# CHAPTER 1

# INTRODUCTION

## 1.1 INTRODUCTION TO ORGANIZATION

ANSH InfoTech is an Information Technology & Product Base Company, in India. ANSH InfoTech is committed to Empowering Business Transformation. The company was founded with its core values focused on Creative, People Driven, Socially Conscious, Intense and Ethical. The company provides software products, IT services, for a variety of industries including insurance, banking, capital markets and retail markets. These products and services include managed IT services, application software development and maintenance, payment systems, business intelligence, web development, IT consulting and various transaction processing services. We undertake projects located around the world and are committed to creating a challenging and rewarding work environment whereby our team members are goal focused and motivated throughout our clients' engagement. Our goal is to provide the most cost effective solutions while keeping in mind the urgency to launch products without sacrificing the quality. Excellence lies at our heart; we constantly challenge the convention. Excellence, Partnership and Commitment are the three hallmarks with which we approach our clients, we believe and you will see this in our service. Company also has a Training division that offers learning management and training delivery solutions to Corporations, Institutions and Individuals. It is Global Leader in Skills & Talent Development having three main lines of business: Corporate Learning Group (CLG), Skills and Careers Group (SCG), and (SLG).

**Services Offered by Company:**

* Software Development
* Web Designing/Web Development
* Internet Marketing
* Application Development (iPhone/Android)
* Business Consultancy
* Job Consultancy
* Project & Research Training
* Robotics & Embedded System Development
* Home Automation

## 1.2 INTRODUCTION TO PROJECT

There are many tools related to Disease Prediction. But particularly heart related diseases, tuberculosis, malaria have been analysed and risk level is generated. But generally there are no such tools that are used for prediction of general diseases. So Disease Predictor helps for the prediction of the general diseases analysis and diagnosing of disease become a challenging factor faced by doctors and hospitals both in India and abroad. In order to reduce the large scale of deaths from disease, a quick and efficient Prediction technique is to be discovered. The researchers accelerating their research works to develop a software with the help machine learning algorithm which can help doctors to take decision regarding both prediction and diagnosing of disease. The main objective of this project is predicting the disease of a patient using machine learning algorithms. Comparative study of the various performance of machine learning algorithms is done through graphical representation of the results. The highest mortality of both India and abroad is due to disease. So it is vital time to check this death toll by correctly identifying the disease before time. The matter become a headache for all doctors both in India and abroad. Now a days doctors are adopting many scientific technologies and methodology for both identification and diagnosing not only common disease, but also many fatal diseases. The successful treatment is always attributed by right and accurate diagnosis. Doctors may sometimes fail to take accurate decisions while diagnosing the disease of a patient, therefore disease prediction systems which use machine learning algorithms assist in such cases to get accurate results.

## 1.3 PROJECT CATEGORY

This project is based on Software Development which makes it portable for the hospitals and the doctors to manage it easily. The patients will not have to spend explicitly on the check-up and its cure. This makes it more efficient for use.

## 1.4 PROJECT OBJECTIVES

The objectives of the project DISEASE PREDICTION are:

1. To find the rate of disease in human beings.
2. To measure the probability of a user for having disease.
3. To implement rule based algorithm as analysing technique into a system.
4. To predict that whether a person will have disease or not.
5. To represent and measure the variation of each factor responsible for disease.

## 1.5 PROBLEM FORMULATION

Disease Prediction determines whether person can have disease in near future or having it right now. I used various features for determining the disease using the dataset for evaluation. The dataset in the atlas is sectioned, based on various features. A machine learning algorithm, designed using python language libraries, is used. The outputs generated are reliable and can be readily used by the doctors.

## 1.6 IDENTIFICATION/REORGANIZATION OF NEED

The most important tasks of the system analysis are to identify the problem because without having an idea of the problem it is impossible to specify the requirement for a new project with any accuracy.

The main problems faced in today’s bone age analysis are:

* The designing of the system is complex.
* Can lead to ambiguity in some case.
* Manual analysis requires a lot of expertise.
* No working system is available in the market.

It was analysed that all these needs of the user are being fulfilled by this system, so this system will be appropriate to introduce.

## 1.7 EXISTING SYSTEM

Though there are many existing systems, but those seems to be very expensive as if a user wants a regular check-up then hospitals are quite expensive. So better system instead of that can be disease prediction machine which is developed and modified using python language. so, it can give an idea to the user that he will face disease in near future or not. Through this user can regularly check disease just to get a rough idea without wasting a huge amount of money in hospitals.

**Limitations of exiting system**

* It is a time-consuming process.
* It is very expensive.
* No system tells us about General Diseases.

## 1.8 PROPOSED SYSTEM

The aim of the proposed system is to develop a system of improved facilities. The proposed system can overcome all the limitations of the manual assessment system. The system provides proper security and reduces the manual work. The proposed system will help the user to sustain in market increasing rapidly and reduces mental conflict. The proposed system helps the doctors to work user friendly and he can easily do his jobs without time lacking.

## 1.9 UNIQUE FEATURES OF THE SYSTEM

The system is very simple in design and to implement. The system requires very low system resources and the system will work in almost all configurations.

The features are:

* Minimize manual data entry.
* Interface that is both doctor and patient friendly.
* Provides accurate bone age to the doctors.
* Ensures precision.
* Minimum Time needed for various processing.
* Reduces doctor’s efforts to analyse bone age and hence predict the adult height.
* Greater efficiency.
* Reduces workload.

# 

# CHAPTER 2

# REQUIREMENT ANALYSIS AND SYSTEM SPECIFICATION

## 2.1 FEASIBILITY STUDY

The feasibility of the project is analysed in this phase to put forth a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are:

1. Technical Feasibility
2. Economic Feasibility
3. Operational Feasibility

A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing business or proposed venture, opportunities and threats present in the natural environment, the resources required to carry through, and ultimately the prospects for success. In this project, we will look into all the possibilities and analyze whether it is profitable to work onto it or not.

### 2.1.1 TECHNICAL FEASIBILITY

This assessment is based on an outline design of system requirements, to determine whether the store has the technical expertise to handle completion of the project. When writing the report, the following were under consideration:

* A brief description of the framework to assess more possible factors which could affect the study
* The part of the business being examined (marketing, economical)
* The human and economic factor
* The possible solutions to the problem within minimum time

The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the store and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system.

After the above analysis, it was concluded that the system is technically feasible.

### 2.1.2 ECONOMICAL FEASIBILITY

The purpose of an economic feasibility study is to demonstrate the net benefit of a proposed project for accepting or disbursing electronic funds/benefits, taking into consideration the benefits and costs to the agency, other state agencies, and the general public as a whole.

The following costs were estimated:

* One-time development cost
* One-time H/W cost (if the user does not have a computer system)
* Benefits in reduced cost, error and saving will be made by reduction of present system expenses, time saving and increased accuracy

Once the application is developed, it would yield the user results if it is used effectively. If the offers are sent strategically, it will surely be profitable for the user. After the above analysis, it was concluded that the system is economically feasible.

### 2.1.3 OPERATIONAL FEASIBILITY

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

The operational feasibility assessment focuses on the degree to which the proposed development project fits in with the existing business environment and objectives with regard to development schedule, delivery date, corporate culture and existing business processes.

To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters as reliability, maintainability, supportability, usability, reducibility, disposability, sustainability, affordability and others.

.

## 2.2 SOFTWARE REQUIREMENT SPECIFICATION

### 2.2.1 INTRODUCTION

A software requirements specification (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. The SRS is developed based the agreement between customer and contractors. It may include the use cases of how user is going to interact with software system. The software requirement specification document consistent of all necessary requirements required for project development. To develop the software system, we should have clear understanding of Software system. To achieve this, we need to continuous communication with customers to gather all requirements.

A good SRS defines the how Software System will interact with all internal modules, hardware, communication with other programs and human user interactions with wide range of real life scenarios. Using the Software requirements specification (SRS) document on QA lead, managers creates test plan. It is very important that testers must be cleared with every detail specified in this document in order to avoid faults in test cases and its expected results.

It is highly recommended to review or test SRS documents before start writing test cases and making any plan for testing

### 2.2.2 SOFTWARE REQUIREMENTS

The technical specifications of requirements for the software are as follows:

* About Operating System: Windows XP or Later.
* Windows 7 or later
* Linux / Ubuntu (Excluding Kali Linux)
* Python Version 3.5 or more

### 2.2.3 HARDWARE REQUIREMENTS

Hardware requirements include that hardware which is required for its working. It includes:

* Pentium 4 Computer (Minimum)
* 512 MB RAM (Minimum)
* Hard Disk - 40GB (Minimum)

### 2.2.4 FUNCTIONAL REQUIREMENTS

A functional requirement defines a function of a system or its component. A function is described as a set of inputs, the behavior, and outputs.

Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability).

As defined in requirements engineering, functional requirements specify particular results of a system. This should be contrasted with non-functional requirements which specify overall characteristics such as cost and reliability. Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.

The functional requirements are defined below:

1. Keeping the dataset of the atlas.
2. Keeping records of all the features required for analysis.
3. Keeping the record of the data to be plotted.
4. Keeping details about system used for prediction.

### 2.2.5 NON FUNCTIONAL REQUIREMENTS

1. **Performance requirement:** All data entered shall be up to mark and no flaws shall be there for the performance to be 100%.
2. **Platform constraints:** The main target is to generate an intelligent system to predict the adult height.
3. **Accuracy and Precision**: Requirements are accuracy and precision of the data
4. **Modifiability:** Requirements about the effort required to make changes in the software. Often, the measurement is personnel effort (person- months).
5. **Portability:** Since mobile phone is handy so it is portable and can be carried and used whenever required.
6. **Reliability**: Requirements about how often the software fails. The definition of a failure must be clear. Also, don't confuse reliability with availability which is quite a different kind of requirement.  Be sure to specify the consequences of software failure, how to protect from failure, a strategy for error Prediction, and a strategy for correction.
7. **Security**: One or more requirements about protection of your system and its data.
8. **Usability:** Requirements about how difficult it will be to learn and operate the system. The requirements are often expressed in learning time or similar metrics.

## 2.3 VALIDATION

It is the process of checking that a software system meets specifications and that it fulfils its intended purpose. It may also be referred to as software quality control. It is normally the responsibility of software testers as part of the software development lifecycle.   
Software validation checks that the software product satisfies or fits the intended use (high-level checking), i.e., the software meets the user requirements, not as specification artefacts or as needs of those who will operate the software only; but, as the needs of all the stakeholders (such as users, operators, administrators, managers, investors, etc.). There are two ways to perform software validation: internal and external.

## 2.4 EXPECTED HURDLES

Expected hurdles were some hardware requirements that include hardware which were expensive. Next was searching the required and proper dataset upon which we can work easily and which gives a good no. of accuracy to our project. Then choosing the machine learning model which fits and helps in providing the required solution. High variance was also one of the problem. Other hurdles were solved using normalization and standardization

## 2.5 SDLC MODEL TO BE USED

We chose Iterative Waterfall model as our systems development life cycle (SDLC), also referred to as the application development life-cycle. In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. After each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

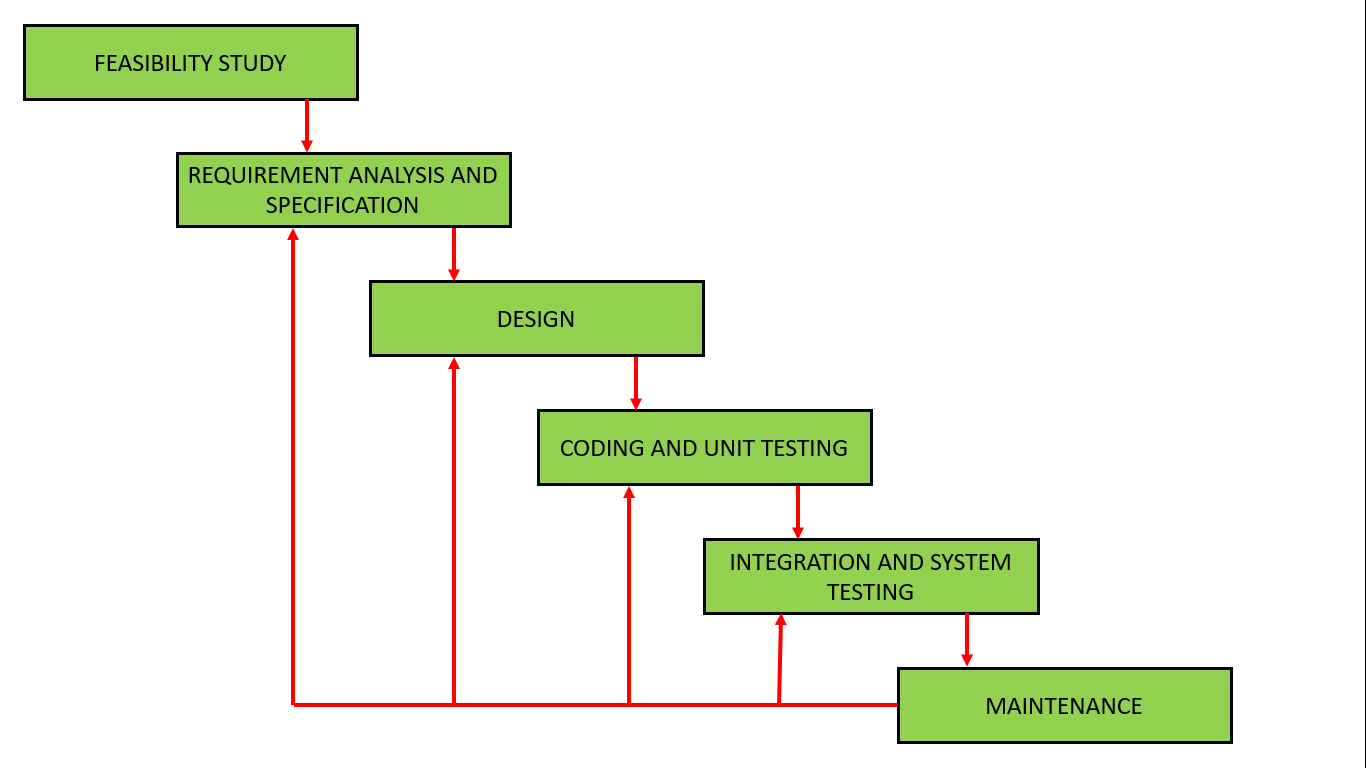


Figure 2.1: Iterative Waterfall Model

The sequential phases in Iterative Waterfall model are −

1. **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
2. **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
3. **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
4. **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
5. **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
6. **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

# CHAPTER 3

# SYSTEM DESIGN

## 3.1 DESIGN APPROCH

OBJECT ORIENTED DESIGN APPROCH

## 3.2 DETAIL DESIGN

After the analysis phase, the conceptual model is developed further into an object-oriented model using object-oriented design (OOD). In OOD, the technology-independent concepts in the analysis model are mapped onto implementing classes, constraints are identified, and interfaces are designed, resulting in a model for the solution domain. In a nutshell, a detailed description is constructed specifying how the system is to be built on concrete technologies

The stages for object–oriented design can be identified as −

* Definition of the context of the system
* Designing system architecture
* Identification of the objects in the system
* Construction of design models
* Specification of object interfaces

Object-oriented system design involves defining the context of a system followed by designing the architecture of the system.

**Context** − The context of a system has a static and a dynamic part. The static context of the system is designed using a simple block diagram of the whole system which is expanded into a hierarchy of subsystems. The subsystem model is represented by UML packages. The dynamic context describes how the system interacts with its environment. It is modelled using **use case diagrams**.

**System Architecture** − The system architecture is designed on the basis of the context of the system in accordance with the principles of architectural design as well as domain knowledge. Typically, a system is partitioned into layers and each layer is decomposed to form the subsystems.

**Identifying Patterns**

While designing applications, some commonly accepted solutions are adopted for some categories of problems. These are the patterns of design. A pattern can be defined as a documented set of building blocks that can be used in certain types of application development problems.

Some commonly used design patterns are −

Façade pattern

Model view separation pattern

Observer pattern

Model view controller pattern

Publish subscribe pattern

Proxy pattern

**Object Design**

After the hierarchy of subsystems has been developed, the objects in the system are identified and their details are designed. Here, the designer details out the strategy chosen during the system design. The emphasis shifts from application domain concepts toward computer concepts. The objects identified during analysis are etched out for implementation with an aim to minimize execution time, memory consumption, and overall cost.

Object design includes the following phases −

Object identification

Object representation, i.e., construction of design models

Classification of operations

Algorithm design

Design of relationships

Implementation of control for external interactions

**Object Identification**

The first step of object design is object identification. The objects identified in the object–oriented analysis phases are grouped into classes and refined so that they are suitable for actual implementation.

The functions of this stage are −

Identifying and refining the classes in each subsystem or package

Defining the links and associations between the classes

Designing the hierarchical associations among the classes, i.e., the generalization/specialization and inheritances

Designing aggregations

**Object Representation**

Once the classes are identified, they need to be represented using object modelling techniques. This stage essentially involves constructing UML diagrams.

There are two types of design models that need to be produced −

**Static Models** − To describe the static structure of a system using class diagrams and object diagrams.

**Dynamic Models** − To describe the dynamic structure of a system and show the interaction between classes using interaction diagrams and state–chart diagrams.

