** A**

**Project Report**

on

**Multiple Disease Prediction**

submitted as partial fulfillment for the award of

**BACHELOR OF TECHNOLOGY**

**DEGREE**

SESSION 2023-24

in

**COMPUTER SCIENCE AND ENGINEERING**

By

Sudheer Kumar Singh (2000290100160)

Varun Saini (2000290100183)

Viraj Singh (2000290100190)

**Under the supervision of**

Prof. Dharmendra Kumar

**KIET Group of Institutions, Ghaziabad**

Affiliated to

**Dr. A.P.J. Abdul Kalam Technical University, Lucknow**

(Formerly UPTU)

**May, 2024**

**DECLARATION**

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature: Signature:

Name: Sudheer Kumar Singh Name: Varun Saini

Roll No.: 2000290100160 Roll No.: 2000290100183

Date: Date:

Signature:

Name: Viraj Singh

Roll No.: 2000290100190

Date:

## CERTIFICATE

This is to certify that Project Report entitled “**Multiple Disease Prediction**” which is submitted by **Varun Saini, Sudheer Kumar Singh and Viraj Singh** in partial fulfillment of the requirement for the award of degree B. Tech. in Department of Computer Science & Engineering of Dr. A.P.J. Abdul Kalam Technical University, Lucknow is a record of the candidates own work carried out by them under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

.

**Prof. Dharmendra Kumar Dr. Vineet Sharma**

**Assistant Professor (Head of Department)**

**Date:**

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to **Prof. Dharmendra Kumar**, Department of Computer Science & Engineering, KIET, Ghaziabad, for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of Dr. Vineet Sharma, Head of the Department of Computer Science & Engineering, KIET, Ghaziabad, for his full support and assistance during the development of the project. We also do not like to miss the opportunity to acknowledge the contribution of all the faculty members of the department for their kind assistance and cooperation during the development of our project.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members, especially **Prof. Gaurav Parashar**, of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Signature: Signature:

Name: Sudheer Kumar Singh Name: Varun Saini

Roll No.: 2000290100160 Roll No.: 2000290100183

Date: Date:

Signature:

Name: Viraj Singh

Roll No.: 2000290100190

Date:

**ABSTRACT**

There are multiple techniques in machine learning that can in a variety of industries, do predictive analytics on large amounts of data. Predictive analytics in healthcare is a difficult endeavour, but it can eventually assist practitioners in making timely decisions regarding patients' health and treatment based on massive data. Diseases like Breast cancer, diabetes, and heart-related diseases are causing many deaths globally but most of these deaths are due to the lack of timely check-ups of the diseases. The above problem occurs due to a lack of medical infrastructure and a low ratio of doctors to the population. The statistics clearly show the same, WHO recommended, the ratio of doctors to patients is 1:1000 whereas India's doctor-to population ratio is 1:1456, this indicates the shortage of doctors.

The diseases related to heart, cancer, and diabetes can cause a potential threat to mankind, if not found early. Therefore, early recognition and diagnosis of these diseases can save a lot of lives. This work is all about predicting diseases that are harmful using machine learning classification algorithms. In this work, breast cancer, heart, and diabetes are included.

To make this work seamless and usable by the mass public, our team made a medical test web application that makes predictions about various diseases using the concept of machine learning. In this work, our aim to develop a disease-predicting web app that uses the concept of machine learning-based predictions about various diseases like Breast cancer, Diabetes, and Heart diseases.

|  |  |
| --- | --- |
| **TABLE OF CONTENTS** | **Page No.** |
| DECLARATION……………………………………………………………………. | ii |
| CERTIFICATE……………………………………………………………………… | iii |
| ACKNOWLEDGEMENTS…………………………………………………………. | iv |
| ABSTRACT………………………………………………………………………..... | v |
| LIST OF FIGURES………………………………………………………………….. | viii |
| LIST OF TABLES……………………………………………………………………  LIST OF ABBREVIATIONS……………………………………………………… | ix  x |
| CHAPTER 1 (INTRODUCTION)…………………………………………………. | 1 |
| CHAPTER 2 (LITERATURE REVIEW)…………………………………………. | 3 |
| CHAPTER 3 (PROBLEM IDENTIFICATION AND DESIGN)……………………  3.1. Existing System………………………………………………………………….  3.2.Proposed System………………………………………………………………….  3.3.Feasibility………………………………………………………………………...  3.4.Requirement Analysis…………………………………………………………….  3.5.System Architecture……………………………………………………………...  3.6.UML Diagrams…………………………………………………………………...  CHAPTER 4 (IMPLEMENTATION) …..................................................................... | 7  8  12 |
| 4.1. Modules ……………………................................................................................  4.2. Technologies Used……………..………………………………………………..  4.3. Algorithms………………………………………………………………………. | 18  23  30 |
| CHAPTER 5 (TESTING AND RESULTS ) .......................................................  5.1. Types of Testing ……………………….……………..…………………………  5.2. Unit Testing ………………………...……………..…………………………….  5.3. Integration Testing ……………………………..………………………………..  5.4. User Acceptance Testing ………………………...…………………………..….  5.5. System Testing …………………………………………………………….........  5.6. Manual Testing.…………………………………………………………………. | 36    36  38  39  40  43  43 |
| CHAPTER 6 (CONCLUSION AND FUTURE SCOPE).................................. | 46 |
| 6.1. Conclusion...................................................................................................... | 46 |
| 6.2. Future Scope................................................................................................... | 47 |
|  |  |
| REFERENCES………………………………………………………………………. | 48 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Description** | **Page No.** |

Fig.3.1 System Architecture 12

FIG.3.6 Class Diagram 13

FIG.3.6 Use Case Diagram 14

FIG.3.6 Sequence Diagram 15

FIG.3.6 Component Diagram 16

FIG.3.6 Deployment Diagram 17

FIG.4.3 K Value Graph 32

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table. No.** | **Description** | **Page No.** |
| 2.1  3.1 | Findings and limitations of papers  Table depicting the number of components used. | 9  13 |

**LIST OF ABBREVIATIONS**

**ANN -** Artificial Neural Network

**SRS** - Software Requirements Specification

**STRS** - Stakeholder Requirements Specification

**UML -** Unified Modellling Language

**SRS** - System Requirements Specification

**ADT-** Automatic Drum Transcription

**AM-** Activation Maximization

**AUC-** Area Under the (ROC) Curve

**BLEU-** Bilingual Evaluation Understudy

**BiFPN-** Bidirectional Feature Pyramid Network

**CHAPTER 1**

**INTRODUCTION**

Multiple disease prediction using machine learning is an innovative approach to healthcare that aims to use machine learning algorithms to accurately predict the likelihood of multiple diseases in a patient based on their medical history and other relevant factors.

The goal of this approach is to enable earlier diagnosis, better treatment, and improved patient outcomes. Machine learning algorithms are particularly well-suited to the task of disease prediction, as they can learn from large datasets of patient information and identify patterns and correlations that might not be immediately apparent to human clinicians.

By analyzing data from a wide range of sources, including electronic health records, medical images, and genetic data, machine learning algorithms can identify subtle indicators of disease that might be missed by traditional diagnostic methods.

Multiple disease prediction using machine learning has the potential to revolutionize healthcare by enabling more accurate and personalized diagnoses, earlier interventions, and more effective treatments.

However, there are also challenges and limitations to this approach, including the need for diverse and representative data, the risk of bias in algorithms, and the need for transparent and ethical implementation.

Despite these challenges, multiple disease prediction using machine learning is a rapidly advancing field that holds great promise for the future of healthcare. As technology continues to evolve and more data becomes available, it is likely that machine learning algorithms will become increasingly sophisticated and accurate, leading to improved patient outcomes and better overall health.

Machine learning (ML) is one of the most rapidly developing fields of computer science, with several applications. It refers to the process of extracting useful information from a large set of data. ML techniques are used in different areas such as medical diagnosis, marketing, industry, and other scientific fields. ML algorithms have been widely used in medical datasets and are best suited for medical data analysis.

There are various forms of ML, including classification, regression, and clustering. , we focus on classification methods, which are applied to classify a given dataset into predefined groups and to predict future activities or information to that data due to its good accuracy and performance.

**CHAPTER 2**

**LITERATURE REVIEW**

Anila M and Dr G Pradeepini proposed the paper titled “Diagnosis of Parkinson’s disease using Artificial Neural network” .

The main objective of this paper is that the detection of the disease is performed by using the voice analysis of the people affected with Parkinson's disease. For this purpose, various machine learning techniques like ANN, Random Forest, KNN, SVM, XG Boost are used to classify the best model, error rates are calculated, and the performance metrics are evaluated for all the models used. The main drawback of this paper is that it is limited to ANN with only two hidden layers. And this type of neural networks with two hidden layers are sufficient and efficient for simple datasets. They used only one technique for feature selection which reduces the number of features. Arvind Kumar Tiwari Proposed the paper titled “Machine Learning-based Approaches for Prediction of Parkinson’s Disease” .

