A

Industry Oriented Mini Project Report

On

**LIVER DISEASE PREDICTION USING MACHINE LEARNING**

(Submitted in partial fulfilment of the requirements for the award of Degree)

Bachelor of Technology

in

**COMPUTER SCIENCE & ENGINEERING (DATA SCIENCE)**

By

**SHAMAKURA SHRUTHIKA REDDY (217R1A67J7)**

**PEDDIREDDY RISHIKA REDDY (217R1A67H9)**

**KANIKIREDDY GNANADEEP (217R1A67F4)**

Under the guidance of

**A.VEERENDER**

Asst Professor

A logo with a flower

Description automatically generated

**Department of Computer Science & Engineering (Data Science)**

**CMR TECHNICAL CAMPUS**

**UGC AUTONOMOUS**

**Accredited by NBA & NAAC with 'A' Grade**

**Approved by AICTE, New Delhi and JNTU, Hyderabad.**

2024-2025

**Department of Computer Science & Engineering (Data Science)**

A logo with a flower

Description automatically generated

**CERTIFICATE**

This is to certify that the project entitled **“Liver Disease Prediction Using Machine Learning”** being submitted by **SHAMAKURA SHRUTHIKA REDDY (217R1A67J7), PEDDIREDDY RISHIKA REDDY (217R1A67H9), KANIKIREDDY GNANADEEP (207R1A67F4)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering (Data Science) to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2024-25.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

**A.Veerender Dr. K. Murali**

Asst Professor Associate Professor & HoD

**Internal Guide**

**External Examiner**

**Submitted for viva-voce Examination held on:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ACKNOWLEDGMENT**

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **“A.Veerender”**, Designation for “his” exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **Mr. Sanjib Kumar Nayak, Mrs. V. Prema Tulasi, Mr. B. Ramesh, Mr. A. Veerender, Mrs. V. Sandya, Ms. Sandhyarani** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Murali**, Head, Department of Computer Science and Engineering (Data Science) for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy**, Director for being cooperative throughout the course of this project. We also express our sincere gratitude to **Sri. Ch. Gopal Reddy**, Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

**S. SHRUTHIKA REDDY (217R1A67J7)**

**P. RISHIKA REDDY (217R1A67H9)**

**K. GNANADEEP (217R1A67F4)**

**ABSTRACT**

Liver is an important organ that performs hundreds of tasks related to metabolism, energy storage, and detoxification of waste. It helps you to digest the food, and convert it to energy, and store the energy until you need it. It also helps filter toxic substances out of your bloodstream. Liver disease is a general term that refers to any condition affecting your liver. These conditions may develop for different reasons, but they can all damage your liver and impact its function. Machine Learning is a process which is used to discover patterns in huge data/ large data set to enable decision, thereby allowing machines to go through a learning process (i.e. supervised, unsupervised and semi-supervised or reinforced). The algorithms used here for predicting liver patients are Logistic regression, Decision Tree, Random Forest, K-Nearest Neighbour, Gradient Boosting, Extreme Gradient Boosting. Based on the analysis and result calculations, it was found that these algorithms has obtained good accuracy after feature selection.