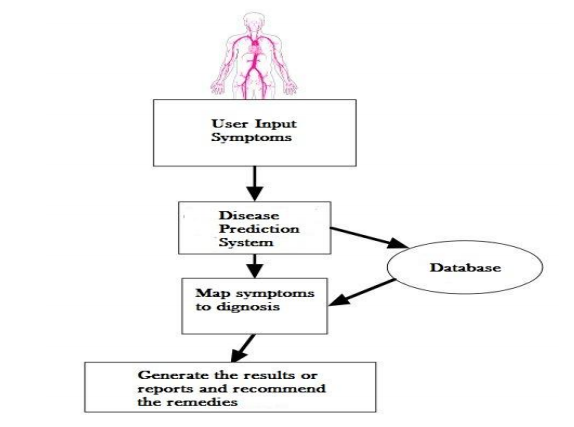
## 3.3 SYSTEM DESIGN USING VARIOUS STRUCTURED ANALYSIS AND DESIGN TOOL

**USER REQUIREMENTS**

* The system should ensure the security of the data.
* The System should reduce human efforts and saves time.
* Project should be able to analyse the data and predict results with great precision.
* This project should enable you to navigate through the features in dataset.
* This project should enable you to create duplicate data if it is required.
* The system should also be user-friendly. It should use a Graphical User Interface (GUI).

### 3.3.1 DATA FLOW DIAGRAMS

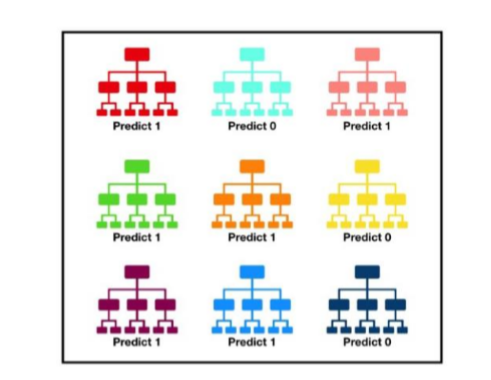
A data dictionary is a structured repository of data about data. It is a set of rigorous definition of all DFD data elements and data structures.



**Figure 3.1: Data flow Diagram**

Figure 3.1 show the data flow diagram of the Decision Tree. In this diagram it shows that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

Random forest, like its name implies, consists of a large num individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction (see figure below).



**Figure 3.2: Prediction (six 1’s and 2 0’s)**

Figure 3.2 Depict the Concept: The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate that of the individual predictions. The reason for this wonderful effect is that the trees protect each other from their individual errors (as long as they don’t constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction.

**DFD Levels and Layers**

A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece.  DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish.

## 3.4 USER INTERFACE DESIGN

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications.

**To create a tkinter :**

Importing the module – tkinter

Create the main window (container)

Add any number of widgets to the main window

Apply the event Trigger on the widgets.

Importing tkinter is same as importing any other module in the python code. Note that the name of the module in Python 2.x is ‘Tkinter’ and in Python 3.x is ‘tkinter’.

## 3.5 Database Design

* Symptoms.
* Name.
* Diseases Predicted.

## 3.5.1 ER Diagrams

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases.

At first glance an entity relationship diagram looks very much like a flowchart. It is the specialized symbols, and the meanings of those symbols, that make it unique.

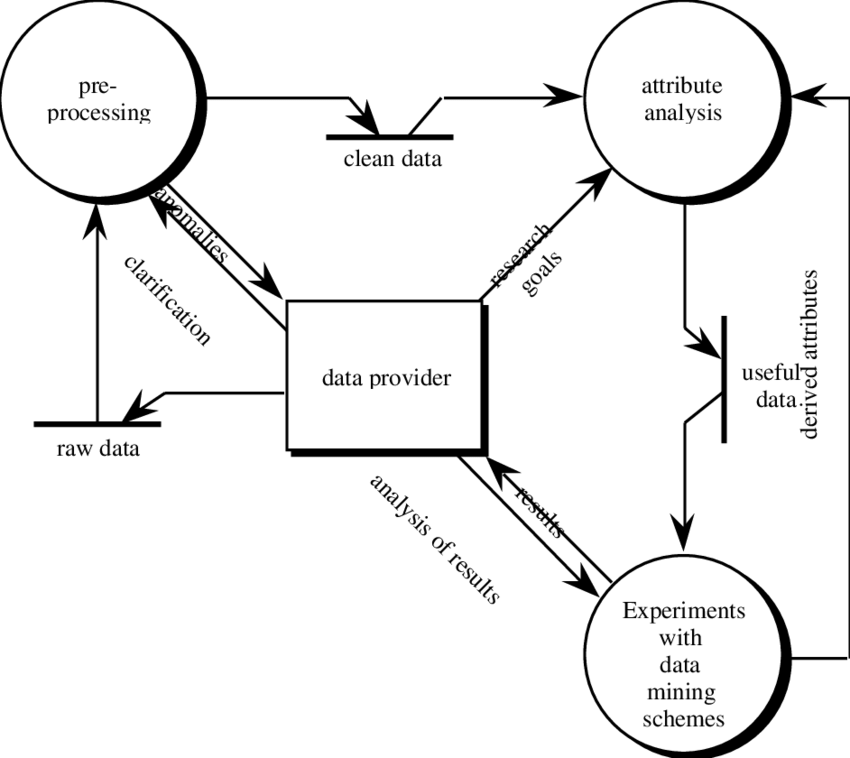


Figure 3.3: ER Diagram

## 3.5.2 DATABASE MANIPULATION

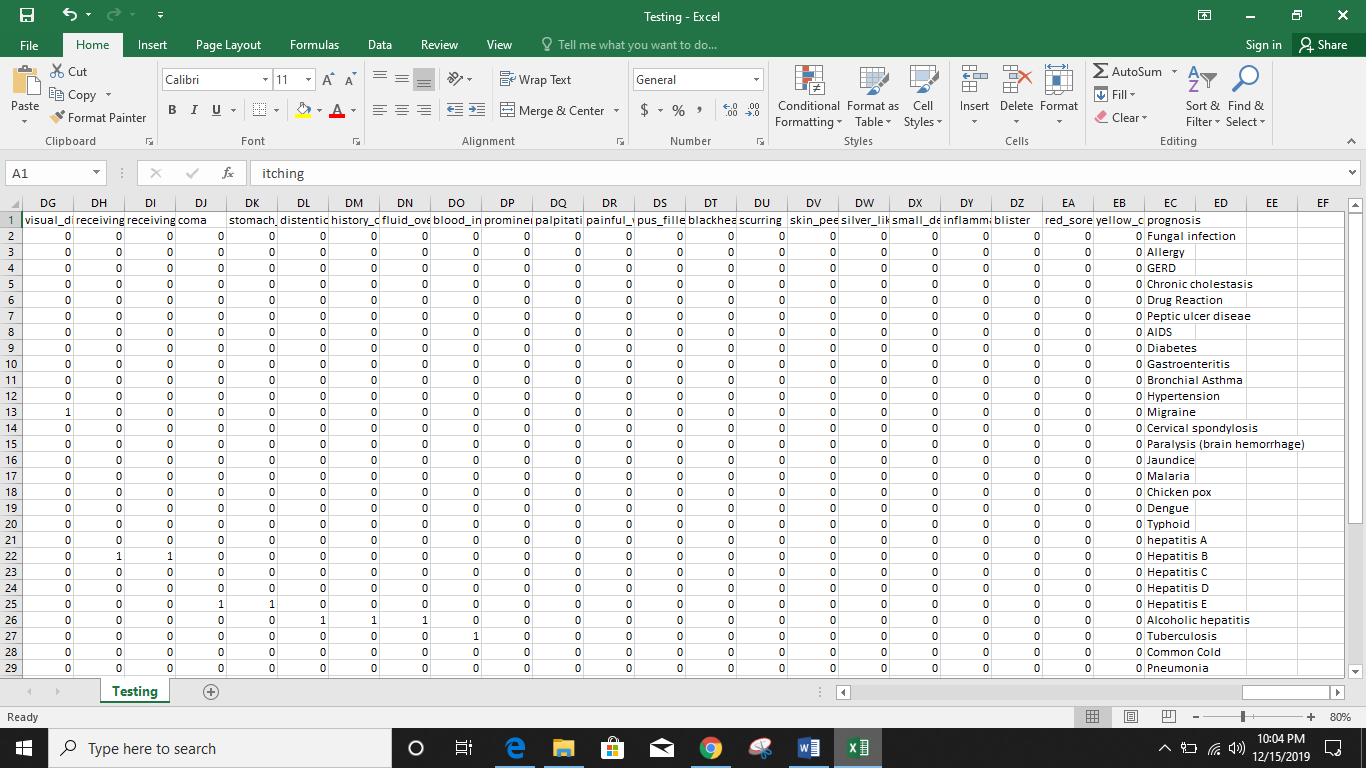
**Data Collection**

Data has been collected from KAGGLE website (Kaggle Indians Disease Database), Class variable (0 or 1) Which is the original source of Research Center of National Institute of Disease and Digestive and Kidney Diseases, RMI Group Leader Applied Physics Laboratory the Johns Hopkins University. This data has already been used for forecast the onset of disease using ADAP learning algorithm. The total Number of Instances is 42, which is completely used in this study. It contains 106 Attributes plus one class (Label) column. Each Attribute is numeric-valued; attributes of this dataset are as follow:

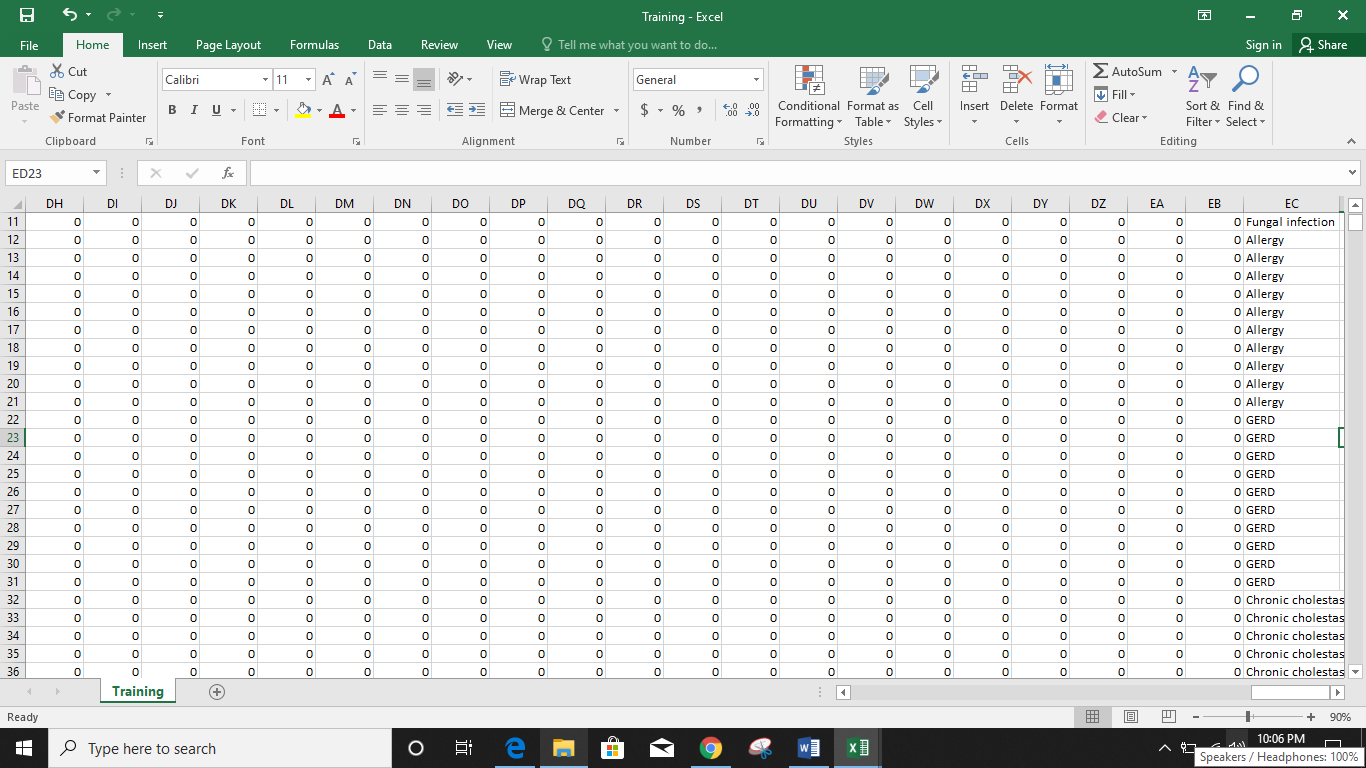
This dataset also contains the Missing Attribute Values, which is handled in the next step is Pre-processing using some statistical techniques. In Class values, distribution is like if there is 1, then it interpreted as” tested positive for disease”, if the Class value is 0, it means” tested negative for disease” Training and test data divided into a certain number - 107 for training and 42 for testing data.

**Data Pre-processing**

Data value in diﬀerent attributes are having some missing values. These missing values can lead to inaccurate result; also it may reduce the model accuracy.



**Fig. 3.4 Testing Sample Data**



**Fig. 3.5 Training Sample Data**

So to handle these missing value mean of column method used to replace 0 with appropriate calculation. To handle programmatically this missing values, Numpy package of python were used to get mean function and manipulate the existing column array value from 0 to calculated result. One thing is also important to prioritize the attribute, so that Artiﬁcial Neural Network calculate weight of each neurons (Attribute) as per the given priority. Prioritizing attribute is need to get more accuracy of disease detection, which shows that which cause aﬀects the disease detection on which priority. Table 2 show the attribute priority:

## 3.6 METHODOLOGY

**Step 1**: Data collection and dataset preparation This will involve collection of medical information from various sources like hospitals, then pre-processing is applied on dataset which will remove all the unnecessary data and extract important features from data.

**Step 2**: Developing a probabilistic model and deep learning approach (RNN) for Disease Prediction in this step probabilistic model and deep learning approach based on RNN is to be developed it will run effectively on extensive databases of healthcare. And generate decision tree also it can deal with a huge number of information variables without variable deletion.

**Step 3**: Training and experimentation on datasets The Disease Prediction model will be trained on the dataset of diseases to do the prediction accurately and produce Confusion matrix. In this project 3 different algorithms were used -

* Decision Tree
* Random Forest
* Naïve Bayes

**Step 4**: Deployment and analysis on real life scenario The trained and tested prediction model will be deployed in a real-life scenario made by the human experts & will be leveraged for further improvement in the methodology.

# CHAPTER 4

# IMPLEMENTATION, TESTING AND MAINTENANCE

## 4.1 INTRODUCTION TO LANGUAGES, IDE’S AND TECHNOLOGY USED FOR IMPLEMENTATION

### 4.1.1 PYTHON

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community after 30 years.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter(), map(), and reduce() functions; list comprehensions, dictionaries, and sets; and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

* Beautiful is better than ugly
* Explicit is better than implicit
* Simple is better than complex
* Complex is better than complicated
* Readability counts

Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

While offering choice in coding methodology, the Python philosophy rejects exuberant syntax (such as that of Perl) in favor of a simpler, less-cluttered grammar. As Alex Martelli put it: "To describe something as 'clever' is not considered a compliment in the Python culture." Python's philosophy rejects the Perl "there is more than one way to do it" approach to language design in favor of "there should be one—and preferably only one—obvious way to do it".

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of CPython that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name—a tribute to the British comedy group Monty Python—and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard foo and bar.

A common neologism in the Python community is pythonic, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic. Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as Pythonists, Pythonistas, and Pythoneers.

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal.

## 4.2 MACHINE LEARNING

Machine learning (ML) is a field of artificial intelligence that uses statistical techniques to give computer systems the ability to "learn" (e.g., progressively improve performance on a specific task) from data, without being explicitly programmed.

The name machine learning was coined in 1959 by Arthur Samuel. Machine learning explores the study and construction of algorithms that can learn from and make predictions on data – such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions, through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or infeasible; example applications include email filtering, Prediction of network intruders, and computer vision.

Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data mining, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning.

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

**Machine learning tasks** Machine learning tasks are typically classified into several broad categories:

**Supervised learning**: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs. As special cases, the input signal can be only partially available, or restricted to special feedback.

**Semi-supervised learning**: The computer is given only an incomplete training signal: a training set with some (often many) of the target outputs missing.

**Active learning**: The computer can only obtain training labels for a limited set of instances (based on a budget), and also has to optimize its choice of objects to acquire labels for. When used interactively, these can be presented to the user for labelling.

**Unsupervised learning**: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

**Reinforcement learning**: Data (in form of rewards and punishments) are given only as feedback to the program's actions in a dynamic environment, such as driving a vehicle or playing a game against an opponent.

A support vector machine is a classifier that divides its input space into two regions, separated by a linear boundary. Here, it has learned to distinguish black and white circles.

Another categorization of machine learning tasks arises when one considers the desired output of a machine-learned system

* In classification, inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".
* In regression, also a supervised problem, the outputs are continuous rather than discrete.
* In clustering, a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.
* Density estimation finds the distribution of inputs in some space.
* Dimensionality reduction simplifies inputs by mapping them into a lower-dimensional space. Topic modeling is a related problem, where a program is given a list of human language documents and is tasked to find out which documents cover similar topics.