In this paper, minimum redundancy maximum relevance feature selection algorithms were used to select the most important feature among all the features to predict Parkinson diseases. Here, it was observed that the random forest with 20 number of features selected by minimum redundancy maximum relevance feature selection algorithms provide the overall accuracy 90.3%, precision 90.2%, Mathews correlation coefficient values of 0.73 and ROC values 0.96 which is better in comparison to all other machine learning based approaches such as bagging, boosting, random forest, rotation forest, random subspace, support vector machine, multilayer perceptron, and decision tree based methods. Afzal Hussain Shahid and Maheshwari Prasad Singh proposed the paper titled “A deep learning approach for prediction of Parkinson’s disease progression”

. This paper proposed a deep neural network (DNN) model using the reduced input feature space of Parkinson’s telemonitoring dataset to predict Parkinson’s disease (PD) progression and also proposed a PCA based DNN model for the prediction of Motor-UPDRS and Total-UPDRS in Parkinson's Disease progression. The DNN model was evaluated on a real-world PD dataset taken from UCI. Being a DNN model, the performance of the proposed model may improve with the addition of more data points in the datasets Sarwar and al ,

discuss predictive analytics in healthcare, a number of machine learning algorithms are used in this study. For experiment purposes, a dataset of patient's medical is obtained. The performance and accuracy of the applied algorithms are discussed and compared. In the paper

the authors propose a diabetes prediction model for the classification of diabetes including external factors responsible for diabetes along with regular factors like Glucose, BMI, Age, Insulin, etc. Classification accuracy is improved with the novel dataset compared with existing dataset. On a dataset of 521 instances (80 % and 20 % for training testing respectively),

Authors applied 8 ML algorithms such as logistic regression, support vector machines-linear, and nonlinear kernel, random forest, decision tree, adaptive boosting classifier, K-nearest neighbor, and naïve bayes. According to the results, the Random Forest classifier achieved 98 % accuracy compared to the other. In [9], the researchers used machine-learning algorithms including Logistic Regression, Gaussian Process, Adaptive Boosting (AdaBoost), Decision Tree, K-Nearest Neighbors, Multilayer Perceptron, Support Vector Machine, Bernoulli Naive Bayes, Bagging Classifier, Random Forest, and Quadratic Discriminant Analysis.

The Random Forest classifier performs better and achieved a 98 % accuracy, which is higher than the other three algorithms. Aditi Gavhane, Gouthami Kokkula, Isha Panday, Prof. Kailash Devadkar, “Prediction of Heart Disease using Machine Learning” Gavhane et al.

Have worked on the multi-layer perceptron model for the prediction of heart diseases in human being and the accuracy of the algorithm using CAD technology. If the number of person using the prediction system for their diseases prediction, then the awareness about the diseases is also going to increases and it make reduction in the death rate of heart patient. Pahulpreet Singh Kohli and Shriya Arora, “Application of Machine Learning in Diseases Prediction Machine learning algorithms are used for various type of diseases predication and many of the researchers have work on this like Kohali et al.

Work on heart diseases prediction using logistic regression, diabetes prediction using support vector machine, breast cancer prediction using Adaboot classifier and concluded that the logistic regression give the accuracy of 87.1%, support vector machine give the accuracy of 85.71%, Adaboot classifier give the accuracy up to 98.57%which good for predication point of view. In another way, the authors of the paper have built models to predict and classify diabetes complications. In this work, several supervised classification algorithms were applied to predict and classify 8 diabetes complications. The complications include some parameters such as metabolic syndrome, dyslipidemia, nephropathy, diabetic foot, obesity, and retinopathy.

The data mining techniques is a more popular in many field of medical, business, railway, education etc. They are most commonly used for medical diagnosis and disease prediction at the early stage. The data mining is utilized for healthcare sector in industrial societies. This paper to provide a survey of data mining techniques of using Parkinson’s disease.

Parkinson disease is a global public health issue. Machine learning technique would be a best solution to classify individuals and individuals with Parkinson's sickness (PD). This paper gives a complete review for the forecast of Parkinson disease by utilizing the machine learning based methodologies. A concise presentation of different computational system based methodologies utilized for the forecast of Parkinson disease are introduced. This paper likewise displays the outline of results acquired by different scientists from accessible information to predict the Parkinson disease.

In this experimental analysis four machine learning algorithms, Random Forest, Knearest neighbor, Support Vector Machine, and Linear Discriminant Analysis are used in the predictive analysis of early-stage diabetes. High accuracy of 87.66 % goes to the Random Forest classifier. In another way, the authors of the paper have built models to predict and classify diabetes complications. In this work, several supervised classification algorithms were applied to predict and classify 8 diabetes complications. The complications include some parameters such as metabolic syndrome, dyslipidemia, nephropathy, diabetic foot, obesity, and retinopathy.

In , the authors present two approaches of machine learning to predict diabetes patients. Random forest algorithm for the classification approach, and XGBoost algorithm for a hybrid approach. The results show that XGBoost outperforms in terms of an accuracy rate of 74.10%. Authors in this article [15] tested machine learning algorithms such as support vector machine, logistic regression, Decision Tree, Random Forest, gradient boost, K-nearest neighbor, Naïve Bayes algorithm. According to the results, Naïve Base and Random Forest classifiers achieved 80% accuracy compared to the other algorithms

**CHAPTER 3**

**PROBLEM IDENTIFICATION AND DESIGN**

Many of the existing machine learning models for health care analysis are concentrating on one disease per analysis. For example first is for liver analysis, one for cancer analysis, one for lung diseases like that. If a user wants to predict more than one disease, he/she has to go through different sites. There is no common system where one analysis can perform more than one disease prediction. Some of the models have lower accuracy which can seriously affect patients’ health. When an organization wants to analyse their patient’s health reports, they have to deploy many models which in turn increases the cost as well as time Some of the existing systems consider very few parameters which can yield false results.

**3.1 EXISTING SYSTEM**

The study has identified multiple risk factors for cardiovascular disease, including high blood pressure, high cholesterol, smoking, and diabetes. Based on these risk factors, a risk score can be calculated to predict an individual's likelihood of developing cardiovascular disease. Traditional statistical methods are used to identify risk factors and calculate a risk score, which

can be used for disease prevention and management. One of the biggest concerns with machine learning systems is data bias. If the training data used to develop the system is biased or incomplete, it can lead to inaccurate predictions and misdiagnosis. This is especially problematic when it comes to underrepresented populations, as their data may not be well-represented in the training set. Overfitting occurs when a machine learning model is trained too closely to a particular dataset and becomes overly specialized in predicting it. This can result in poor generalization to new data and lower accuracy. Many machine learning algorithms are "black boxes," meaning that it is difficult to understand how they arrive at their predictions. This can be problematic in healthcare, where it is important to be able to explain how a diagnosis was made. Limited data availability: Some diseases are rare, which means that there may not be enough data available to train a machine learning model accurately. This can limit the effectiveness of the system for predicting such diseases. Cost and implementation: Implementing machine learning systems for healthcare can be expensive and time-consuming. Hospitals and clinics may need to invest in new hardware, software, and staff training to implement these systems effectively.

**3.2 PROPOSED SYSTEM**

This project involved analyzing a multiple disease patient dataset with proper data processing. Different algorithms were used to train and predict, including Decision Trees, Random Forest, SVM, and Logistic Regression,adaboost. In a multi-disease model, it is possible to predict more than one disease at a time, reducing the need to traverse multiple models to predict disease. Diverse training data: To address data bias, a proposed system would use a diverse range of training data, including data from underrepresented populations, to ensure that the system can accurately predict diseases across all groups. Robust algorithms: The system would use algorithms that are robust to overfitting and have high accuracy on unseen data. This could be achieved by using techniques such as regularization and cross-validation. Explainable AI: To address the lack of interpretability of machine learning models, the proposed system would use explainable AI techniques to provide clear and understandable reasons for its predictions. This would increase the trust and acceptance of the system among healthcare providers and patients.

**3.3 FEASIBILITY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with 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. Multiple Disease prediction using machine learning 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 • Economical Feasibility • Technical Feasibility • Social Feasibility ECONOMICAL FEASIBILITY This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased. TECHNICAL FEASIBILITY During this study, the analyst identifies the existing computer systems of the concerned department and determines whether these technical resources are sufficient for the proposed system or not. If they are not sufficient, the analyst suggests the configuration of the computer systems that are required. The analyst generally pursues two or three different configurations which satisfy the key technical requirements but which represent different costs. During technical feasibility study, financial resources and budget is also considered. The main objective of technical feasibility is to determine whether the project is technically feasible or not, provided it is economically feasible. SOCIAL FEASIBILITY The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system. Multiple Disease prediction using machine learning

**3.4 REQUIREMENTS**

A software requirements specification (SRS) is a description of a software system to be developed, its defined after business requirements specification (CONOPS) also called stakeholder requirements specification (STRS) other document related is the system requirements specification (SYRS). HARDWARE AND SOFTWARE REQUIREMENTS All computer software needs certain hardware components or other software resources to be present on a computer. These prerequisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements. A second meaning of the term of System requirements is a generalization of this first definition, giving the requirements to be met in the design of a system or sub-system.

**3.5 SYSTEM DESIGN**

This chapter provides information of software development life cycle, design model i.e.various UML diagrams and process specification. Systems design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap and synergy with the disciplines of systems analysis, systems architecture and systems engineering. The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces. This design activity describes the system in narrative form using non-technical terms. It should provide a high-level system architecture diagram showing a subsystem breakout of the system, if applicable. The high-level system architecture or subsystem diagrams should, if applicable, show interfaces to external systems. Supply a high-level context diagram for the system and subsystems, if applicable. Refer to the requirements trace ability matrix (RTM) in the Functional Requirements Document (FRD), to identify the allocation of the functional requirements into this design document. This section describes any constraints in the system design (reference any trade-off analyses conducted such, as resource use versus productivity, or conflicts with other systems) and includesany assumptions made by the project team in developing the system design. This section describes any contingencies that might arise in the design of the system that may change the development direction. Possibilities include lack of interface agreements with outside agencies or unstable architectures at the time this document is produced. Address any possible workarounds or alternative plans. Multiple Disease prediction using machine learning To design a system for Multiple Disease prediction based on lab reports using machine learning, we can follow the following steps:

1. Data Collection: The first component of the system involves collecting a large dataset of medical records containing patient information and various medical features related to multiple diseases. This dataset will be used to train the machine learning models.

