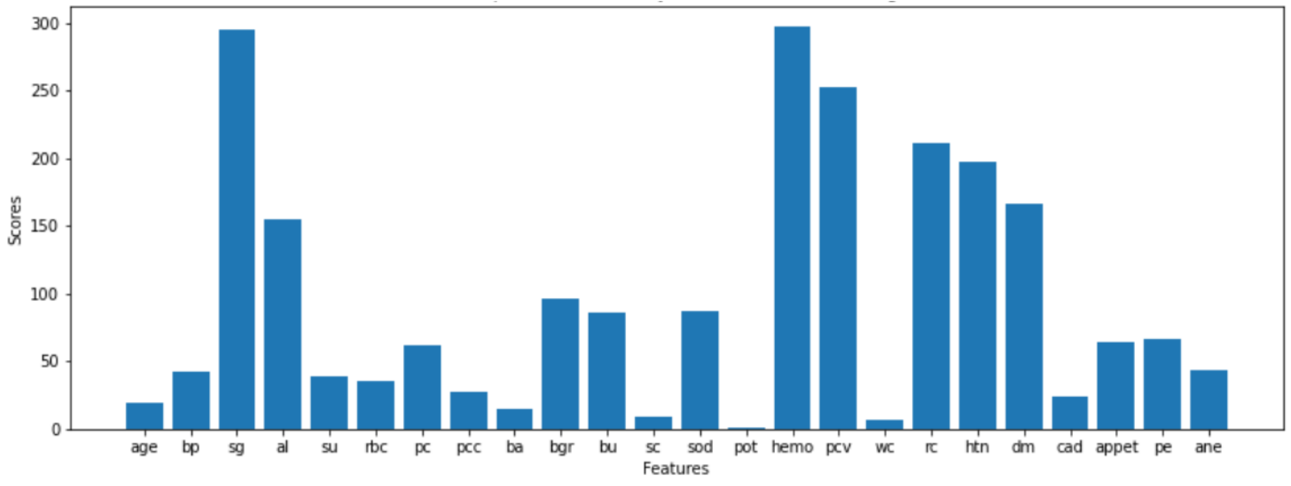


Figure. A 2.6 Feature Importance in Kidney Disease Dataset