Among other categories of machine learning problems, learning to learn learns its own inductive bias based on previous experience. Developmental learning, elaborated for robot learning, generates its own sequences (also called curriculum) of learning situations to cumulatively acquire repertoires of novel skills through autonomous self-exploration and social interaction with human teachers and using guidance mechanisms such as active learning, maturation, motor synergies, and imitation.

**Relation to data mining**

Machine learning and data mining often employ the same methods and overlap significantly, but while machine learning focuses on prediction, based on known properties learned from the training data, data mining focuses on the discovery of (previously) unknown properties in the data (this is the analysis step of knowledge discovery in databases). Data mining uses many machine learning methods, but with different goals; on the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy. Much of the confusion between these two research communities (which do often have separate conferences and separate journals, ECML PKDD being a major exception) comes from the basic assumptions they work with: in machine learning, performance is usually evaluated with respect to the ability to reproduce known knowledge, while in knowledge discovery and data mining (KDD) the key task is the discovery of previously unknown knowledge. Evaluated with respect to known knowledge, an uninformed (unsupervised) method will easily be outperformed by other supervised methods, while in a typical KDD task, supervised methods cannot be used due to the unavailability of training data. Relation to optimization

Machine learning also has intimate ties to optimization: many learning problems are formulated as minimization of some loss function on a training set of examples. Loss functions express the discrepancy between the predictions of the model being trained and the actual problem instances (for example, in classification, one wants to assign a label to instances, and models are trained to correctly predict the pre-assigned labels of a set of examples). The difference between the two fields arises from the goal of generalization: while optimization algorithms can minimize the loss on a training set, machine learning is concerned with minimizing the loss on unseen samples.

**Relation to statistics**

Machine learning and statistics are closely related fields. According to Michael I. Jordan, the ideas of machine learning, from methodological principles to theoretical tools, have had a long pre-history in statistics. He also suggested the term data science as a placeholder to call the overall field.

Leo Breiman distinguished two statistical modelling paradigms: data model and algorithmic model, wherein "algorithmic model" means more or less the machine learning algorithms like Random forest. Some statisticians have adopted methods from machine learning, leading to a combined field that they call statistical learning.

**Deep learning**

Falling hardware prices and the development of GPUs for personal use in the last few years have contributed to the development of the concept of deep learning which consists of multiple hidden layers in an artificial neural network. This approach tries to model the way the human brain processes light and sound into vision and hearing. Some successful applications of deep learning are computer vision and speech recognition.

**Decision tree learning**

Decision tree learning uses a decision tree as a predictive model, which maps observations about an item to conclusions about the item's target value.

**Association rule learning**

Association rule learning is a method for discovering interesting relations between variables in large databases.

**Clustering**

Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to some predesignated criterion or criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated for example by internal compactness (similarity between members of the same cluster) and separation between different clusters. Other methods are based on estimated density and graph connectivity. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis.

**One Hot and Label Encoding**

These two encoders are parts of the SciKit Learn library in Python, and they are used to convert categorical data, or text data, into numbers, which our predictive models can better understand. Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning. What one hot encoding does is, it takes a column which has categorical data, which has been label encoded, and then splits the column into multiple columns. The numbers are replaced by 1s and 0s, depending on which column has what value.

## 4.3 TESTING TECHNIQUES AND TEST PLANS

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects).

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

• Meets the requirements that guided its design and development

• Responds correctly to all kinds of inputs

• Performs its functions within an acceptable time

• Sufficiently usable

• Can be installed and run in its intended environments

• Achieves the general result its stakeholders desire

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects). The job of testing is an iterative process as when one bug is fixed, it can illuminate other, deeper bugs, or can even create new ones.

Software testing can provide objective, independent information about the quality of software and risk of its failure to users and/or sponsors.

Software testing can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system requirements have been defined and then implemented in testable programs. In contrast, under an Agile approach, requirements, programming, and testing are often done concurrently.

**Data-driven testing** (DDT) is a term used in the testing of computer software to describe testing done using a table of conditions directly as test inputs and verifiable outputs as well as the process where test environment settings and control are not hard-coded. In the simplest form the tester supplies the inputs from a row in the table and expects the outputs which occur in the same row. The table typically contains values which correspond to boundary or partition input spaces. In the control methodology, test configuration is "read" from a database.

Data-driven testing is the creation of test scripts to run together with their related data sets in a framework. The framework provides re-usable test logic to reduce maintenance and improve test coverage. Input and result (test criteria) data values can be stored in one or more central data sources or databases, the actual format and organization can be implementation specific.

The data comprises variables used for both input values and output verification values. In advanced (mature) automation environments data can be harvested from a running system using a purpose-built custom tool or sniffer, the DDT framework thus performs playback of harvested data producing a powerful automated regression testing tool.

Navigation through the program, reading of the data sources, and logging of test status and information are all coded in the test script.

# CHAPTER 5

# RESULTS AND DISCUSSIONS

## 5.1 BRIEF DISCRIPTION OF VARIOUS MODULES OF THE SYSTEM

### 5.1.1 LIBRARIES USED

**NumPy**

NumPy stands for ‘Numerical Python’ or ‘Numeric Python’. It is an open source module of Python which provides fast mathematical computation on arrays and matrices. Since, arrays and matrices are an essential part of the Machine Learning ecosystem, NumPy along with Machine Learning modules like Scikit-learn, Pandas, Matplotlib, TensorFlow, etc. complete the Python Machine Learning Ecosystem.

NumPy provides the essential multi-dimensional array-oriented computing functionalities designed for high-level mathematical functions and scientific computation. Numpy can be imported into the notebook using

|  |  |
| --- | --- |
| 1 | >>> import numpy as np |

NumPy’s main object is the homogeneous multidimensional array. It is a table with same type elements, i.e, integers or string or characters (homogeneous), usually integers. In NumPy, dimensions are called axes. The number of axes is called the rank.

There are several ways to create an array in NumPy like np.array, np.zeros, no.ones, etc. Each of them provides some flexibility.

Some of the important attributes of a NumPy object are:

1. **Ndim:** displays the dimension of the array
2. **Shape:** returns a tuple of integers indicating the size of the array
3. **Size:** returns the total number of elements in the NumPy array
4. **Dtype**: returns the type of elements in the array, i.e., int64, character
5. **Itemsize:** returns the size in bytes of each item
6. **Reshape**: Reshapes the NumPy array

**Pandas**

Similar to NumPy, Pandas is one of the most widely used python libraries in data science. It provides high-performance, easy to use structures and data analysis tools. Unlike NumPy library which provides objects for multi-dimensional arrays, Pandas provides in-memory 2d table object called Dataframe. It is like a spreadsheet with column names and row labels.

Hence, with 2d tables, pandas is capable of providing many additional functionalities like creating pivot tables, computing columns based on other columns and plotting graphs. Pandas can be imported into Python using:

|  |  |
| --- | --- |
| 1 | >>> import pandas as pd |

Some commonly used data structures in pandas are:

1. **Series objects**: 1D array, similar to a column in a spreadsheet
2. **DataFrame objects:** 2D table, similar to a spreadsheet
3. **Panel objects:** Dictionary of DataFrames, similar to sheet in MS Excel

Pandas Series object is created using pd.Series function. Each row is provided with an index and by defaults is assigned numerical values starting from 0. Like NumPy, Pandas also provide the basic mathematical functionalities like addition, subtraction and conditional operations and broadcasting.

Pandas dataframe object represents a spreadsheet with cell values, column names, and row index labels. Dataframe can be visualized as dictionaries of Series. Dataframe rows and columns are simple and intuitive to access. Pandas also provide SQL-like functionality to filter, sort rows based on conditions.

**Matplotlib**

Data visualization helps you to better understand your data, discover things that you wouldn’t discover in raw format and communicate your findings more efficiently to others.  
The best and most well-known Python data visualization library is Matplotlib.

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They’re typically instruments for reasoning about quantitative information.