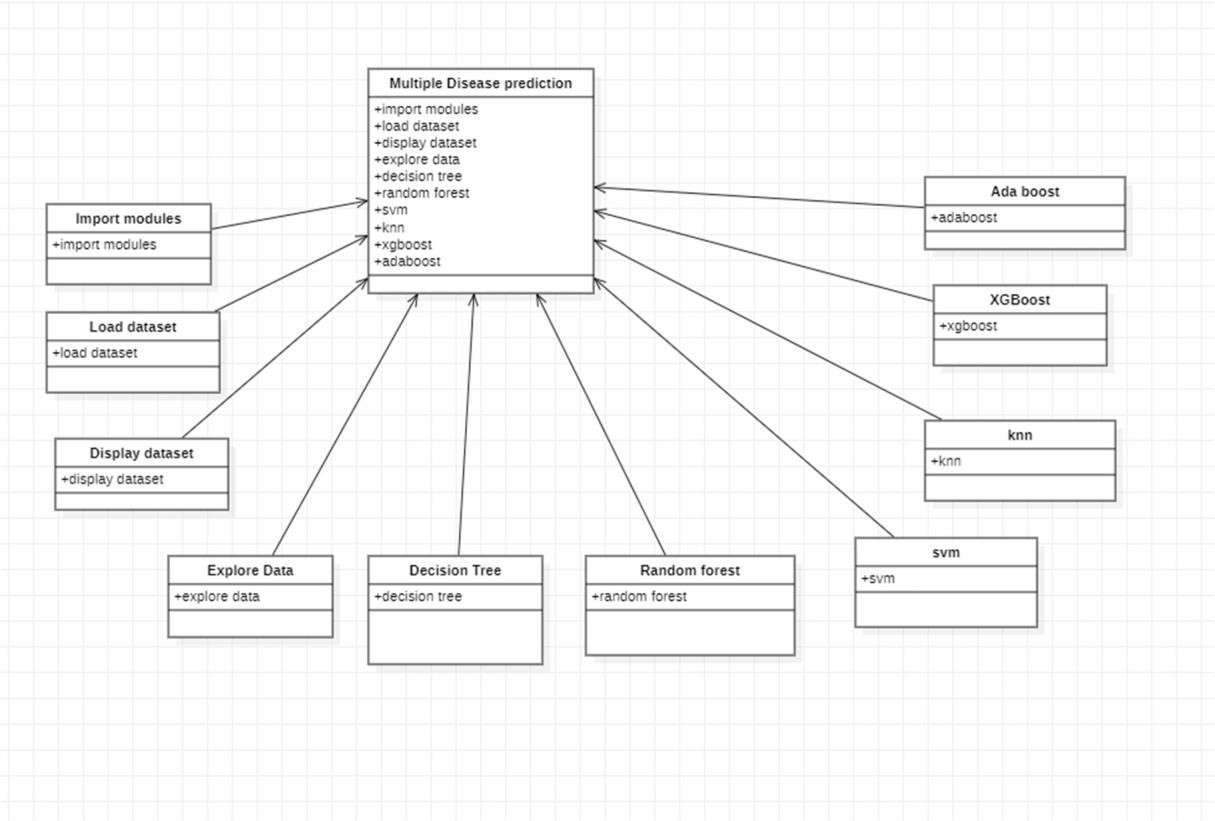
2. Data Preprocessing: The collected data will be preprocessed to handle missing values, outliers, and to perform feature scaling. This component of the system involves cleaning and preparing the data for model training.

3. Model Training: This component involves training different machine learning algorithms such as decision trees, random forests, and artificial neural networks on the preprocessed data. The trained models will be used for disease prediction.

4. Model Selection: The performance of different machine learning algorithms will be compared using metrics such as accuracy, precision, and recall, and the bestperforming model will be selected for disease prediction.

5. Model Evaluation: The selected model will be evaluated on a separate test dataset to measure its accuracy and reliability in predicting multiple diseases. This component of the system involves testing the model and measuring its performance.

6. User Interface Development: The final component of the system involves developing a user-friendly interface that allows healthcare professionals to input patient information and receive predictions for multiple diseases. The interface will be designed to provide an easy-to-use tool for disease prediction .A system architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system. A system architecture can consist of system components and the sub-systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture, collectively these are called architecture description languages. Machine learning has given computer systems the ability to automatically learn without being explicitly programmed. In this, the author has used three machine learning algorithms (Logistic Regression, KNN, and Naïve Bayes). The architecture diagram describes the highlevel overview of major system components and important working relationships.



##### 3.6 UML DIAGRAM

A UML diagram is a partial graphical representation (view) of a model of a system under design, implementation, or already in existence. UML diagram contains graphical elements (symbols) - UML nodes connected with edges (also known as paths or flows) - that represent elements in the UML model of the designed system. The UML model of the system might also contain other documentation such as use cases written as templated texts. The kind of the diagram is defined by the primary graphical symbols shown on the diagram.

For example, a diagram where the primary symbols in the contents area are classes is class diagram. A diagram which shows use cases and actors is use case diagram. A sequence diagram shows sequence of message exchanges between lifelines. UML specification does not preclude mixing of different kinds of diagrams, e.g. to combine structural and behavioral elements to show a state machine nested inside a use case. Consequently, the boundaries between the various kinds of diagrams are not strictly enforced. At the same time, some UML Tools do restrict set of available graphical elements which could be used when working on specific type of diagram. UML specification defines two major kinds of UML diagram: structure diagrams and behavior diagrams.

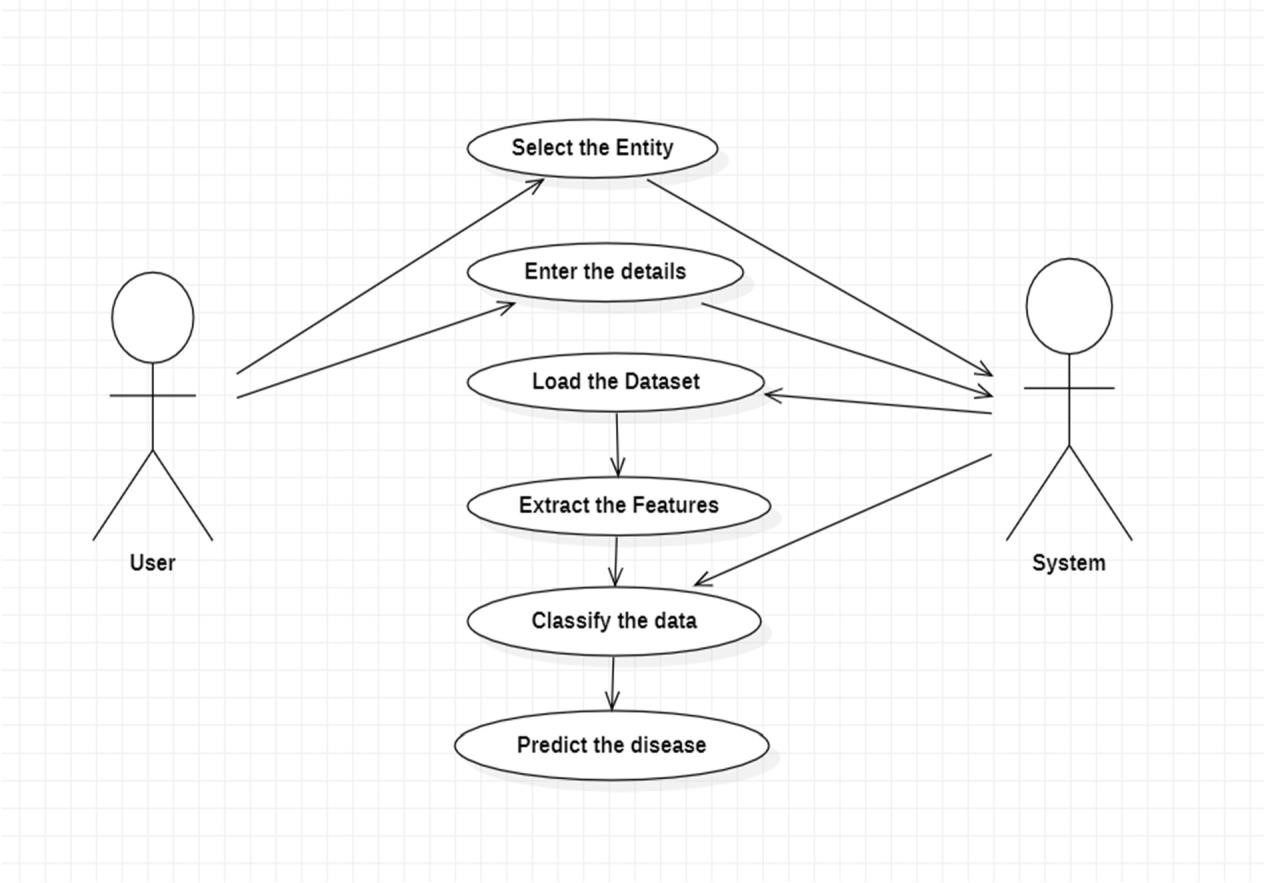
Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts. Behavior diagrams show the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time.

During detailed design the internal logic of each of the modules specified in system design is decided. During this phase further details of the data structures and algorithmic design of each of the modules is specified. The logic of a module is usually specified in a high-level design description language, which is independent of the target language in which the application will eventually be implemented. In system design the focus is on identifying the modules, whereas during detailed design the focus is on designing the logic for each of the modules.

A Class diagram gives an overview of a system by showing its classes and the relationships among them. Class diagrams are static they display what interacts but not what happens when they do interact. The class chart delineates the attributes and operations of a class moreover the goals constrained on the structure.

The class frameworks are extensively used as a piece of the showing of article arranged structures in light of the way that they are the primary UML diagrams which can be mapped direct with thing orchestrated vernaculars. The class graph shows a collection of classes, interfaces, affiliations, joint endeavors and confinements. It is generally called an assistant layout.