**TABLE OF CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Certificates | | |  | ii |
| Acknowledgement | | |  | iv |
| Abstract | | |  | vi |
| Table of Contents | | |  | x |
| List of Tables | | |  | xii |
| List of Figures | | |  | xiii |
| List of Screenshots | | |  |  |
| List of Abbreviations | | | | xiv |
| **CHAPTER – 1** [**introduction**](file:///C:\Users\ADMIN\Downloads\1.docx#_Toc465724132) | | | | **1** |
| 1.1 | | PROJECT SCOPE | | 1 |
| 1.2 | | PROJECT PURPOSE | | 3 |
| 1.3 | | PROJECT FEATURES | | 12 |
| 1.4 | |  | | 13 |
| 1.5 | |  | | 14 |
| 1.6 | |  | | 14 |
| **CHAPTER – 2 SYSTEM ANALYSIS** | | | | **16** |
| 2.1 | PROBLEM DEFINITION | | | 16 |
| 2.2 | EXISTING SYSTEM | | | 19 |
|  | 2.2.1 LIMITATIONS OF THE EXISTING SYSTEM | | |  |
| 2.3 | PROPOSED SYSTEM | | | 22 |
|  | 2.3.1 ADVANTAGES OF PROPOSED SYSTEM | | |  |
| 2.4 | FEASIBILITY STUDY | | | 28 |
|  | 2.4.1 ECONOMICAL FEASIBILITY | | |  |
|  | 2.4.2 TECHNICAL FEASIBILITY | | |  |
|  | 2.4.3 SOCIAL FEASIBILITY | | |  |
| 2.5 | HARDWARE & SOFTWARE REQUIREMENTS | | | 33 |
|  | 2.5.1 HARDWARE REQUIREMENTS | | |  |
|  | 2.5.2 SOFTWARE REQUIREMENTS | | |  |
| 2.6 | PYTHON OR JAVA | | | 36 |
| **CHAPTER – 3 HARDWARE & SOFTWARE REQUIREMENTS** | | | | **38** |
| 3.1 | PROJECT ARCHITECTURE | | | 38 |
| 3.2 | USE CASE DIAGRAM | | | 40 |
| 3.3 | CLASS DIAGRAM | | | 43 |
| 3.4 | SEQUENCE DIAGRAM | | | 43 |
| 3.5 | ACTIVITY DIAGRAM | | |  |
|  | 3.5.1 RMOTE USER | | |  |
|  | 3.5.2 SERVICE PROVIDER | | |  |
| **CHAPTER – 4 IMPLEMENTATION** | | | | **46** |
| 4.1 | MACHINE LEARNING ALGORITHMS | | | 46 |
| 4.2 | DATA SET | | | 48 |
| 4.3 | SAMPLE CODE | | | 56 |
| 4.4 |  | | | 56 |
| 4.5 |  | | | 57 |
| **CHAPTER – 5 SCREENSHOTS** | | | | **59** |
| 5.1 |  | | | 59 |
| [**CHAPTER – 6 TESTING**](file:///C:\Users\ADMIN\Downloads\1.docx#_Toc465724211) | | | | **113** |
| 6.1 | INTRODUCTION TO TESTING | | | 113 |
| 6.2 | TYPES OF TESTING | | | 116 |
|  | 6.2.1 UNIT TESTING | | |  |
|  | 6.2.2 INTEGRATION TESTING | | |  |
|  | 6.2.3 FUNCTIONAL TESTING | | |  |
| 6.3 | TEST CASES | | |  |
| [**CHAPTER – 7 CONCLUSION & FUTURE SCOPE**](file:///C:\Users\ADMIN\Downloads\1.docx#_Toc465724211) | | | | **113** |
| 7.1 | PROJECT CONCLUSION | | | 113 |
| 7.2 | FUTURE SCOPE | | | 116 |
| [**CHAPTER – 8 CONCLUSION & FUTURE SCOPE**](file:///C:\Users\ADMIN\Downloads\1.docx#_Toc465724211) | | | | **113** |
|  | REFERENCES | | | 113 |
|  | GITHUB LINK | | | 116 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No:** | **Name of the table** | **Page No** |
| Table 6.1 |  | 73 |
| Table 6.2 |  | 75 |
| Table 6.3 |  | 77 |
| Table 6.4 |  | 93 |
| Table 6.5 |  | 95 |
| Table 6.6 |  | 96 |
| Table 6.7 |  | 98 |
| Table 6.8. |  | 100 |
| Table 6.9 |  | 102 |
| Table 6.10 |  | 104 |
| Table 6.11 |  | 105 |
| Table 6.12 |  | 106 |
| Table 6.13 |  | 107 |
| Table 6.14 |  | 108 |
| Table 7.1 |  | 110 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No** | **Name of the Figure** | **Page no** |
| Fig: 1.1 |  | 12 |
| Fig: 1.2 |  | 13 |
| Fig: 1.3 |  | 13 |
| Fig: 1.4 |  | 14 |
| Fig: 6.1 |  | 74 |
| Fig: 6.2 |  | 77 |
| Fig: 6.3 |  | 78 |
| Fig: 6.4 |  | 94 |
| Fig: 6.5 |  | 97 |
| Fig: 6.6 |  | 99 |
| Fig: 6.7 |  | 101 |
| Fig: 6.8 |  | 103 |
| Fig: 6.9 |  | 104 |
| Fig 6.10 |  | 107 |
| Fig 7.1 |  | 110 |
| Fig 7.2 |  | 111 |

**LIST OF SCREENSHOTS**

|  |  |  |
| --- | --- | --- |
| **Screen Shot** | Name of the Screen Shot | **Page no** |
| Screen Shot 3.1 |  | 12 |
| Screen Shot 3.2 |  | 13 |
|  |  | 13 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| 1. CCAF |  |
| 1. CCSs |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Chapter 1**