Features:

* Usable as a MATLAB replacement, with the advantage of being free and open source
* Supports dozens of backends and output types, which means you can use it regardless of which operating system you’re using or which output format you wish to use
* Pandas itself can be used as wrappers around MATLAB API to drive MATLAB like a cleaner
* Low memory consumption and better runtime behavior

**Seaborn**

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Here is some of the functionality that seaborn offers:

* A dataset-oriented API for examining [relationships](https://seaborn.pydata.org/examples/scatter_bubbles.html#scatter-bubbles) between [multiple variables](https://seaborn.pydata.org/examples/faceted_lineplot.html#faceted-lineplot)
* Specialized support for using categorical variables to show [observations](https://seaborn.pydata.org/examples/jitter_stripplot.html#jitter-stripplot) or [aggregate statistics](https://seaborn.pydata.org/examples/pointplot_anova.html#pointplot-anova)
* Options for visualizing [univariate](https://seaborn.pydata.org/examples/distplot_options.html#distplot-options) or [bivariate](https://seaborn.pydata.org/examples/joint_kde.html#joint-kde) distributions and for [comparing](https://seaborn.pydata.org/examples/horizontal_boxplot.html#horizontal-boxplot) them between subsets of data
* Automatic estimation and plotting of [linear regression](https://seaborn.pydata.org/examples/anscombes_quartet.html#anscombes-quartet) models for different kinds [dependent](https://seaborn.pydata.org/examples/logistic_regression.html#logistic-regression) variables
* Convenient views onto the overall [structure](https://seaborn.pydata.org/examples/scatterplot_matrix.html#scatterplot-matrix) of complex datasets
* High-level abstractions for structuring [multi-plot grids](https://seaborn.pydata.org/examples/faceted_histogram.html#faceted-histogram) that let you easily build [complex](https://seaborn.pydata.org/examples/pair_grid_with_kde.html#pair-grid-with-kde) visualizations
* Tools for choosing [color palettes](https://seaborn.pydata.org/tutorial/color_palettes.html" \l "palette-tutorial) that faithfully reveal patterns in your data

Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

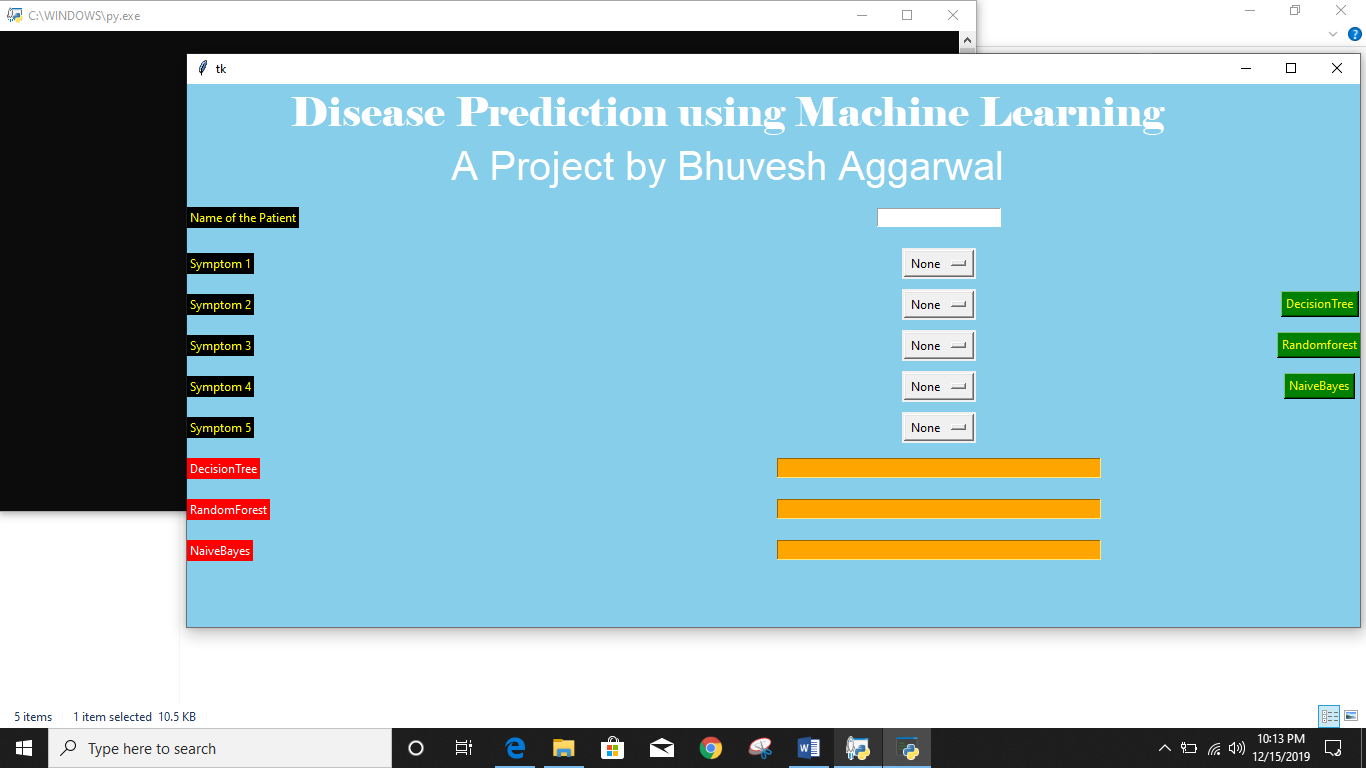
**Scikit-learn**

Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It’s built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib!

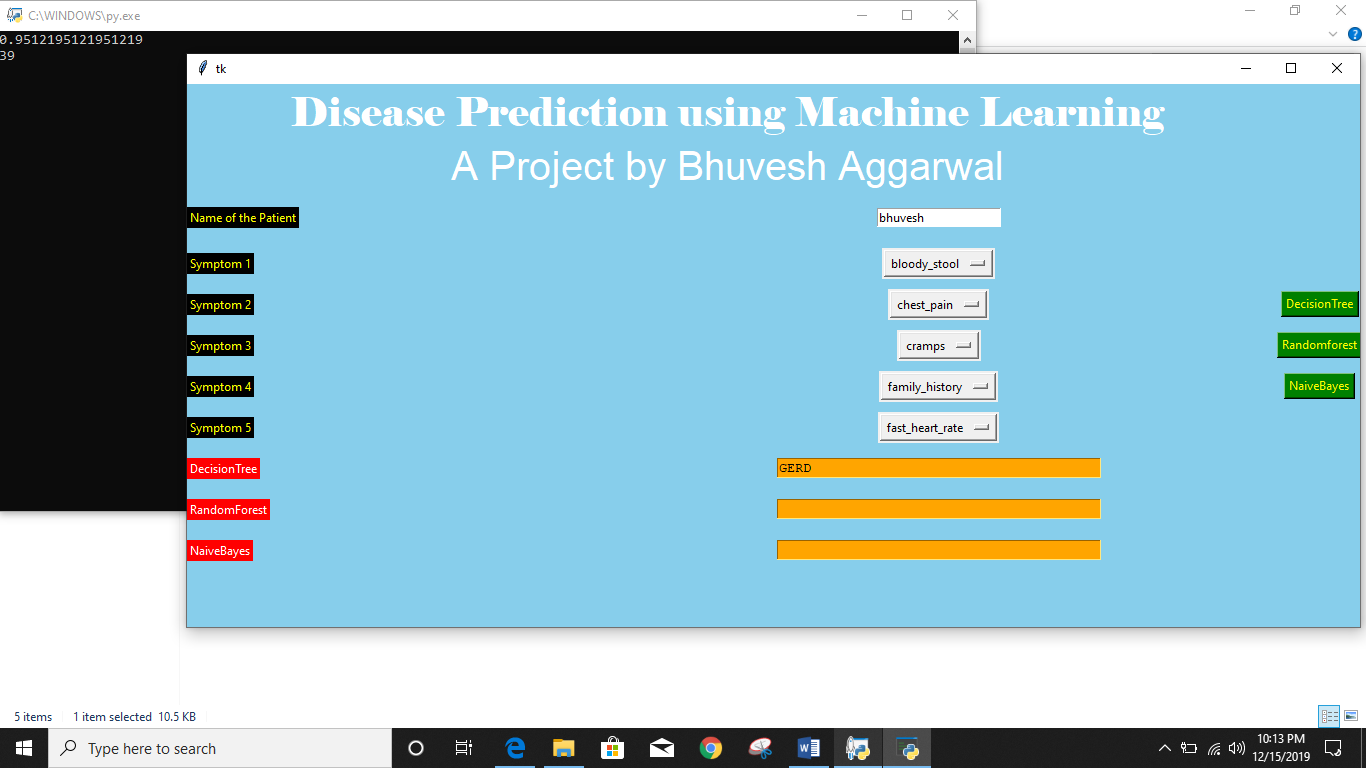
The functionality that scikit-learn provides include:

* **Regression**, including Linear and Logistic Regression
* **Classification**, including K-Nearest Neighbors
* **Clustering**, including K-Means and K-Means++
* **Model selection**
* **Preprocessing**, including Min-Max Normalization