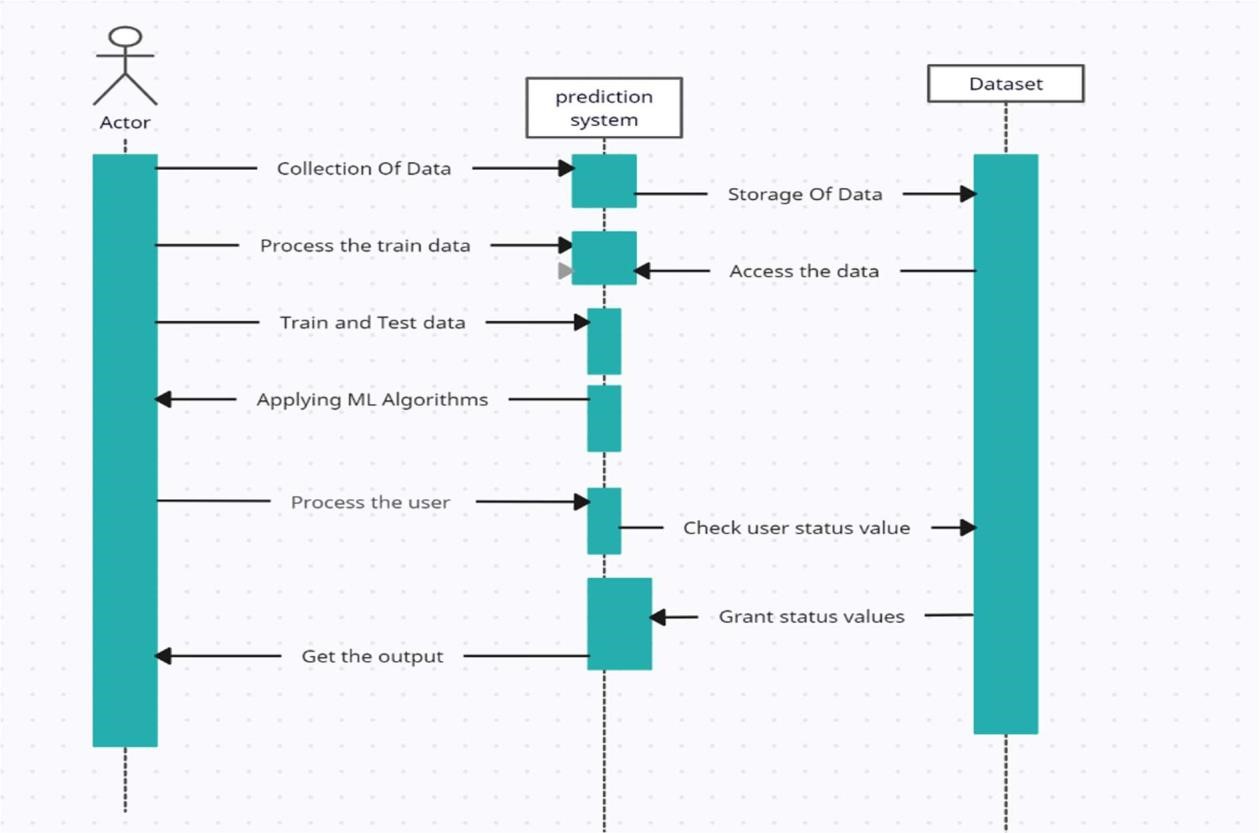
Use case diagrams model behavior within a system and helps the developers understand of what the user require. Use case diagram can be useful for getting an overall view of the system and clarifying who can do and more importantly what they can’t do. Use case diagram consists of use cases and actors and shows the interaction between the use case and actors.



Above figure use case diagram consists of two actors named as user and system. User can perform actions like select the Entity and Enter the details. System perform actions select the entity means select the disease and enter the patient details then load the data set and classify the data finally predict the disease.

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

One of the primary uses of sequence diagrams is in the transition from requirements expressed as use cases to the next and more formal level of refinement. Use cases are often refined into one or more sequence diagrams.



From sequence diagram the prediction system can collect the data from actor and store the data in dataset.Prediction system processes the train data and access the data from dataset then prediction system use the train and test data and apply ML algorithms and check user status value and grand status values then get the output.

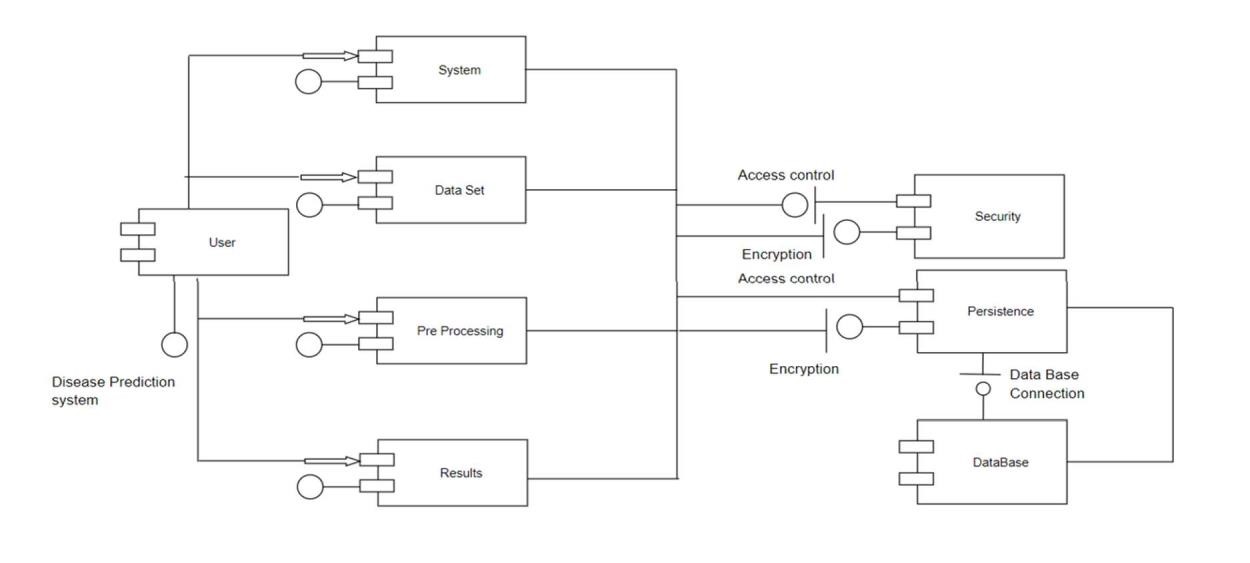
A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces.

This diagram is also used as a communication tool between the developer and stakeholders of the system. Programmers and developers use the diagrams to formalize a roadmap for the implementation, allowing for better decision-making about task assignment or needed skill improvements.

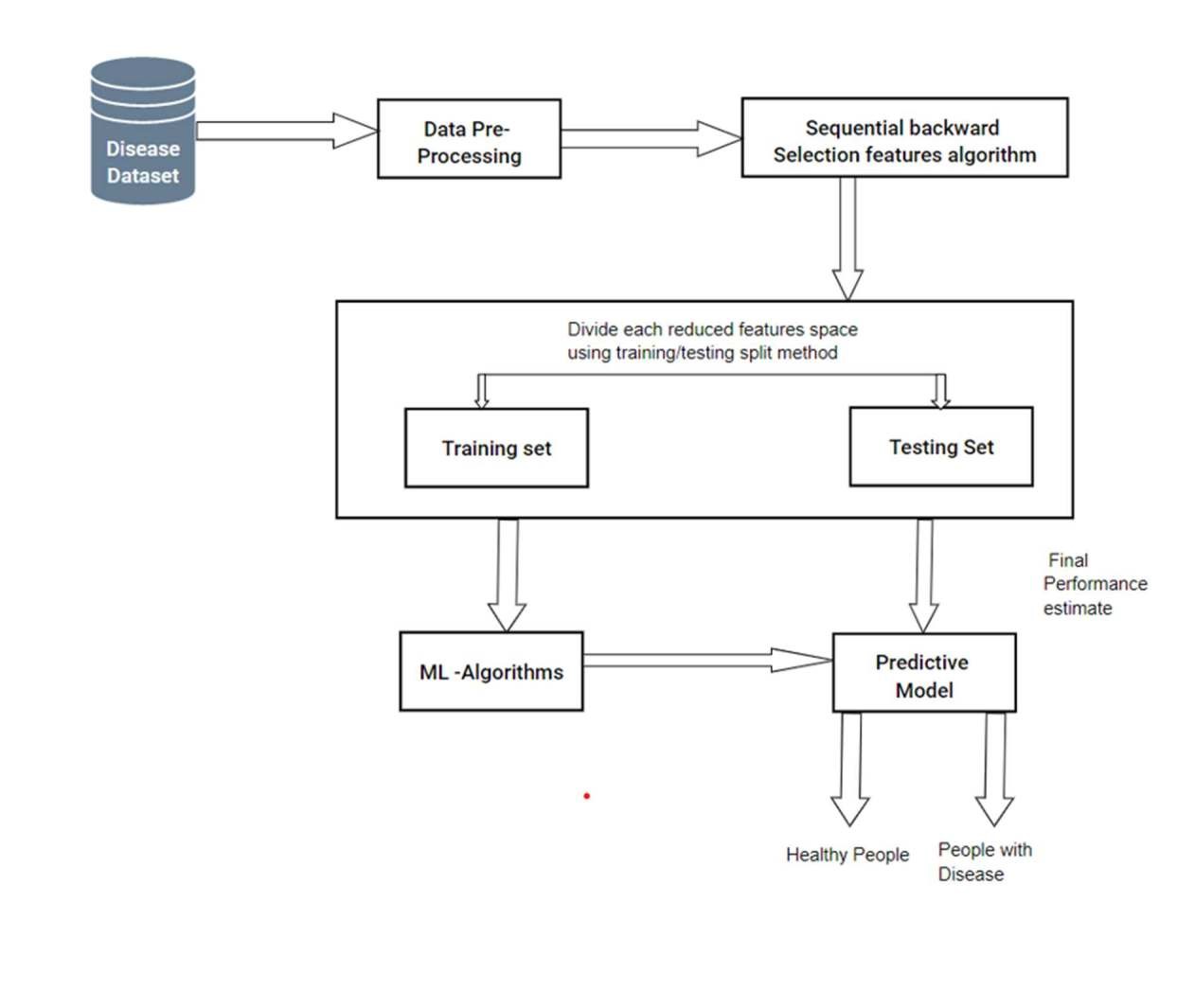
System administrators can use component diagrams to plan ahead, using the view of the logical software components and their relationship on system.