**INTRODUCTION**

* 1. **Project Scope**

This project is to predict the liver diseases for a patient with the maximum amount of accuracy in our prediction. This dataset was collected from Indian patient liver disease dataset from Kaggle database of Indian Liver patient records and used that dataset in our three modules to predict the liver disease using various machine learning techniques

* 1. **Project Purpose**

Machine learning has become one of the most evolving technologies in the current period. Machine learning can be simply explained as scientific study of algorithms and models in statistics where machines can easily understand to perform and solve specific tasks. This technique has become agile and it has been a requirement in most of the fields.

* 1. **Project Features**

The features of a project are the distinctive characteristics and components that define its scope, objectives, and implementation. For the project titled "Liver Disease Prediction Using Machine Learning" the key features are Data Collection and Integration, Data Preprocessing, Algorithm Selection and Implementation, Model Training and Testing, Comparative Analysis, Visualization and Reporting, Project Timeline and Milestones.

**Chapter 2**

**SYSTEM ANALYSIS**

1. **Problem Definition**

Liver is an important organ that performs hundreds of tasks related to metabolism, energy storage, and detoxification of waste. It helps you digest food, convert it to energy, and store the energy until you need it. It also helps filter toxic substances out of your bloodstream. Liver disease is a general term that refers to any condition affecting your liver. These conditions may develop for different reasons, but they can all damage your liver and impact its function.

**2.2 Existing System**

The traditional liver disease diagnosis system involves doctors or medical professionals using various medical tests, such as blood tests, biopsy, and imaging techniques like ultrasound, MRI, or CT scans to identify liver disease in patients. The interpretation of the test results and diagnosis is done by medical professionals based on their experience and knowledge. This approach can be time-consuming and costly, and the accuracy of diagnosis may depend on the skills and experience of the medical professionals.

**2.2.1 Limitations of the Existing System**

* **Invasive Procedures**: Liver biopsy remains the most definitive method but is invasive and associated with patient discomfort and risk.
* **Variable Accuracy**: Non-invasive methods like elastography can have variable accuracy based on patient-specific factors and operator skill.
* **Limited Biomarker Sensitivity**: Serum markers may lack sensitivity and specificity, potentially leading to misclassification of fibrosis stages.
* **Complex Integration**: Combining multiple non-invasive methods requires complex data integration and may not always yield accurate results.

**2.3 Proposed System**

In the proposed system, the system concentrates on comparing prediction performance of machine learning models (Decision Tree, XG Boost, SVC, Naïve Bayes, KNN, Logistic Regression and ANN) and two deep learning methods (RNN and LSTM) to predict liver disease. A multilayer perceptron (MLP)is a neural network model which can map liver datasets of input file onto a collection of applicable outputs. So, in the proposed work the liver disease prediction model was build. There are many factors which causes the liver disease. Some of them which influence to detect the liver disease are Total Bilirubin (Total amount of bilirubin when old red blood cell breaks down inside the human body), Direct Bilirubin (It is a substance is made when the body breakdowns the old red blood cells.)

**2.3.1 Advantages of the Proposed System**

* The proposed system implements a graph-based theory and using the machine learning and deep learning techniques.
* In the proposed system, each of the algorithms can effectively solve stock prediction problems.
* The system is more effective due to presence of Extreme Gradient Boosting (XG Boost), Support Vector Classier (SVC) techniques.

#### **2.4 Feasibility Study**

The feasibility of the project is analyzed in this phase and a 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. This is to ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis:

* Economic Feasibility
* Technical Feasibility
* Social Feasibility

**2.4.1 Economic Feasibility**

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on a project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require. The following are some of the important financial questions asked during preliminary investigation:

* The costs conduct a full system investigation.
* The cost of the hardware and software.
* The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication that the system is economically possible for development.

**2.4.2 Technical Feasibility**

This study is carried to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**2.4.3 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.

**2.5 Hardware and Software Requirements**

**2.5.1 Hardware Requirements**

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

* PROCESSOR : i3 or above
* RAM : 4GB (min)
* HARD DISK : 20 GB
* INPUT DEVICES : Keyboard, Mouse

**2.5.2 Software Requirements**

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements.