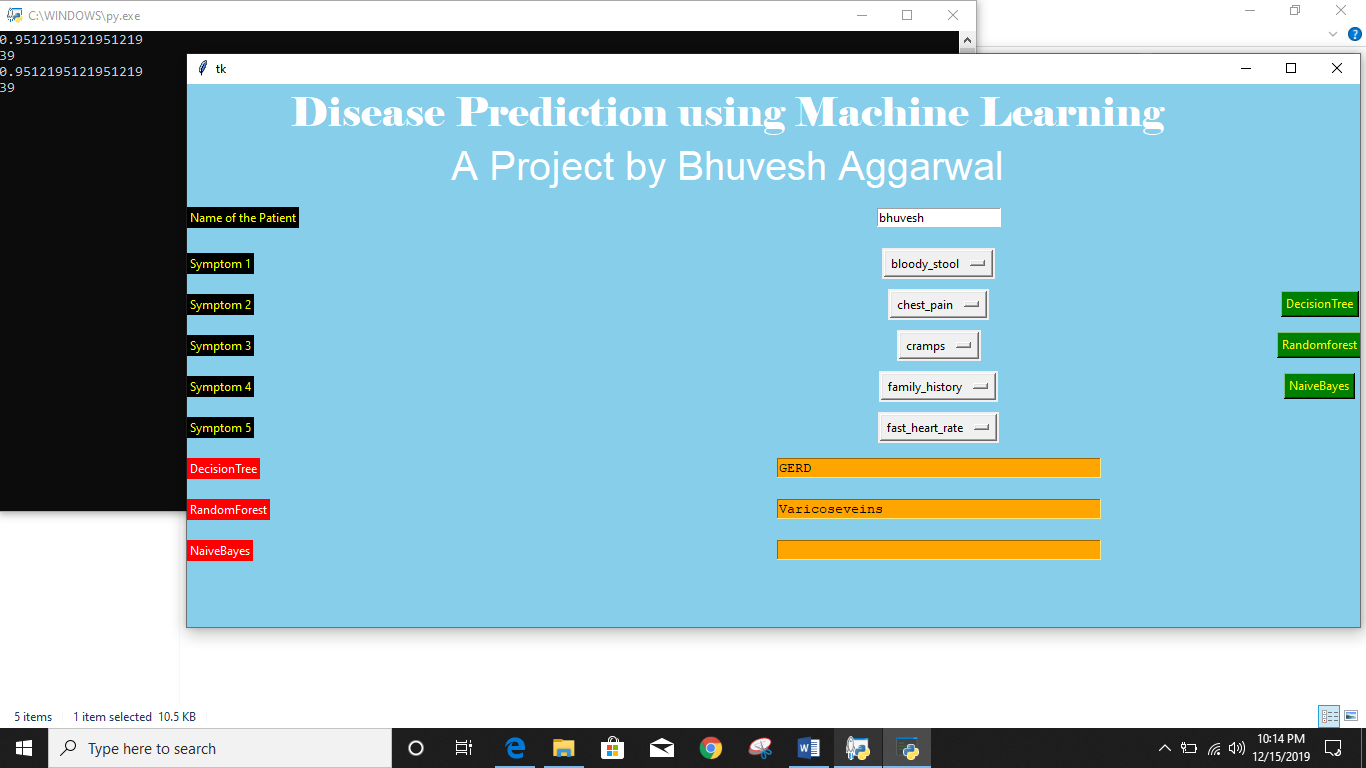
## 5.2 SNAPSHOTS OF SYSTEM



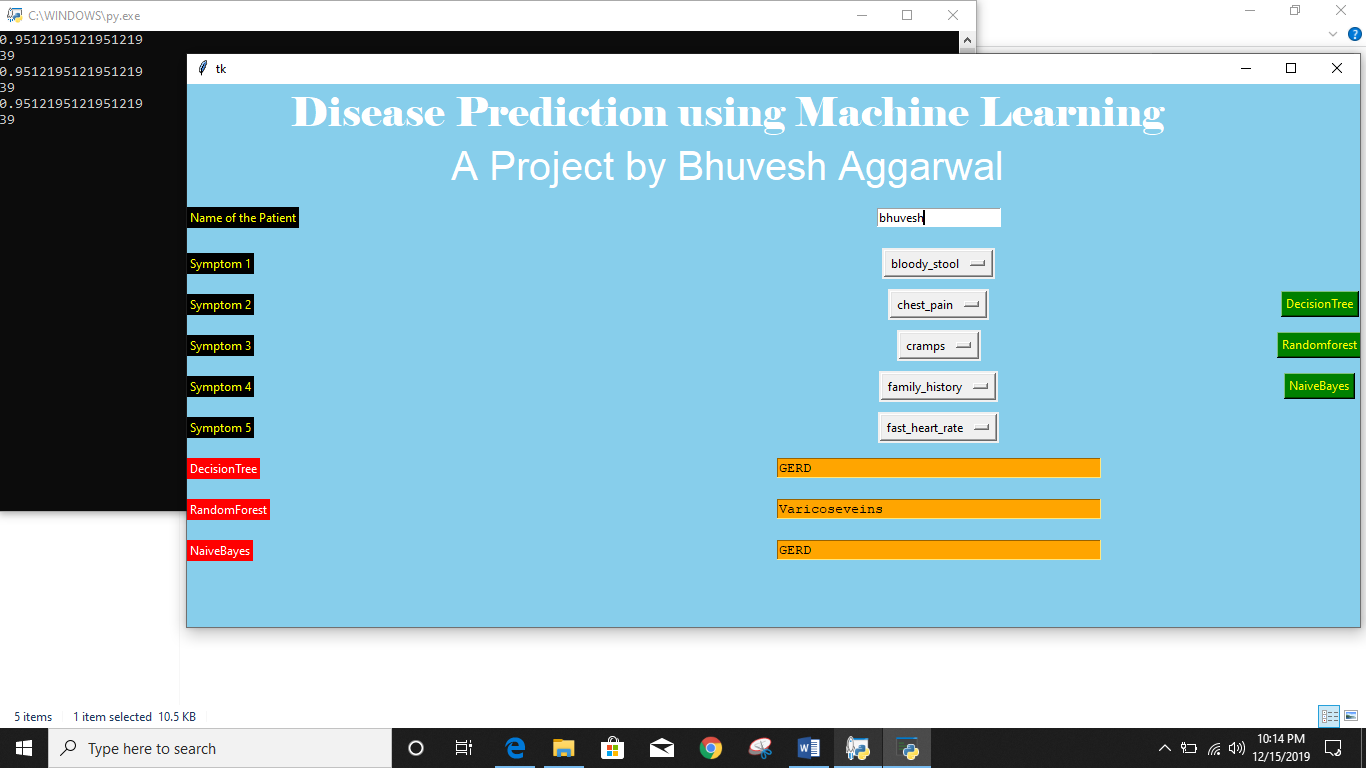
**Figure 5.1 Graphical User Interface of System**



**Figure 5.2 Graphical User Interface of System**



**Figure 5.3 Graphical User Interface of System**



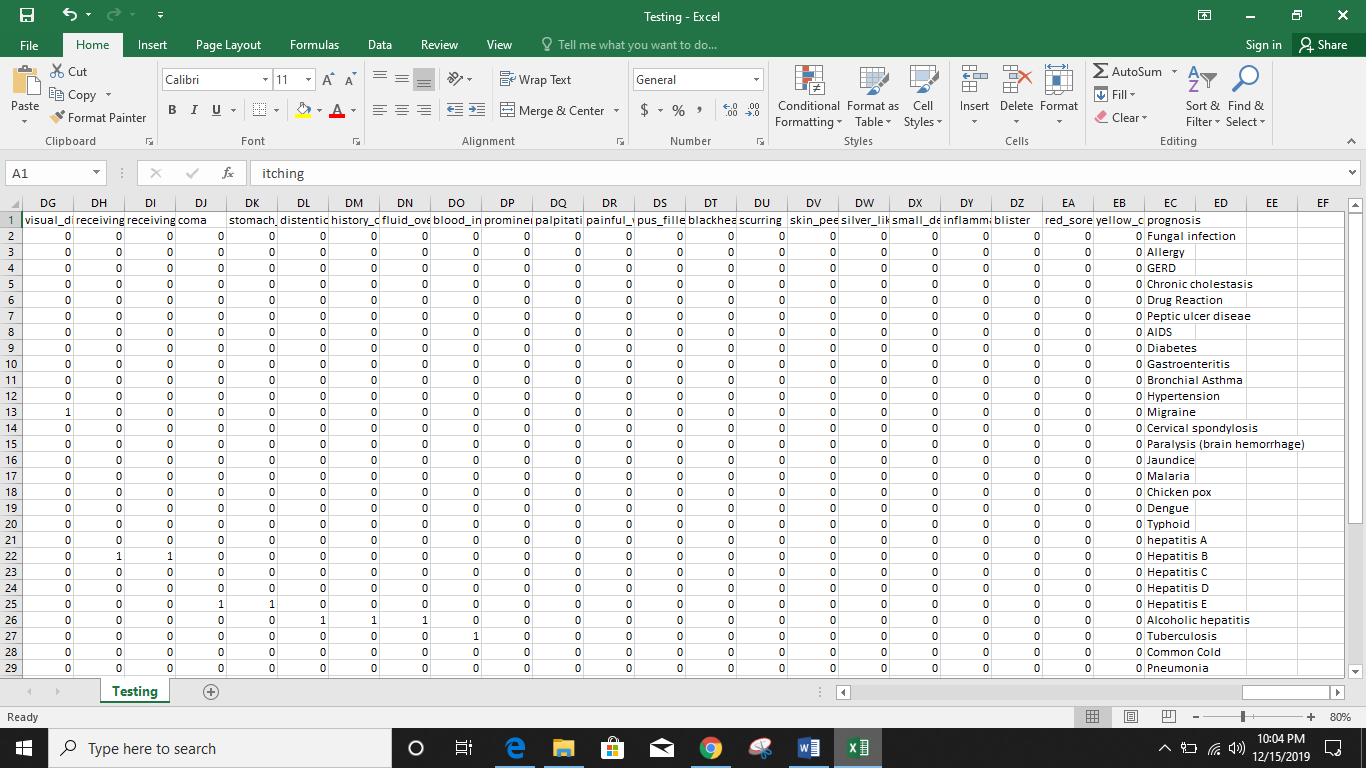
**Figure 5.4 Graphical User Interface of System**