With ANOVA Algorithm

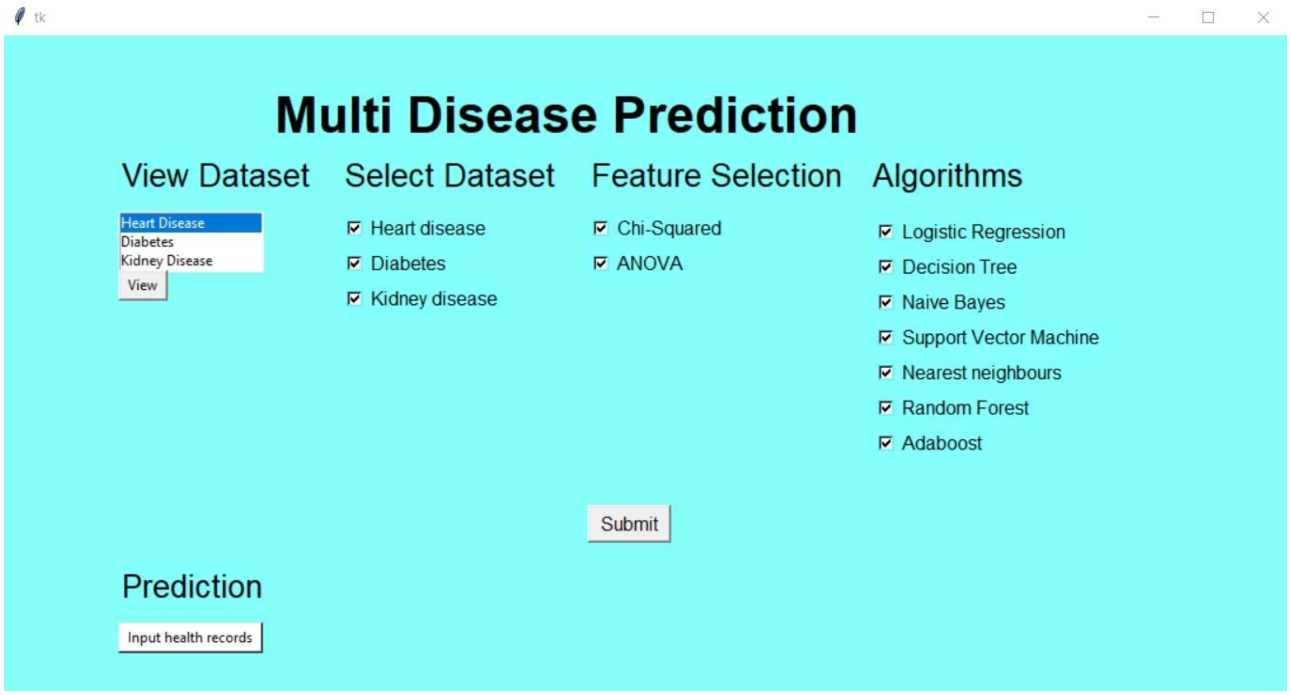
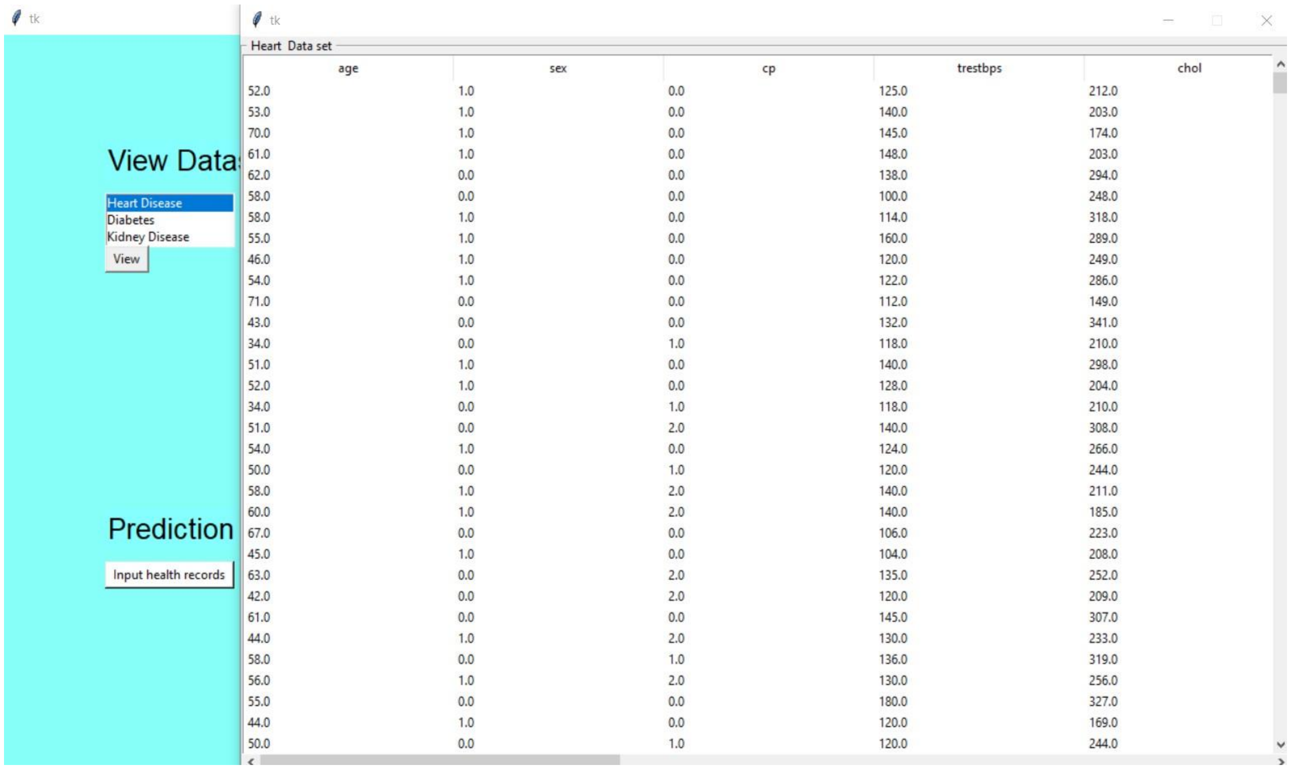