* OPERATING SYSTEM : Windows 10
* CODE LANGUAGE : Python
* LIBRARIES : Numpy, Pandas, Matplotlib
* FRONT-END : Python
* BACK-END : Django-ORM
* DESIGNING : HTML, CSS, JavaScript
* DATABASE : MySQL (WAMP Server)

**2.6 Python**

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

* 1. [Machine Learning](https://www.geeksforgeeks.org/machine-learning/)
  2. GUI Applications (like Kivy, Tkinter, PyQt etc.. )
  3. Web frameworks like Django (used by YouTube, Instagram, Dropbox)
  4. Image processing (like Opencv, Pillow)
  5. Web scraping (like Scrapy, BeautifulSoup, Selenium)
  6. Test frameworks
  7. Multimedia

### Advantages of Python :-

### 1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

#### 2. Extensible

As we have seen earlier, Python can be**extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

#### 3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities**to our code in the other language.

#### 4. Improved Productivity

The language’s simplicity and extensive libraries render programmers**more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

#### 5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

#### 6. Simple and Easy

When working with Java, you may have to create a class to print **‘Hello World’**. But in Python, just a print statement will do. It is also quite **easy to learn, understand,** and**code.** This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

#### 7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory.** This further aids the readability of the code.

#### 8. Object-Oriented

This language supports both the **procedural and object-oriented**programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

#### 9. Free and Open-Source

Like we said earlier, Python is **freely available.** But not only can you[**download Python**](https://data-flair.training/blogs/install-python-windows/) for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

#### 10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to**code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA).** However, you need to be careful enough not to include any system-dependent features.

### Disadvantages of Python:

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

#### 1. Speed Limitations

We have seen that Python code is executed line by line. But since [Python](https://www.python.org/) is interpreted, it often results in **slow execution**. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

#### 2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle.**

The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

#### 3. Design Restrictions

As you know, Python is **dynamically-typed**. This means that you don’t need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can**raise run-time errors**.

#### 4. Underdeveloped Database Access Layers

Compared to more widely used technologies like **JDBC (Java DataBase Connectivity)**and**ODBC (Open DataBase Connectivity),** Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

#### 5. Simple

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

**History of Python : -**

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

**Chapter 3**

**HARDWARE & SOFTWARE REQUIREMENTS**

1. **Project Architecture**

This project architecture shows the procedure followed for classification, starting from input to final prediction.

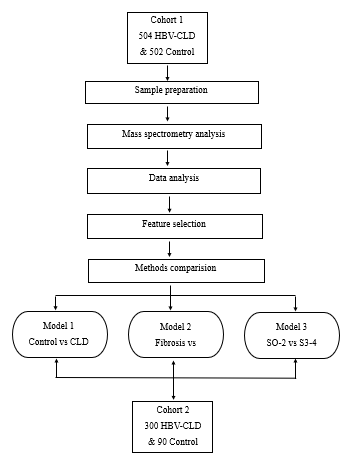


Figure 3.1: Project Architecture of Liver Disease Prediction Using Machine learning

1. **Use Case Diagram**

In the use case diagram, we have basically one actor who is the user in the trained model.

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

****Figure 3.2: Use Case Diagram of Liver Disease Prediction Using Machine learning

1. **Class Diagram**

Class diagram is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations and the relationships among objects

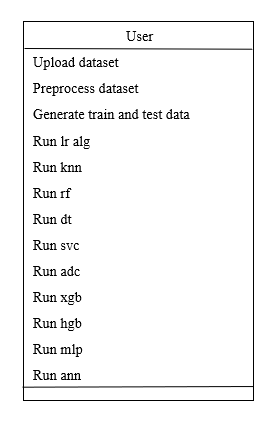
****

Figure 3.3: Class Diagram of Liver Disease Prediction Using Machine learning

1. **Sequence Diagram**

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development.



Figure 3.4: Sequence Diagram of Liver Disease Prediction Using Machine learning

1. **Activity Diagram**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. They can also include elements showing the flow of data between activities through one or more datastores.

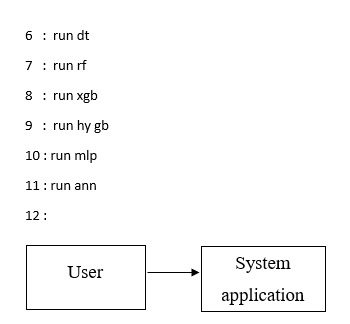


Figure 3.5: Activity Diagram of Liver Disease Prediction Using Machine learning

**Chapter 4**

**IMPLEMENTATION**

* 1. **Machine Learning Algorithms**

Machine learning is often categorized as a subfield of artificial intelligence**.** The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data. Fundamentally, machine learning involves building mathematical models to help understand data. Understanding the problems in machine learning it is essential to use these algorithms effectively, and so we will start with some broad categorizations of the algorithms.