## 

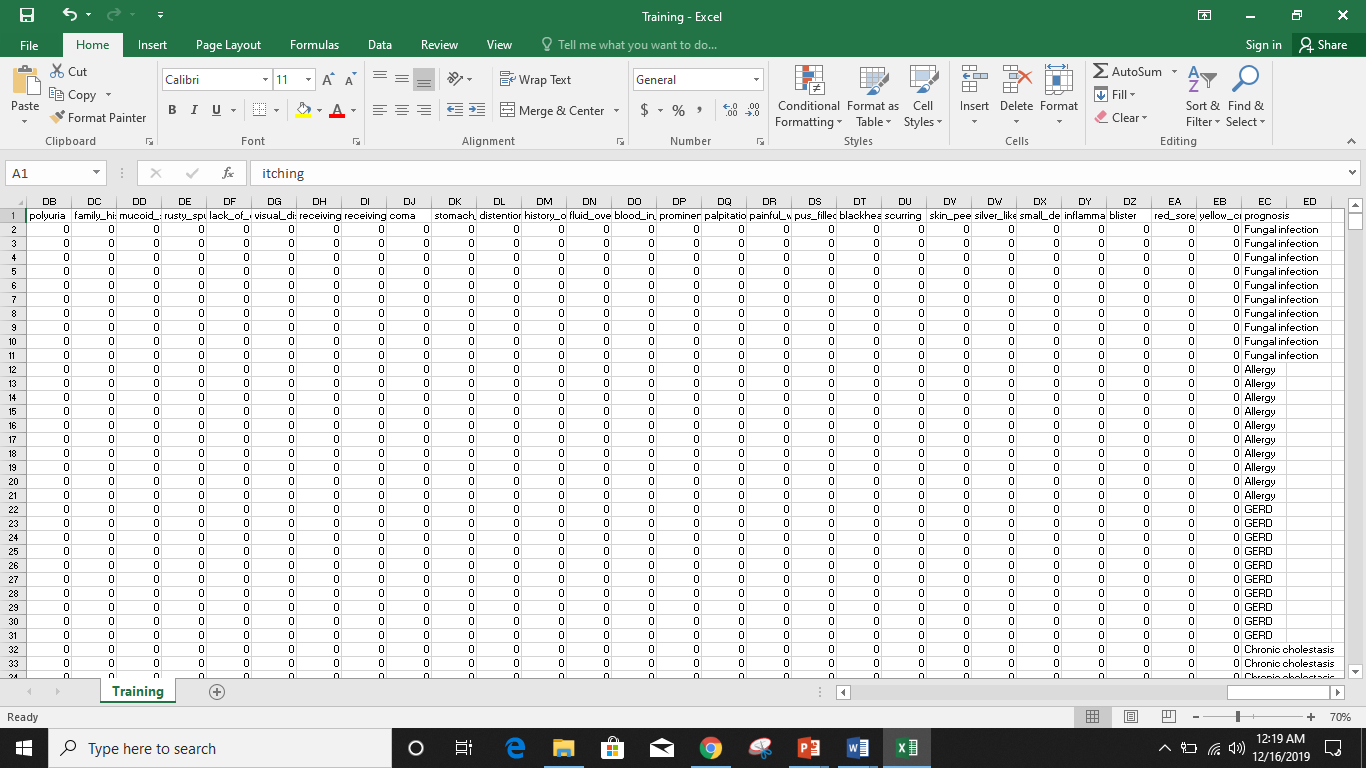
## 5.3 BACK ENDS REPRESENTATION

The database or we can say the dataset was downloaded from kaggle website named as The KAGGLE Indian Diabetic database. The extension of this file is .csv.

### 5.3.1 SNAPSHOTS OF DATABASE TABLE



**Figure 5.5 Tested data**

****

**Figure 5.6 Trained data**

# CHAPTER 6

# CONCLUSION AND FUTURE SCOPE

## 

## 6.1 CONCLUSION

Machine learning has the great ability to revolutionize the disease risk prediction with the help of advanced computational methods and availability of large amount of epidemiological and genetic disease risk dataset. Detection of disease in its early stages is the key for treatment. This work has described a machine learning approach to predicting disease levels. The technique may also help researchers to develop an accurate and effective tool that will reach at the table of clinicians to help them make better decision about the disease status.

The goals that are achieved by the software are:

* Optimum utilization of resources.
* Simplification of the operations.
* Less processing time and getting required information.
* Portable and flexible for further enhancement.
* Since it does not require internet connection, so it is more effective.

## 6.1.1 ACCURACY

We used three different algorithms to analyse the disease using various symptoms.

The algorithms are:-

* **DECISION TREE :- Accuracy calculated using following code.**

from sklearn.metrics import accuracy\_score

y\_pred=clf3.predict(X\_test)

print(accuracy\_score(y\_test, y\_pred))

print(accuracy\_score(y\_test, y\_pred,normalize=False))

* **RANDOM FOREST :-** **Accuracy calculated using following code.**

from sklearn.metrics import accuracy\_score

y\_pred=clf4.predict(X\_test)

print(accuracy\_score(y\_test, y\_pred))

print(accuracy\_score(y\_test, y\_pred,normalize=False))

* **NAÏVE BAYES :-** **Accuracy calculated using following code.**

from sklearn.metrics import accuracy\_score

y\_pred=gnb.predict(X\_test)

print(accuracy\_score(y\_test, y\_pred))

print(accuracy\_score(y\_test, y\_pred,normalize=False))

**\*Using these three algorithms the accuracy achieved is 95%**

## 6.2 FUTURE SCOPE

As no application is perfect so there are certain amendments in every application with passage of time, so there will be big need for the updates in this application also. This is a rich topic to work on and a lot of further work can be done to improve the efficiency of this project in terms of both speed and accuracy. Moreover, using deep learning, AI systems can be created that can predict the onset of disease well before a patient is diagnosed with it. As far as the code is concerned, it is running properly. There is always room for improvement in any Ml technique, however efficient the code may be. The important thing is that code should be optimal in the technique which it is used. The system has been factored into different modules to make system adaptable to further changes. ANN can be used to make it easy and more coder friendly. In future the disease attributes should be more refined and list of added symptoms may be increased more for more accurate result for prediction for the diseases. And not only cancer, heart disease, arthritis, other general diseases should also be taken care of for earlier detection.

## REFERENCES

[1] Kaveeshwar, S.A., and Cornwall, J., 2014, “The current state of disease mellitus in India”. AMJ, 7(1), pp. 45-48.

[2] Dean, L., McEntyre, J., 2004, “The Genetic Landscape of Disease [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); Chapter 1, Introduction to Disease. 2004 Jul 7.

[3] Mohammed, A.K., Sateesh, K. P., Dash G. N., 2013, “A Survey of Data Mining Techniques on Medical Data for Finding Locally Frequent Diseases” International Journal of Advanced Research in Computer Science and Software Engineering, 3(8), pp. 149-153.

[4] Chunhui, Z., Chengxia, Y., 2015, “Rapid Model Identification for Online Subcutaneous Glucose Concentration Prediction for New Subjects with Type I Disease”, IEEE Transactions on Biomedical Engineering, 62 (5), pp. 1333 – 1344

[5] Vaishali, A., Harsh, K., Anil, K.A, 2016, “Performance Analysis of the Competitive Learning Algorithms on Gaussian Data in Automatic Cluster Selection”, 2016 Second International Conference on Computational Intelligence & Communication Technology.