Figure. A 2.7 Home Page GUI of Multiple Disease Prediction System



Heart Data set

| age | sex | cp | trestbps | chol |
|---|---|---|---|---|
| 52.0 | 1.0 | 0.0 | 125.0 | 212.0 |
| 53.0 | 1.0 | 0.0 | 140.0 | 203.0 |
| 70.0 | 1.0 | 0.0 | 145.0 | 174.0 |
| 61.0 | 1.0 | 0.0 | 148.0 | 203.0 |
| 62.0 | 0.0 | 0.0 | 138.0 | 294.0 |
| 58.0 | 0.0 | 0.0 | 100.0 | 248.0 |
| 58.0 | 1.0 | 0.0 | 114.0 | 318.0 |
| 55.0 | 1.0 | 0.0 | 160.0 | 289.0 |
| 46.0 | 1.0 | 0.0 | 120.0 | 249.0 |
| 54.0 | 1.0 | 0.0 | 122.0 | 286.0 |
| 71.0 | 0.0 | 0.0 | 112.0 | 149.0 |
| 43.0 | 0.0 | 0.0 | 132.0 | 341.0 |
| 34.0 | 0.0 | 1.0 | 118.0 | 210.0 |
| 51.0 | 1.0 | 0.0 | 140.0 | 298.0 |
| 52.0 | 1.0 | 0.0 | 128.0 | 204.0 |
| 34.0 | 0.0 | 1.0 | 118.0 | 210.0 |
| 51.0 | 0.0 | 2.0 | 140.0 | 308.0 |
| 54.0 | 1.0 | 0.0 | 124.0 | 266.0 |
| 50.0 | 0.0 | 1.0 | 120.0 | 244.0 |
| 58.0 | 1.0 | 2.0 | 140.0 | 211.0 |
| 60.0 | 1.0 | 2.0 | 140.0 | 185.0 |
| 67.0 | 0.0 | 0.0 | 106.0 | 223.0 |
| 45.0 | 1.0 | 0.0 | 104.0 | 208.0 |
| 63.0 | 0.0 | 2.0 | 135.0 | 252.0 |
| 42.0 | 0.0 | 2.0 | 120.0 | 209.0 |
| 61.0 | 0.0 | 0.0 | 145.0 | 307.0 |
| 44.0 | 1.0 | 2.0 | 130.0 | 233.0 |
| 58.0 | 0.0 | 1.0 | 136.0 | 319.0 |
| 56.0 | 1.0 | 2.0 | 130.0 | 256.0 |
| 55.0 | 0.0 | 0.0 | 180.0 | 327.0 |
| 44.0 | 1.0 | 0.0 | 120.0 | 169.0 |
| 50.0 | 0.0 | 1.0 | 120.0 | 244.0 |

Figure. A 2.8 Viewing the requested Heart Dataset Values (after selecting Heart dataset option and clicking view button)