* + 1. **Logistic Algorithm**

Logistic regression is a machine learning algorithm used for binary classification tasks, where the goal is to predict one of two possible outcomes based on input features. It works by fitting a linear equation to the input data, then applying the sigmoid (logistic) function to transform the linear output into a probability between 0 and 1. The sigmoid function ensures that the predictions are interpreted as probabilities, making it useful for classifying data into two categories. The model predicts the class by applying a threshold, typically 0.5, to decide whether the output corresponds to class 0 or class 1. Despite its name, logistic regression is primarily used for classification rather than regression. It is widely used in applications such as spam detection, medical diagnosis, and fraud detection.

* + 1. **K Nearest Neighbor Algorithm**

The K-Nearest Neighbors (KNN) algorithm is a simple, non-parametric method used in machine learning for both classification and regression tasks. It works by finding the **k** closest data points, or neighbors, to a given input based on a distance metric like Euclidean distance. For classification, KNN assigns the most common class among the **k** nearest neighbors as the predicted class, while for regression, it averages the values of the **k** neighbors to make a prediction. Since KNN doesn't build a model during training (making it a "lazy learner"), it stores the entire dataset and makes predictions based on the neighbors at the time of the query. The choice of **k** is important, as smaller values may overfit the data, while larger values could overlook local patterns.

* + 1. **Random Forest Algorithm**

The Random Forest algorithm is an ensemble learning technique used for both classification and regression tasks in machine learning. It works by creating a "forest" of multiple decision trees during training, where each tree is trained on a random subset of the data and a random subset of the features. The predictions of all the trees are combined to make the final output—by majority vote for classification or averaging for regression. This randomness helps reduce overfitting and improves the model's accuracy and robustness. Random Forest is popular due to its ability to handle large datasets, manage missing data, and reduce variance while maintaining high predictive power.

* + 1. **Decision Tree Algorithm**

The Decision Tree algorithm is a popular supervised learning technique used for both classification and regression tasks in machine learning. It works by splitting the data into subsets based on feature values, forming a tree-like structure where each internal node represents a decision based on a feature, and each leaf node represents a predicted outcome. The tree is built by recursively selecting the feature that best separates the data, often using metrics like Gini impurity or information gain for classification, and mean squared error for regression. Decision Trees are easy to interpret and can handle both categorical and numerical data, but they are prone to overfitting, especially with complex trees, which can be mitigated by techniques like pruning.

* + 1. **Support Vector Classifier Algorithm**

The Support Vector Classifier (SVC) is a powerful supervised learning algorithm used for classification tasks in machine learning. It works by finding a hyperplane that best separates the data points of different classes in a high-dimensional space. SVC aims to maximize the margin between the closest points (called support vectors) from each class and the separating hyperplane. This maximized margin ensures better generalization to unseen data. SVC can handle both linear and non-linear classification problems, and for non-linear cases, it uses kernel functions (like polynomial or radial basis function kernels) to project data into higher dimensions where a linear separation is possible. It is effective for high-dimensional datasets but can be computationally expensive for large datasets.

* + 1. **Analog & Digital Classifier Algorithm**

In machine learning, classifiers are generally designed to work with digital data, but the distinction between **analog** and **digital classifiers** is more conceptual. An **analog classifier** would theoretically process continuous, non-discrete input data (like voltages or signals), but most modern machine learning classifiers operate in the **digital domain**, where data is represented as discrete values. Digital classifiers, such as decision trees, support vector machines, or neural networks, rely on processing structured, numerical, or categorical inputs to classify data into different categories. While analog signal processing techniques might classify signals in real-time in areas like communication systems, machine learning classifiers typically work with pre-processed, digitized data to predict or categorize outcomes in various fields like image recognition, text analysis, and speech processing. The term "analog classifier" is not commonly used in the context of machine learning algorithms, as most classification tasks involve digital data.