.



From the above diagram component diagram has components like user,system,data set,pre processing,results,security, persistence and data base these are tha components of Multiple Disease prediction system.

The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships. It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths



A deployment diagram for multiple disease prediction includes components such as disease dataset,data preprocessing, , Ml algorithms, predictive model . The user interface collects input data from disease dataset and processes using Ml algorithms and then predict the disease using predict model.

**CHAPTER 4**

**IMPLEMENTATION**

An Implementation is a realization of a technical specification or algorithm as a program, software components, or other computer system though computer programming and deployment. Many implementations may exist for specifications or standards. A special case occurs in object- oriented programming, when a concrete class implements an interface.

First step for predication system is data collection and deciding about the training and testing dataset. In this project we have used training dataset and testing dataset.

Attribute of dataset are property of dataset which are used for system and for heart many attributes are like heart bit rate of person, gender, sex of the person, age of the person and many more for predication system.

Pre processing needed for achieving prestigious result from the machine learning algorithms. For example Random forest algorithm does not support null values dataset and for this we have to manage null values from original raw data.

For our project we have to convert some categorized value by dummy value means in the form of “0”and “1” by using following code. Imbalanced datasets can be balanced in two ways. They are Under Sampling and Over Sampling.

Dataset balance is done by the reduction of the size of the data set. This process is considered when the amount of data is adequate.

In Over Sampling, dataset balance is done by increasing the size of the dataset. This process is considered when the amount of data is inadequate.

**4.1 MODULES**

1.Parkinson’s Disease

The Parkinson Disease prediction module is one of the core of a multiple Disease

prediction system. It uses data about the Effected and normal people data preferences to

generate the result of the patient.

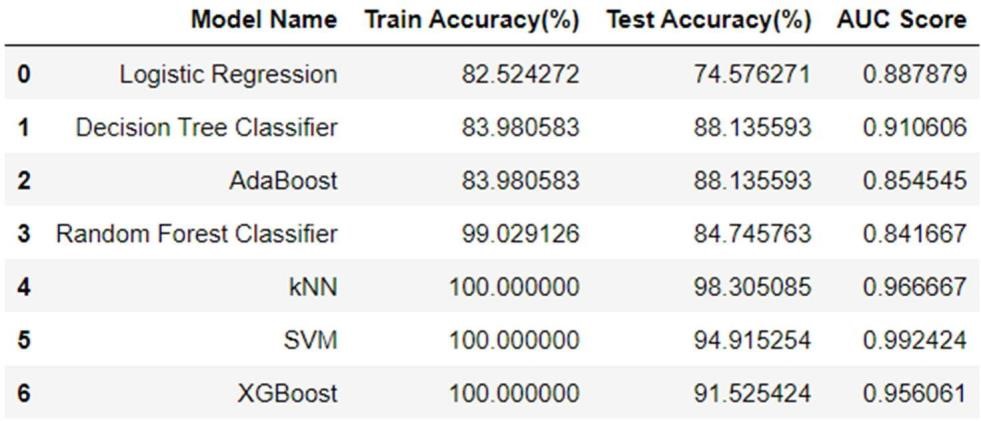
Attribute Information:

1. Name - ASCII subject name and recording number.
2. MDVP:Fo(Hz)- Average vocal fundamental frequency.
3. MDVP:Fhi(Hz) - Maximum vocal fundamental frequency.
4. MDVP:Flo(Hz) - Minimum vocal fundamental frequency.
5. MDVP:Jitter(%), MDVP:Jitter(Abs), MDVP:RAP, MDVP:PPQ, Jitter:DDP - Several measures of variation in fundamental frequency.
6. MDVP:Shimmer,MDVP:Shimmer(dB),Shimmer:APQ3,Shimmer:APQ5,MDVP:APQ,

Shimmer:DDA - Several measures of variation in amplitude.

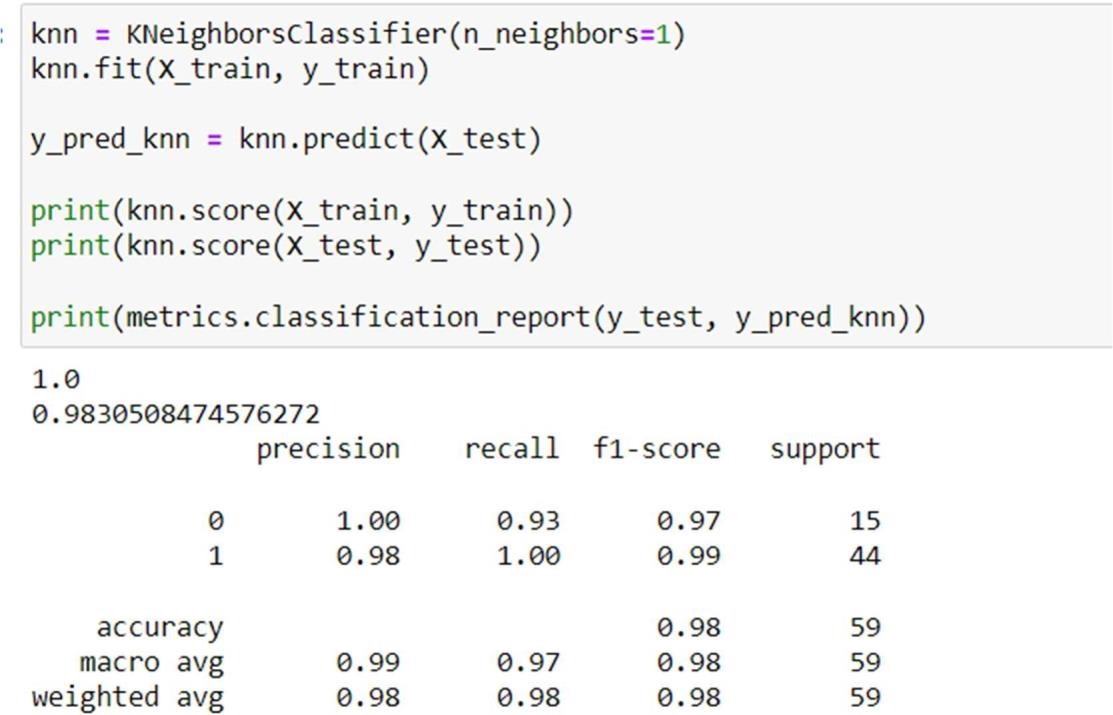
1. NHR, HNR- Two measures of the ratio of noise to tonal components in the voice.
2. status - The health status of the subject (one) - Parkinson's, (zero) – healthy.
3. RPDE, D2- Two nonlinear dynamical complexity measures.
4. DFA - Signal fractal scaling exponent.
5. spread1,spread2,PPE - Three nonlinear measures of fundamental frequency variation.

Comparison of Models



* We can say that kNN Model is good for our dataset but SVM giving more AUC.
* The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

Classification Report



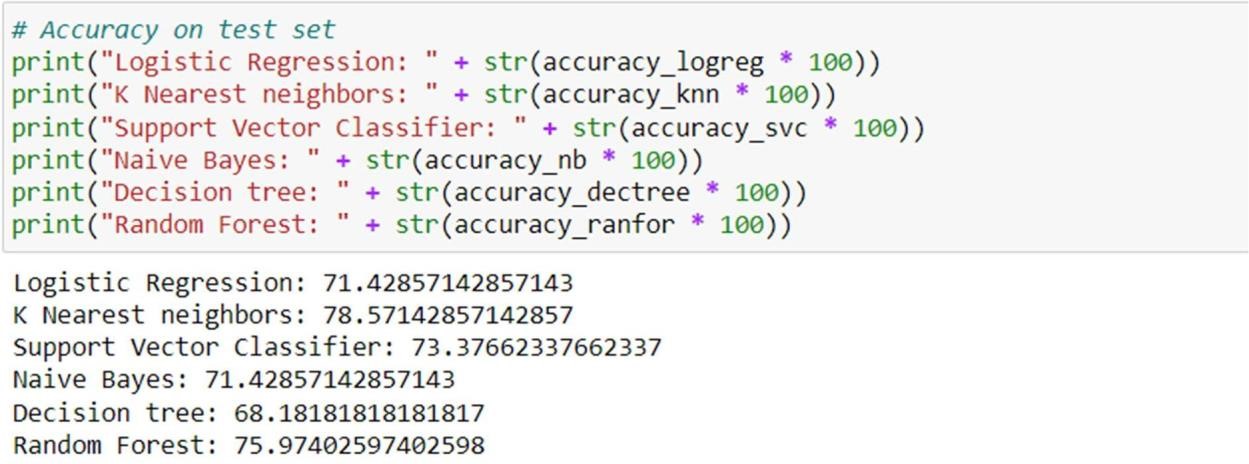
2. Diabetes disease prediction

* The aim of the prediction is which can perform early prediction of diabetes of a patient.
* This aims to predict via different supervised machine learning methods.
* It uses data about the Effected and normal people data preferences to generate Whether person is effected or not from a particular Disease.