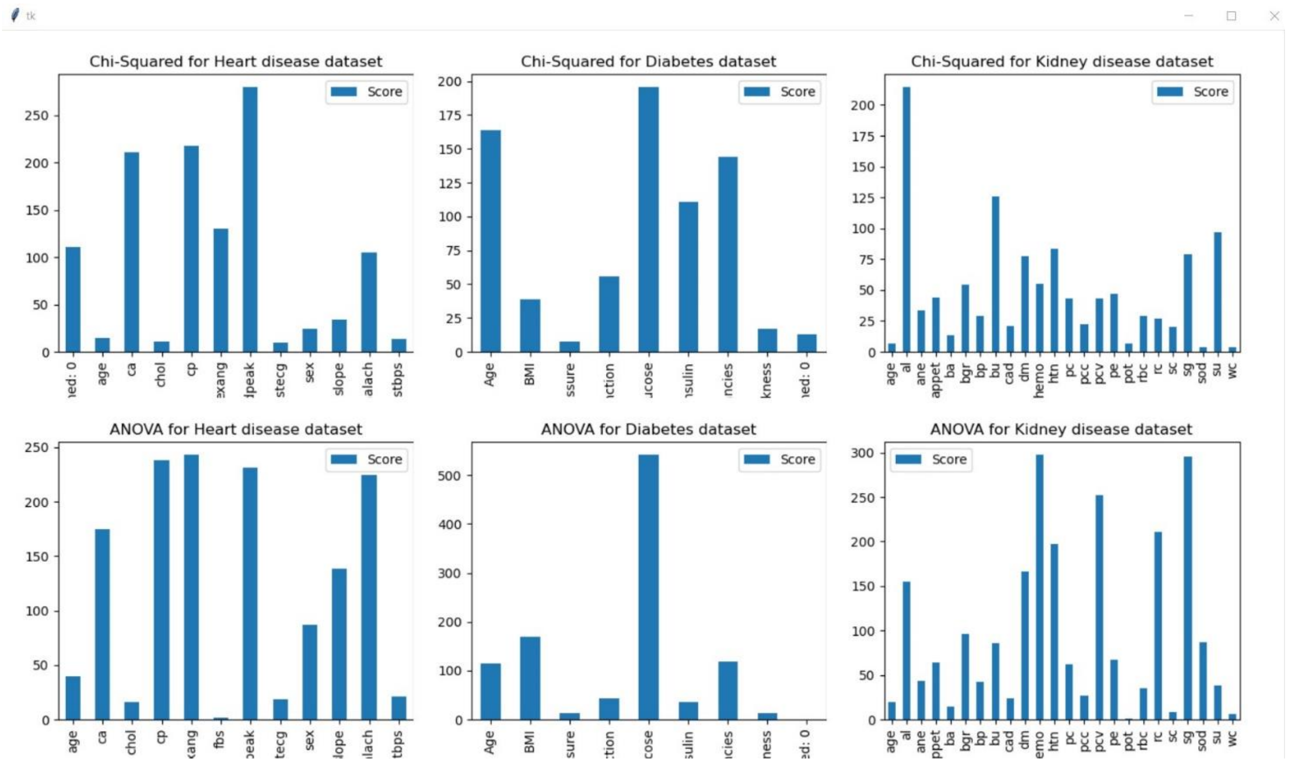
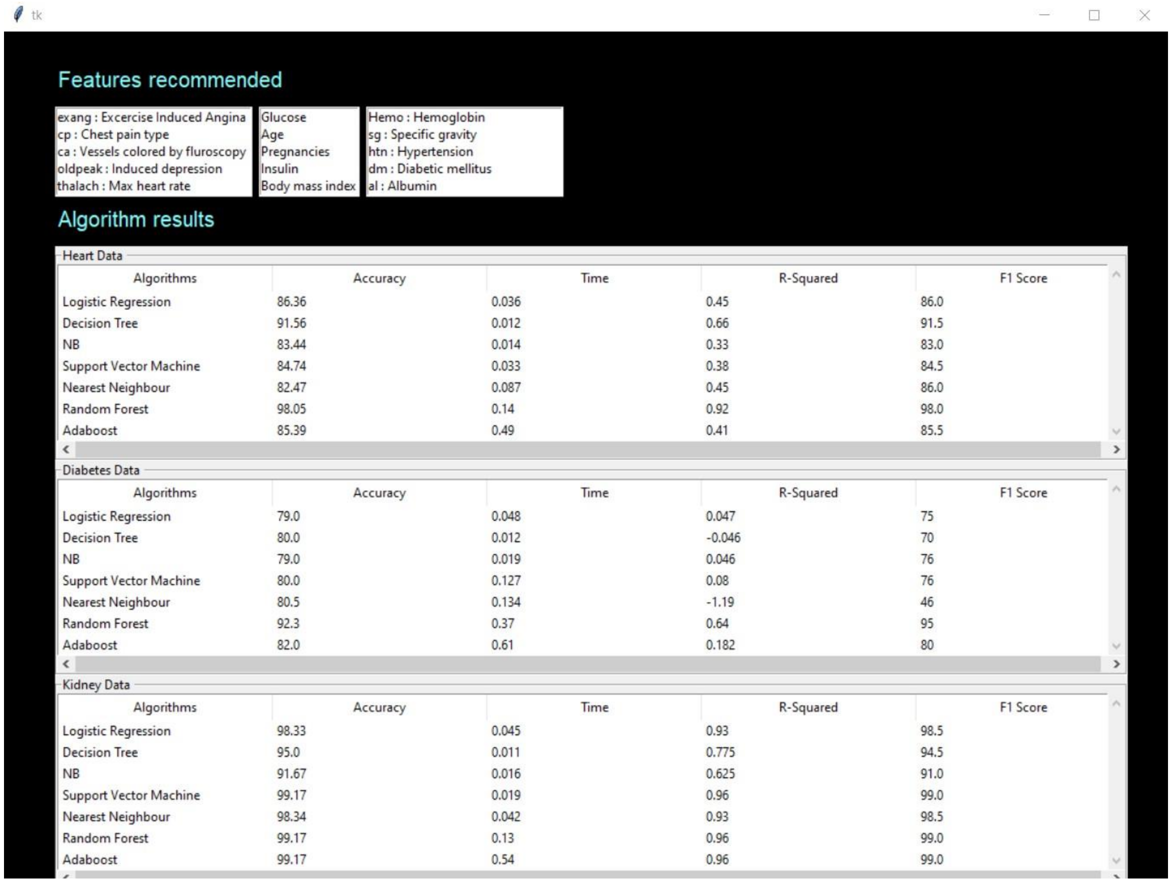