* + 1. **eXtreme Gradient Boosting Algorithm**

eXtreme Gradient Boosting (XGBoost) is a highly efficient and scalable machine learning algorithm based on the gradient boosting framework. It is used for both classification and regression tasks and works by sequentially building decision trees, where each new tree corrects the errors of the previous ones. XGBoost differs from traditional gradient boosting by implementing several optimizations, such as regularization to prevent overfitting, parallelized tree construction for faster training, and handling missing values effectively. It also uses a more advanced loss function to improve predictive performance. Due to its speed, flexibility, and ability to handle large datasets, XGBoost has become a popular choice for competitive machine learning tasks and is widely used in applications like ranking, recommendation systems, and time series forecasting.

* + 1. **Artificial Neural network Algorithm**

An Artificial Neural Network (ANN) is a machine learning algorithm inspired by the structure and function of the human brain. It is composed of layers of interconnected nodes (neurons), where each neuron processes input data and passes it to the next layer. ANNs typically consist of an input layer, one or more hidden layers, and an output layer. Each connection between neurons has an associated weight, and neurons apply activation functions (like ReLU or sigmoid) to decide whether to pass a signal forward. The network learns by adjusting these weights through a process called backpropagation, which minimizes the error between the predicted output and the actual target. ANNs are widely used for complex tasks like image recognition, natural language processing, and speech recognition due to their ability to model intricate, non-linear relationships in data.

* 1. **Dataset**
  2. **Sample Code**

from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

from imutils import paths

from tkinter.filedialog import askopenfilename

from tkinter import scrolledtext

import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder, MinMaxScaler

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

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

from sklearn.svm import SVC

from sklearn.preprocessing import MinMaxScaler

from xgboost import XGBClassifier

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

#get\_ipython().run\_line\_magic('matplotlib', 'inline')

import scikitplot as skplt

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn import metrics

from matplotlib import pyplot as plt

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import AdaBoostClassifier

from sklearn.svm import LinearSVC

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import make\_pipeline

from sklearn.ensemble import StackingClassifier

from sklearn.neural\_network import MLPClassifier

from sklearn.experimental import enable\_hist\_gradient\_boosting # noqa

from sklearn.ensemble import HistGradientBoostingClassifier

import keras

from keras.models import Sequential

from keras.layers import Dense

from keras.models import load\_model

import os

from os import path

main = tkinter.Tk()

main.title("Severity of Liver Fibrosis for Chronic HBV based on Physical Layer with Serum Markers")

main.geometry("1300x1200")

global filename

global raw\_data

global X, y, X\_train, X\_test, y\_train, y\_test

global ltsm\_acc, ann\_acc, mlp\_acc,cnn\_acc

global MODEL\_PATH

def upload():

global filename

text.delete('1.0', END)

filename = askopenfilename(initialdir="dataset")

pathlabel.config(text=filename)

text.insert(END, "Dataset loaded\n\n")

def get\_redundant\_pairs(df):

'''Get diagonal and lower triangular pairs of correlation matrix'''

pairs\_to\_drop = set()

cols = df.columns

for i in range(0, df.shape[1]):

for j in range(0, i+1):

pairs\_to\_drop.add((cols[i], cols[j]))

return pairs\_to\_drop

def get\_top\_abs\_correlations(df, n=5):

au\_corr = df.corr().abs().unstack()

labels\_to\_drop = get\_redundant\_pairs(df)

au\_corr = au\_corr.drop(labels=labels\_to\_drop).sort\_values(ascending=False)

return au\_corr[0:n]

def preprocess():

global filename

global raw\_data

global X,y

text.delete('1.0',END)

text.insert(END,"Importing dataset\n")

raw\_data = pd.read\_excel(filename)

text.insert(END,"Data column information: "+str(raw\_data.columns)+"\n\n")

text.insert(END,"data shape"+str(raw\_data.shape)+"\n\n")

raw\_data = raw\_data.drop(['Physical Activity','PVD', 'Source of Care','Family HyperTension','Family Hepatitis','Chronic Fatigue','PVD','Region'],axis=1)

raw\_data.head()

raw\_data['Gender'] = raw\_data['Gender'].map({'F': 0, 'M': 1})

text.insert(END,"Top Absolute Correlations")

text.insert(END,"Top Correlation values: "+str(get\_top\_abs\_correlations(raw\_data, 10))+"\n\n")

raw\_data.isnull().sum()

raw\_data = raw\_data.drop(['Weight','Obesity', 'Waist','Bad Cholesterol'],axis=1)

raw\_data.dtypes

print("Top Absolute Correlations")