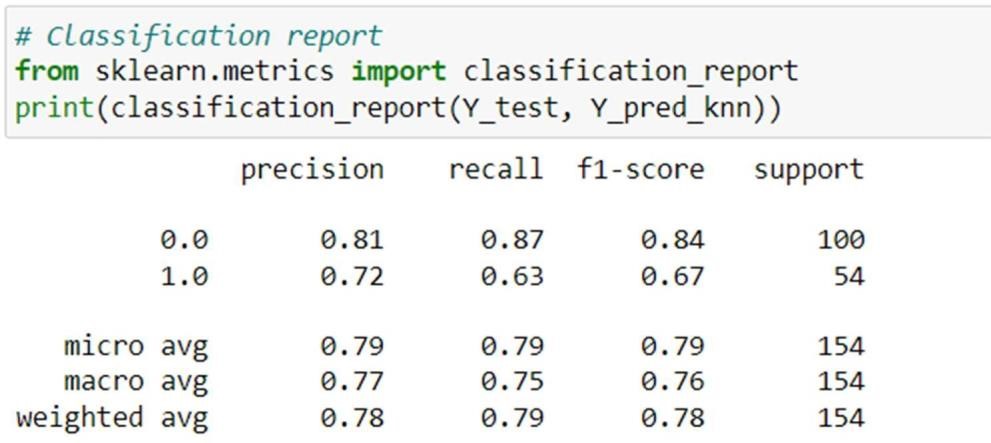
Attribute Information

1. Pregnancies
2. Glucose
3. Blood pressure
4. SkinThickness
5. Insulin
6. BMI
7. DiabetesPedigreeFunction
8. Age

Comparison of Models



Classification Report

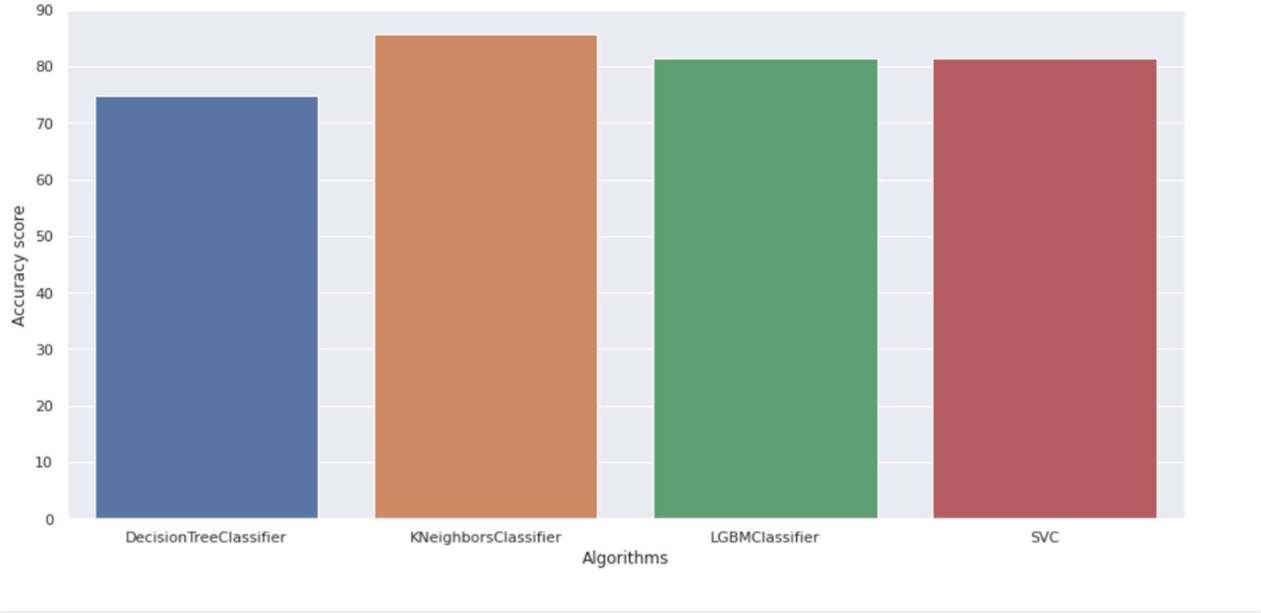


3. Heart disease prediction

* It uses data about the Effected and normal people data preferences to generate the result of the patient.
* It performs the Different machine algorithms like KNN, XGBoost, SVM, RANDOM FOREST,etc.
* This aims to predict via different supervised machine learning methods.
* Attribute Information

1. Age
2. Sex
3. Chest Pain types
4. Resting blood pressure
5. Serum cholestral
6. Fasting Blood sugar
7. Resting Cardiographic Result
8. Maximum Heart rate achieved
9. Exercise Reduced Angina
10. Vessels coloured by Fluroscopy

Accuracy result:



**4..2 TECHNOLOGIES USED**

##### PYTHON

Python is a high-level, general-purpose and a very popular programming language.

Python programming language (latest Python 3) is being used in web development, Machine

Learning applications, along with all cutting edge technology in Software Industry. Python Programming Language is very well suited for Beginners, also for experienced programmers with other programming languages like C++ and Java.

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object- oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural,) object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

Advantages of python

1. Easy to read, learn and code

Python is a high-level language and its syntax is very simple. It does not need any semicolons or braces and looks like English. Thus, it is beginner-friendly. Due to its simplicity, its maintenance cost is less.

2. Dynamic Typing

In Python, there is no need for the declaration of variables. The data type of the variable gets assigned automatically during runtime, facilitating dynamic coding.

3. Free, Open Source

It is free and also has an open-source licence. This means the source code is available to the public for free and one can do modifications to the original code. This modified code can be distributed with no restrictions.

This is a very useful feature that helps companies or people to modify according to their needs and use their version.

4. Portable :

Python is also platform-independent. That is, if you write the code on one of the Windows, Mac, or Linux operating systems, then you can run the same code on the other OS with no need for any changes.

This is called Write Once Run Anywhere (WORA). However, you should be careful while you add system dependent features.

5. Extensive Third-Party Libraries

Python comes with a wide range of libraries like NumPy, Pandas, Tkinter, Django, etc.

The python package installer (PIP) helps you install these libraries in your interpreter/ IDLE.

These libraries have different modules/ packages. These modules contain different inbuilt functions and algorithms. Using these make the coding process easier and makes it look simple.

STREAMLIT

Streamlit is an open-source python framework for building web apps for Machine Learning and Data Science. We can instantly develop web apps and deploy them easily using Streamlit. Streamlit allows you to write an app the same way you write a python code. Streamlit makes it seamless to work on the interactive loop of coding and viewing results in the web app.

The best thing about Streamlit is that you don't even need to know the basics of web development to get started or to create your first web application. So if you're somebody who's into data science and you want to deploy your models easily, quickly, and with only a few lines of code, Streamlit is a good fit.

One of the important aspects of making an application successful is to deliver it with an effective and intuitive user interface. Many of the modern data-heavy apps face the challenge of building an effective user interface quickly, without taking complicated steps. Streamlit is a promising open-source Python library, which enables developers to build attractive user interfaces in no time. Streamlit is the easiest way especially for people with no front-end knowledge to put their code into a web application:

* No front-end (html, js, css) experience or knowledge is required.

* You don't need to spend days or months to create a web app, you can create a really beautiful machine learning or data science app in only a few hours or even minutes.
* It is compatible with the majority of Python libraries (e.g. pandas, matplotlib, seaborn, plotly, Keras, PyTorch, SymPy(latex)).
* Less code is needed to create amazing web apps.

* Data caching simplifies and speeds up computation pipelines.

Streamlit is a popular open-source Python library that allows developers to build interactive web applications for data science and machine learning projects with ease. Here are some of the key features of Streamlit:

1. Ease of Use: Streamlit is easy to use for both beginners and advanced developers. Its simple syntax allows developers to build interactive web applications quickly without having to worry about the details of web development.
2. Data Visualization: Streamlit allows developers to create data visualizations such as charts, plots, and graphs with just a few lines of code. It supports popular data visualization libraries like Matplotlib, Plotly, and Altair.
3. Customizable UI Components: Streamlit provides various UI components that can be customized to fit the needs of the application. These components include sliders, dropdowns, buttons, and text inputs.
4. Real-time Updating: Streamlit automatically updates the web application in real-time as the user interacts with it. This makes it easy to create dynamic applications that respond to user input in real-time.
5. Integration with Machine Learning Libraries: Streamlit integrates seamlessly with popular machine learning libraries like TensorFlow, PyTorch, and Scikit-learn. This allows developers to build and deploy machine learning models with ease.
6. Sharing and Deployment: Streamlit makes it easy to share and deploy applications. Developers can share their applications with others by simply sharing a URL. Streamlit also provides tools for deploying applications to cloud services like Heroku and AWS.

ADVANTAGES OF STREAMLIT

Fast and Easy Development: Streamlit provides a simple and intuitive syntax that makes it easy to build interactive web applications for data science and machine learning projects.

With Streamlit, developers can build applications faster and with less code.

Real-Time Updates: Streamlit automatically updates the web application in real-time as the user interacts with it. This allows developers to create dynamic applications that respond to user input in real-time, without the need for manual updates.

Seamless Integration with Popular Libraries: Streamlit integrates seamlessly with popular data science and machine learning libraries like Matplotlib, Plotly, Pandas, TensorFlow, PyTorch, and Scikit-learn. This allows developers to build powerful and complex applications using their preferred libraries.

Customizable UI Components: Streamlit provides a range of UI components that can be customized to fit the needs of the application. These components include sliders, dropdowns, buttons, and text inputs, which can be easily customized with CSS.

Sharing and Deployment: Streamlit makes it easy to share and deploy applications. Developers can share their applications with others by simply sharing a URL. Streamlit also provides tools for deploying applications to cloud services like Heroku and AWS, making it easy to scale applications as needed.

Active Community Support: Streamlit has an active community of developers and users who contribute to the development of the library, provide support to other developers, and share their own projects and experiences with the library.

JUPYTER NOTEBOOK

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

A Jupyter Notebook document is a browser-based REPL containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media. Underneath the interface, a notebook is a JSON document, following a versioned schema, usually ending with the ".ipynb" extension.

Jupyter Notebook can connect to many kernels to allow programming in different languages. A Jupyter kernel is a program responsible for handling various types of requests (code execution, code completions, inspection), and providing a reply. Kernels talk to the other components of Jupyter using ZeroMQ, and thus can be on the same or remote machines. Unlike many other Notebook-like interfaces, in Jupyter, kernels are not aware that they are attached to a specific document, and can be connected to many clients at once. Usually kernels allow execution of only a single language, but there are a couple of exceptions.[citation needed] By default Jupyter Notebook ships with the IPython kernel. As of the 2.3 releas (October 2014), there are 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia and Haskell

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 libraror "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.