Figure. A 2.9 Important Features selected in all 3 Datasets using Chi-Square and ANOVA Algorithms (after checking the necessary boxes and clicking submit)



Features recommended

| | | |
|---|---|---|
| exang : Excercise Induced Angina | Glucose | Hemo : Hemoglobin |
| cp : Chest pain type | Age | sg : Specific gravity |
| ca : Vessels colored by fluroscopy | Pregnancies | htn : Hypertension |
| oldpeak : Induced depression | Insulin | dm : Diabetic mellitus |
| thalach : Max heart rate | Body mass index | al : Albumin |

Algorithm results

Heart Data

| Algorithms | Accuracy | Time | R-Squared | F1 Score |
|---|---|---|---|---|
| Logistic Regression | 86.36 | 0.036 | 0.45 | 86.0 |
| Decision Tree | 91.56 | 0.012 | 0.66 | 91.5 |
| NB | 83.44 | 0.014 | 0.33 | 83.0 |
| Support Vector Machine | 84.74 | 0.033 | 0.38 | 84.5 |
| Nearest Neighbour | 82.47 | 0.087 | 0.45 | 86.0 |
| Random Forest | 98.05 | 0.14 | 0.92 | 98.0 |
| Adaboost | 85.39 | 0.49 | 0.41 | 85.5 |