print(get\_top\_abs\_correlations(raw\_data, 10))

raw\_data.isnull().sum()

cols\_mode = ['Hepatitis', 'Diabetes', 'HyperTension', 'Education', 'Unmarried','PoorVision','Income']

for column in cols\_mode:

raw\_data[column].fillna(raw\_data[column].mode()[0], inplace=True)

cols\_mode = ['Height', 'Body Mass Index', 'Maximum Blood Pressure', 'Minimum Blood Pressure', 'Good Cholesterol','Total Cholesterol','Income']

for column in cols\_mode:

raw\_data[column].fillna(raw\_data[column].mean(), inplace=True)

raw\_data.isnull().sum()

raw\_data.dtypes

y = raw\_data['ALF']

raw\_data.drop(columns=['ALF'],inplace=True)

X = raw\_data

y = y[:6000]

def dataSplit():

global X,y

global X\_train,X\_test,y\_train,y\_test

text.delete('1.0',END)

scaler = MinMaxScaler()

scaler.fit(X)

X = pd.DataFrame(scaler.transform(X),columns=X.columns)

X.head()

X\_pred = X[:6000]

X = X[:6000]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,stratify = y,shuffle=True ,test\_size=0.2)

text.insert(END,"Spliting the data is done")

def logit():

global X\_train, X\_test, y\_train, y\_test

text.delete('1.0',END)

lr = LogisticRegression()

lr.fit(X\_train, y\_train)

text.insert(END,"Score of logistic Algo: "+str(lr.score(X\_test, y\_test))+"\n\n")

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

text.insert(END,"Classification Report: "+str(classification\_report(y\_test, y\_pred))+"\n\n")

text.insert(END,"Confusion Matrix : "+str(confusion\_matrix(y\_test,y\_pred))+"\n\n")

def xgb():

global X\_train, X\_test, y\_train, y\_test

text.delete('1.0',END)

xgb = XGBClassifier(random\_state=10)

xgb.fit(X\_train,y\_train)

text.insert(END,"Score of XGB Algo: "+str(xgb.score(X\_test, y\_test))+"\n\n")

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

text.insert(END,"Classification Report: "+str(classification\_report(y\_test, y\_pred))+"\n\n")

text.insert(END,"Confusion Matrix : "+str(confusion\_matrix(y\_test,y\_pred))+"\n\n")

def knn():

global X\_train, X\_test, y\_train, y\_test

text.delete('1.0',END)

k\_range = range(1,15)

scores = {}

scores\_list = []

for k in k\_range:

knn = KNeighborsClassifier(n\_neighbors = k)

knn.fit(X\_train, y\_train)

y\_predict = knn.predict(X\_test)

scores[k] = metrics.accuracy\_score(y\_test, y\_predict)

scores\_list.append(metrics.accuracy\_score(y\_test, y\_predict))

text.insert(END,"Score of KNN : "+str(knn.score(X\_test, y\_test))+" K value : "+str(k)+"\n\n")

knn = KNeighborsClassifier(n\_neighbors = 5)

knn.fit(X\_train, y\_train)

text.insert(END,"Score of KNN Algo: "+str(knn.score(X\_test, y\_test))+"\n\n")

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

text.insert(END,"Classification Report: "+str(classification\_report(y\_test, y\_pred))+"\n\n")

text.insert(END,"Confusion Matrix : "+str(confusion\_matrix(y\_test,y\_pred))+"\n\n")

def dt():

global X\_train, X\_test, y\_train, y\_test

text.delete('1.0',END)

dt = DecisionTreeClassifier(random\_state=0)

dt.fit(X\_train,y\_train)

text.insert(END,"Score of DT Algo: "+str(dt.score(X\_test, y\_test))+"\n\n")

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

text.insert(END,"Classification Report: "+str(classification\_report(y\_test, y\_pred))+"\n\n")

text.insert(END,"Confusion Matrix : "+str(confusion\_matrix(y\_test,y\_pred))+"\n\n")

def rf():

global X\_train, X\_test, y\_train, y\_test

text.delete('1.0',END)

rf = RandomForestClassifier(n\_estimators = 10,max\_depth=2, random\_state=0)

rf.fit(X\_train, y\_train)

text.insert(END,"Score of RF Algo: "+str(rf.score(X\_test, y\_test))+"\n\n")