The Jupyter Notebook combines three components:

* The notebook web application: An interactive web application for writing and running code interactively and authoring notebook documents.
* Kernels: Separate processes started by the notebook web application that runs users’ code in a given language and returns output back to the notebook web application. The kernel also handles things like computations for interactive widgets, tab completion and introspection. • Notebook documents: Self-contained documents that contain a representation of all content visible in the note-book web application, including inputs and outputs of the computations, narrative text, equations, images, and rich media representations of objects. Each notebook document has its own kernel.

FEATURES OF JUPYTER NOTEBOOK

In-browser editing for code, with automatic syntax highlighting, indentation, and tab completion/introspection.

* The ability to execute code from the browser, with the results of computations attached to the code which generated them.
* Displaying the result of computation using rich media representations, such as HTML, LaTeX, PNG, SVG, etc.

For example, publication-quality figures rendered by the matplotlib library, can be included inline.

* In-browser editing for rich text using the Markdown markup language, which can provide commentary for the code, is not limited to plain text.

* The ability to easily include mathematical notation within markdown cells using LaTeX, and rendered natively by MathJax.

ADVANTAGES OF JUPYTER NOTEBOOK

There are the following advantages of Jupyter Notebook -

All in one place**:** As you know, Jupyter Notebook is an open-source web-based interactive environment that combines code, text, images, videos, mathematical equations, plots, maps, graphical user interface and widgets to a single document.

Easy to convert:Jupyter Notebook allows users to convert the notebooks into other formats such as HTML and PDF. It also uses online tools and nbviewer which allows you to render a publicly available notebook in the browser direct.

**4.3 ALGORITHMS**

Decision tree classifiers

Decision tree classifiers are used successfully in many diverse areas**.** Their most important feature is the capability of capturing descriptive decision making knowledge from the supplied data**.** Decision tree can be generated from training sets**.** The procedure for such generation based on the set of objects **(**S**)**, each belonging to one of the classes C1, C2, **…**, Ck is as follows**:**

Step 1.If all the objects in S belong to the same class, for example Ci, the decision tree for S consists of a leaf labeled with this class.

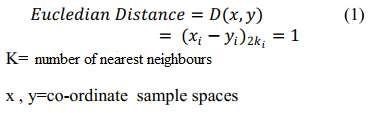
Step 2.Otherwise, let T be some test with possible outcomes O1, O2,**…**, On**.** Each object in S has one outcome for T so the test partitions S into subsets S1, S2,**…** Sn where each object in Si has outcome Oi for T**.** T becomes the root of the decision tree and for each outcome Oi we build a subsidiary decision tree by invoking the same procedure recursively on the set Si**.**

Gradient boosting

Gradient boostingis a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees.[1][2] When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest.A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function.

K-Nearest Neighbors (KNN)

KNN is slow supervised learning algorithm, it take more time to get trained classification like other algorithm is divided into two step training from data and testing it on new instance . The K Nearest Neighbour working principle is based on assignment of weight to the each data point which is called as neighbour. In K Nearest Neighbour distance is calculate for training dataset for each of the K Nearest data points now classification is done on basis of majority of votes there are three types of distances need to be measured in KNN Euclidian, Manhattan, Minkowski distance in which Euclidian will be consider most one the following formula is used to calculate their distance.



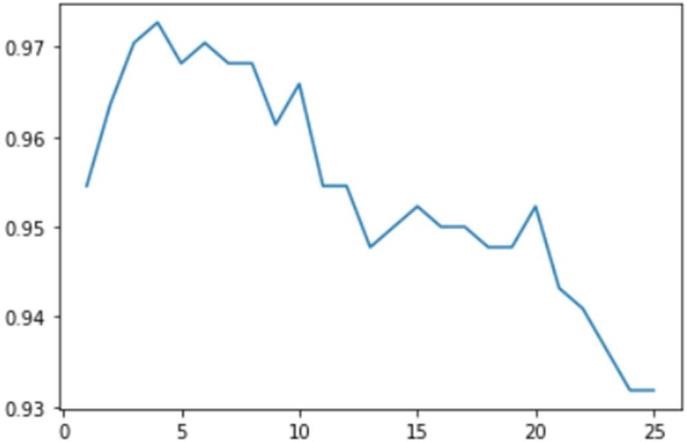
*In N dimensions, the Euclidean distance between two points p and q is √(∑i=1N (pi-qi)²) where pi (or qi) is the coordinate of p (or q) in dimension i*

algorithm for KNN is defined in the steps given below:

1. D represents the samples used in the training and k denotes the number of nearest neighbour.
2. Create super class for each sample class.
3. Compute Euclidian distance for every training sample 4. Based on majority of class in neighbour, classify the sample
4. Algorithm Implementation:

Step 1 − for implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data.

Step 2 − Next, we need to choose the value of K i.e. the nearest data points. K can be any odd integer.



Select the K value with the Highest accuracy estimated by the trail and Error method.

Step 3 − for each point in the test data do the following

3.1 − Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.

3.2 − Now, based on the distance value, sort them in ascending order.

3.3 − Next, it will choose the top K rows from the sorted array.

3.4 − Now, it will assign a class to the test point based on most frequent class of these rows.

Step 4 – End

Note : The selected K value should not be a even number because the even Number may cause Ambiguity.

Logistic regression analysis studies the association between a categorical

dependent variable and a set of independent (explanatory) variables. The name *logistic regression* is used when the dependent variable has only two values, such as 0 and 1 or Yes and No. The name *multinomial logistic regression* is usually reserved for the case when the dependent variable has three or more unique values, such as Married, Single, Divorced, or Widowed. Although the type of data used for the dependent variable is different from that of multiple regression, the practical use of the procedure is similar. Logistic regression competes with discriminant analysis as a method for analyzing categorical-response variables. Many statisticians feel that logistic regression is more versatile and better suited for modeling most situations than is discriminant analysis. This is because logistic regression does not assume that the independent variables are normally distributed, as discriminant analysis does.

This program computes binary logistic regression and multinomial logistic regression on both numeric and categorical independent variables. It reports on the regression equation as well as the goodness of fit, odds ratios, confidence limits, likelihood, and deviance. It performs a comprehensive residual analysis including diagnostic residual reports and plots. It can perform an independent variable subset selection search, looking for the best regression model with the fewest independent variables. It provides confidence intervals on predicted values and provides ROC curves to help determine the best cutoff point for classification. It allows you to validate your results by automatically classifying rows that are not used during the analysis.

Naïve Bayes

The naive bayes approach is a supervised learning method which is based on a simplistic hypothesis: it assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature .

Yet, despite this, it appears robust and efficient. Its performance is comparable to other supervised learning techniques. Various reasons have been advanced in the literature. In this tutorial, we highlight an explanation based on the representation bias. The naive bayes classifier is a linear classifier, as well as linear discriminant analysis, logistic regression or linear SVM (support vector machine). The difference lies on the method of estimating the parameters of the classifier (the learning bias).

While the Naive Bayes classifier is widely used in the research world, it is not widespread among practitioners which want to obtain usable results. On the one hand, the researchers found especially it is very easy to program and implement it, its parameters are easy to estimate, learning is very fast even on very large databases, its accuracy is reasonably good in comparison to the other approaches. On the other hand, the final users do not obtain a model easy to interpret and deploy, they does not understand the interest of such a technique.

Thus, we introduce in a new presentation of the results of the learning process. The classifier is easier to understand, and its deployment is also made easier. In the first part of this tutorial, we present some theoretical aspects of the naive bayes classifier. Then, we implement the approach on a dataset with Tanagra. We compare the obtained results (the parameters of the model) to those obtained with other linear approaches such as the logistic regression, the linear discriminant analysis and the linear SVM. We note that the results are highly consistent. This largely explains the good performance of the method in comparison to others. In the second part, we use various tools on the same dataset (Weka 3.6.0, R 2.9.2, Knime 2.1.1, Orange 2.0band RapidMiner 4.6.0). We try above all to understand the obtained results.

Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

The first algorithm for random decision forests was created in 1995 by Tin Kam Ho[1] using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006 (as of 2019, owned by Minitab, Inc.).The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho[1] and later independently by Amit and Geman[13] in order to construct a collection of decision trees with controlled variance.

Random forests are frequently used as "blackbox" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.

SVM

In classification tasks a discriminant machine learning technique aims at finding, based on an *independent and identically distributed* **(***iid***)** training dataset, a discriminant function that can correctly predict labels for newly acquired instances**.** Unlike generative machine learning approaches, which require computations of conditional probability distributions, a discriminant classification function takes a data point *x* and assigns it to one of the different classes that are a part of the classification task**.** Less powerful than generative approaches, which are mostly used when prediction involves outlier detection, discriminant approaches require fewer computational resources and less training data, especially for a multidimensional feature space and when only posterior probabilities are needed**.** From a geometric perspective, learning a classifier is equivalent to finding the equation for a multidimensional surface that best separates the different classes in the feature space**.**

SVM is a discriminant technique, and, because it solves the convex optimization problem analytically, it always returns the same optimal hyperplane parameter**—**in contrast to *genetic algorithms* **(***GAs***)** or *perceptrons*, both of which are widely used for classification in machine learning**.** For perceptrons, solutions are highly dependent on the initialization and termination criteria**.** For a specific kernel that transforms the data from the input space to the feature space, training returns uniquely defined SVM model parameters for a given training set, whereas the perceptron and GA classifier models are different each time training is initialized**.** The aim of GAs and perceptrons is only to minimize error during training, which will translate into several hyperplanes**’** meeting this requirement**.**

ADA BOOST

AdaBoost, also called Adaptive Boosting, is a technique in Machine Learning used as an Ensemble Method. The most common estimator used with AdaBoost is decision trees with one level which means Decision trees with only 1 split. These trees are also called **Decision Stumps.** Decision stumps are the simplest model we could construct in terms of complexity. The algo would just guess the same label for every new example, no matter what it looked like. The accuracy of such a model would be best if we guess whichever answer, 1 or 0, is most common in the data. If, say, 60 percent of the examples are 1s, then we’ll get 60 percent accuracy just by guessing 1 every time. Decision stumps improve upon this by splitting the examples into two subsets based on the value of one feature. Each stump chooses a feature, say X2, and a threshold, T, and then splits the examples into the two groups on either side of the threshold. To find the decision stump that best fits the examples, we can try every feature of the input along with every possible threshold and see which one gives the best accuracy.