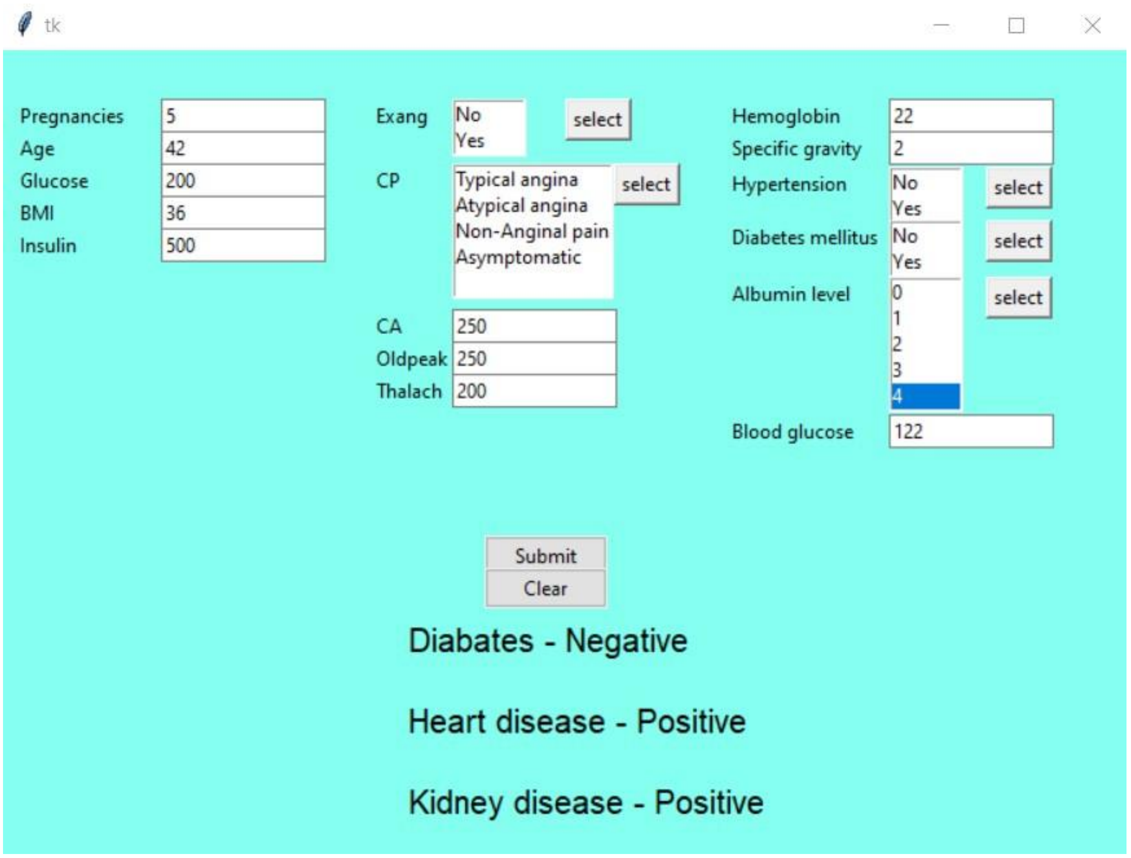
Diabetes Data

| Algorithms | Accuracy | Time | R-Squared | F1 Score |
|---|---|---|---|---|
| Logistic Regression | 79.0 | 0.048 | 0.047 | 75 |
| Decision Tree | 80.0 | 0.012 | -0.046 | 70 |
| NB | 79.0 | 0.019 | 0.046 | 76 |
| Support Vector Machine | 80.0 | 0.127 | 0.08 | 76 |
| Nearest Neighbour | 80.5 | 0.134 | -1.19 | 46 |
| Random Forest | 92.3 | 0.37 | 0.64 | 95 |
| Adaboost | 82.0 | 0.61 | 0.182 | 80 |

Kidney Data

| Algorithms | Accuracy | Time | R-Squared | F1 Score |
|---|---|---|---|---|
| Logistic Regression | 98.33 | 0.045 | 0.93 | 98.5 |
| Decision Tree | 95.0 | 0.011 | 0.775 | 94.5 |
| NB | 91.67 | 0.016 | 0.625 | 91.0 |
| Support Vector Machine | 99.17 | 0.019 | 0.96 | 99.0 |
| Nearest Neighbour | 98.34 | 0.042 | 0.93 | 98.5 |
| Random Forest | 99.17 | 0.13 | 0.96 | 99.0 |
| Adaboost | 99.17 | 0.54 | 0.96 | 99.0 |

Figure. A 2.10 Values of Confusion Matrix for the 7 classifier algorithms used in the proposed model (after checking the necessary boxes and clicking submit)



| | | | | | | |
|---|---|---|---|---|---|---|
| Pregnancies | 5 | Exang | No / Yes | Hemoglobin | 22 | |
| Age | 42 | | | Specific gravity | 2 | |
| Glucose | 200 | CP | Typical angina / Atypical angina / Non-Anginal pain / Asymptomatic | Hypertension | No / Yes | |
| BMI | 36 | | | Diabetes mellitus | No / Yes | |
| Insulin | 500 | | | Albumin level | 0 / 1 / 2 / 3 / 4 | |
| | | CA | 250 | | | |
| | | Oldpeak | 250 | | | |
| | | Thalach | 200 | | | |
| | | | | Blood glucose | 122 | |

Diabates - Negative

Heart disease - Positive

Kidney disease - Positive

Figure. A 2.11 Predicting the Diseases as Positive or Negative with real-time test report values

## REFERENCES

1. Akkem, Y.: “Multi Disease Prediction Model by using Machine Learning and Flask API” IEEE Conference Record #48766; IEEE Xplore ISBN: 978-1-7281-5371-1, International Conference on Communication and Electronics Systems, ICCES (2020).

2. Karthikeyan, H., Menakadevi, T.: Multi-disease prediction model using improved SVM-radial bias technique in healthcare monitoring system. Journal of Ambient Intelligence and Humanized Computing, Springer Germany (2020).

3. Classification and Diagnosis of Diabetes: Standards of Medical Care in Diabetes—2020, American Diabetes Association Diabetes Care 2020, 43 (Suppl. 1): S14–S31 | https://doi.org/10.2337/dc20-S002. (2020).

4. Fontecha, J., González, I., Bravo, J.: A usability study of a mHealth system for diabetes self-management based on framework analysis and usability problem taxonomy methods. J Ambient Intell Hum Comput 1:1–11 (2019).

5. Singh, Y. K., Sinha, N., Singh, S. K.: “Heart Disease Prediction System Using Random Forest, Advances in Computing and Data Sciences”, ICACDS (2016), Communications in Computer and Information Science, vol 721. Springer, Singapore (2017).

6. Devika, R., Avilala, S. N., Subramaniyaswamy, V.: “Comparative Study of Classifier for Chronic Kidney Disease prediction using Naive Bayes, KNN and Random Forest”, 3rd International Conference on Computing Methodologies and Communication (ICCMC), pp. 679-684, Erode (2019).

## PUBLICATION

Dr. R. Shanthakumari, Dr. E. M. Roopa Devi, S. Arvinth, B. Bharaneeshwar, K. V. Hari Prasath, "A Framework for Multi Disease Prediction System Using Random Forest Algorithm", International Virtual Conference on Advances in Communication, Machine Learning and Embedded with Internet of Things (IVCACME 2021).