**CHAPTER 5**

**TESTING AND RESULT**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**5.1 TYPES OF TESTS**

UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

|  |  |
| --- | --- |
| Valid Input | : identified classes of valid input must be accepted. |
| Invalid Input | : identified classes of invalid input must be rejected. |
| Functions | : identified functions must be exercised. |
| Output | : identified classes of application outputs must be exercised. |

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

SYSTEM TESTING

System testing ensures that the entire integrated software system meets

requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner

workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**5.2 Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail. Test objectives

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

Features to be tested

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications,

e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results:All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results:All the test cases mentioned above passed successfully. No defects encountered.

TESTING METHODOLOGIES

The following are the Testing Methodologies: o Unit Testing. o Integration Testing.

* User Acceptance Testing. o Output Testing.
* Validation Testing.

Unit Testing

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module’s control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

**5.3 Integration Testing**

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

1)Top Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

2. Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

* The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
* A driver (i.e.) the control program for testing is written to coordinate test case input and output.
* The cluster is tested.
* Drivers are removed and clusters are combined moving upward in the program structure The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

OTHER TESTING METHODOLOGIES

**5.4 User Acceptance Testing**

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

Validation Checking

Validation checks are performed on the following fields.

Text Field

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

Numeric Field:

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Using Live Test Data:

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

Using Artificial Test Data:

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package “Virtual Private Network” has satisfied all the requirements specified as per software requirement specification and was accepted.

USER TRAINING

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

MAINTAINENCE

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user’s requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future.

The coding and designing is simple and easy to understand which will make maintenance easier.

TESTING STRATEGY

A strategy for system testing integrates system test cases and design techniques into a well-planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation. A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

**5.5 SYSTEM TESTING**

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system.

**5.6 Manual Testing**

Test Case for Heart Disease Prediction

|  |  |
| --- | --- |
| Test Case | Heart Disease Prediction |
| Test Description | The user enters the symptoms.User answers the sub questions |
| Requirement Verified | Yes |
| Test Environment | System should connect to network and server should be always on and tested from dataset |
| Test setup/precondition | Receive should be in connection with accept state |
| Actions | The user will checks the symptoms and attributes and submit the button for diagnosis. |
| Expected Result | The User having that Particular disease or not. |
| Pass /Fail | Pass |
| Note | Successfully Executed |

Describes that The above table shows the test case for Values Entered. In this the main importance is given to check the entered values are Compared with Dataset values. If the values are in Matching then the test is passed.

Test Case for Diabetes Disease Prediction

|  |  |
| --- | --- |
| Test Case | Diabetes Disease Prediction |
| Test Description | The user enters the symptoms.User answers the sub questions |
| Requirement Verified | Yes |
| Test Environment | System should connect to network and server should be always on and tested from dataset |
| Test setup/precondition | Receive should be in connection with accept state |
| Actions | The user will checks the symptoms and attributes and submit the button for diagnosis. |
| Expected Result | The User having that Particular disease or not. |
| Pass /Fail | Pass |
| Note | Successfully Executed |

Describes that The above table shows the test case for Values Entered. In this the main importance is given to check the entered values are Compared with Dataset values. If the values are in Matching then the test is passed.

Test Case for Parkinsons Disease Prediction

|  |  |
| --- | --- |
| Test Case | Parkinsons Disease Prediction |
| Test Description | The user enters the symptoms.User answers the sub questions |
| Requirement Verified | Yes |
| Test Environment | System should connect to network and server should be always on and tested from dataset |
| Test setup/precondition | Receive should be in connection with accept state |
| Actions | The user will checks the symptoms and attributes and submit the button for diagnosis. |
| Expected Result | The User having that Particular disease or not. |
| Pass /Fail | Pass |
| Note | Successfully Executed |

Describes that The above table shows the test case for Values Entered. In this the main importance is given to check the entered values are Compared with Dataset values. If the values are in Matching then the test is passed.

**CHAPTER 6**

**CONCLUSION AND FUTURE SCOPE**

**6.1 Conclusion**

Multiple disease prediction using machine learning is a promising approach to healthcare that has the potential to revolutionize the way we diagnose and treat diseases.

By using machine learning algorithms to analyze large amounts of patient data, we can identify patterns and correlations that may not be immediately apparent to human clinicians.

This approach has the potential to enable earlier diagnosis, better treatment, and improved patient outcomes.

While there are challenges and limitations to the use of machine learning in healthcare, such as the risk of bias and the need for diverse and representative data, ongoing research and development in this field is helping to address these challenges and unlock the full potential of multiple disease prediction using machine learning.

As technology continues to evolve and more data becomes available, it is likely that machine learning algorithms will become increasingly sophisticated and accurate, leading to even better patient outcomes and more personalized medicine.

Multiple disease prediction using machine learning has the potential to transform healthcare, and it is an exciting area of research that holds great promise for the future.

**6.2 Future Scope**

Incorporating more data sources: Currently, multiple disease prediction systems typically rely on electronic health records and medical imaging data. In the future, other data sources such as wearable devices, social media, and environmental data could be integrated into these systems to provide a more comprehensive picture of a patient's health. a

Addressing data bias: As with all machine learning algorithms, bias in the training data can lead to inaccurate predictions and perpetuate health disparities. Future work should focus on developing methods to address and mitigate data bias, such as using more diverse and representative datasets, and incorporating fairness and equity considerations into the algorithm development process.

Advancing personalized medicine: Multiple disease prediction using machine learning has the potential to enable more personalized and precise medicine, by predicting an individual's risk of developing specific diseases based on their unique medical history and other factors. Future work should focus on developing personalized treatment plans based on these predictions, including targeted prevention strategies and personalized treatment options.

**REFERENCES**

1. D. Lahiri, P. C. P. Kandimalla, and A. Jeysekar, “Hybrid multi purpose voice assistant,” in 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC). IEEE, 2023, pp. 816– 822.
2. I. Shazhaev, D. Mikhaylov, A. Shafeeg, A. Tularov, and I. Shazhaev, “Personal voice assistant: from inception to everyday application,” Indonesian Journal of Data and Science, vol. 4, no. 2, pp. 64–70, 2023.
3. A. Valera Roman,´ D. Pato Mart´ınez, A. Lozano Murciego, D. M. Jimenez´-Bravo, and J. F. de Paz, “Voice assistant application for avoiding sedentarism in elderly people based on iot technologies,” Electronics, vol. 10, no. 8, p. 980, 2021.
4. P.-S. Chiu, J.-W. Chang, M.-C. Lee, C.-H. Chen, and D.-S. Lee, “Enabling intelligent environment by the design of emotionally aware virtual assistant: A case of smart campus,” IEEE Access, vol. 8, pp. 62 032–62 041, 2020.
5. P. Cheng and U. Roedig, “Personal voice assistant security and privacy—a survey,” Proceedings of the IEEE, vol. 110, no. 4, pp. 476–507, 2022.
6. F. Elghaish, J. K. Chauhan, S. Matarneh, F. P. Rahimian, and M. R. Hosseini, “Artificial intelligence-based voice assistant for bim data management,” Automation in Construction, vol. 140, p. 104320, 2022.
7. A. Poushneh, “Humanizing voice assistant: The impact of voice assistant personality on consumers’ attitudes and behaviors,” Journal of Retailing and Consumer Services, vol. 58, p. 102283, 2021.
8. M. Malik, M. K. Malik, K. Mehmood, and I. Makhdoom, “Automatic speech recognition: a survey,” Multimedia Tools and Applications, vol. 80, pp. 9411–9457, 2021.
9. S. Raju, V. Jagtap, P. Kulkarni, M. Ravikanth, and M. Rafeeq, “Speech recognition to build context: A survey,” in 2020 international conference on computer science, engineering and applications (ICCSEA). IEEE, 2020, pp. 1–7.
10. A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, “Speech recognition using deep neural networks: A systematic review,” IEEE access, vol. 7, pp. 19 143–19 165, 2019.
11. A. Singh, V. Kadyan, M. Kumar, and N. Bassan, “Asroil: a comprehensive survey for automatic speech recognition of indian languages,” Artificial Intelligence Review, vol. 53, pp. 3673–3704, 2020.
12. M. Alam, M. D. Samad, L. Vidyaratne, A. Glandon, and K. M. Iftekharuddin, “Survey on deep neural networks in speech and vision systems,” Neurocomputing, vol. 417, pp. 302–321, 2020.
13. D. Al-Fraihat, Y. Sharrab, F. Alzyoud, A. Qahmash, M. Tarawneh, and A. Maaita, “Speech recognition utilizing deep learning: A systematic review of the latest developments,” HUMAN-CENTRIC COMPUTING AND INFORMATION SCIENCES, vol. 14, 2024.
14. D. O’Shaughnessy, “Trends and developments in automatic speech recognition research,” Comput. Speech Lang., vol. 83, no. C, jan 2024. [Online]. Available: https://doi.org/10.1016/j.csl.2023.101538
15. O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, “Convolutional neural networks for speech recognition,” IEEE/ACM Transactions on audio, speech, and language processing, vol. 22, no. 10, pp. 1533–1545, 2014.
16. Z. Song, “English speech recognition based on deep learning with multiple features,” Computing, vol. 102, no. 3, pp. 663–682, 2020.
17. P. Bell, J. Fainberg, O. Klejch, J. Li, S. Renals, and P. Swietojanski, “Adaptation algorithms for neural network-based speech recognition: An overview,” IEEE Open Journal of Signal Processing, vol. 2, pp. 33–66, 2020.