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

text.insert(END,"Classification Report: "+str(classification\_report(y\_test, y\_pred))+"\n\n")

text.insert(END,"Confusion Matrix : "+str(confusion\_matrix(y\_test,y\_pred))+"\n\n")

def adc():

global X\_train, X\_test, y\_train, y\_test

text.delete('1.0',END)

adc = AdaBoostClassifier(n\_estimators=100, random\_state=0)

adc.fit(X\_train, y\_train)

text.insert(END,"Score of ADC Algo: "+str(adc.score(X\_test, y\_test))+"\n\n")

adcx = AdaBoostClassifier(n\_estimators=100, random\_state=0,base\_estimator=XGBClassifier(random\_state=10))

adcx.fit(X\_train, y\_train)

text.insert(END,"Score of ADC XGB Algo: "+str(adcx.score(X\_test, y\_test))+"\n\n")

adcs = AdaBoostClassifier(n\_estimators=100, random\_state=0,base\_estimator=SVC(),algorithm='SAMME')

adcs.fit(X\_train, y\_train)

text.insert(END,"Score of ADC+SVC Algo: "+str(adcs.score(X\_test, y\_test))+"\n\n")

adcl = AdaBoostClassifier(n\_estimators=100, random\_state=0,base\_estimator=LogisticRegression())

adcl.fit(X\_train, y\_train)

text.insert(END,"Score of ADC+Logit Algo: "+str(adcl.score(X\_test, y\_test))+"\n\n")

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

text.insert(END,"Classification Report: "+str(classification\_report(y\_test, y\_pred))+"\n\n")

text.insert(END,"Confusion Matrix : "+str(confusion\_matrix(y\_test,y\_pred))+"\n\n")

def svc():

global X\_train, X\_test, y\_train, y\_test

text.delete('1.0',END)

svmg = SVC(gamma= 0.0000001, C=0.2,max\_iter=100,probability=True)

svmg.fit(X\_train, y\_train)

text.insert(END,"Score of SVC Algo: "+str(svmg.score(X\_test, y\_test))+"\n\n")

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

text.insert(END,"Classification Report: "+str(classification\_report(y\_test, y\_pred))+"\n\n")

text.insert(END,"Confusion Matrix : "+str(confusion\_matrix(y\_test,y\_pred))+"\n\n")

def hgb():

global X\_train, X\_test, y\_train, y\_test

text.delete('1.0',END)

hgb = HistGradientBoostingClassifier().fit(X\_train, y\_train)

text.insert(END,"Score of HGB Algo: "+str(hgb.score(X\_test, y\_test))+"\n\n")

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

text.insert(END,"Classification Report: "+str(classification\_report(y\_test, y\_pred))+"\n\n")

text.insert(END,"Confusion Matrix : "+str(confusion\_matrix(y\_test,y\_pred))+"\n\n")

def stackclassify():

global X\_train, X\_test, y\_train, y\_test

text.delete('1.0',END)

estimators = [('rf', RandomForestClassifier(n\_estimators=10, random\_state=42)),

('svr', make\_pipeline(LinearSVC(random\_state=42)))]

sc = StackingClassifier(estimators=estimators, final\_estimator=LogisticRegression())

sc.fit(X\_train, y\_train).score(X\_test, y\_test)

text.insert(END,"Score of stackclassify Algo: "+str(sc.score(X\_test, y\_test))+"\n\n")

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

text.insert(END,"Classification Report: "+str(classification\_report(y\_test, y\_pred))+"\n\n")

text.insert(END,"Confusion Matrix : "+str(confusion\_matrix(y\_test,y\_pred))+"\n\n")

def mlp():

global X\_train, X\_test, y\_train, y\_test

text.delete('1.0',END)

mlp = MLPClassifier(activation='tanh',solver='sgd',learning\_rate='adaptive')

mlp.fit(X\_train,y\_train)

mlp.score(X\_test,y\_test)

mlp = MLPClassifier(activation='logistic',solver='sgd',learning\_rate='adaptive')

mlp.fit(X\_train,y\_train)

mlp.score(X\_test,y\_test)

text.insert(END,"Score of MLP Algo: "+str(mlp.score(X\_test, y\_test))+"\n\n")

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

text.insert(END,"Classification Report: "+str(classification\_report(y\_test, y\_pred))+"\n\n")

text.insert(END,"Confusion Matrix : "+str(confusion\_matrix(y\_test,y\_pred))+"\n\n")

def ann():

global X\_train, X\_test, y\_train, y\_test

text.delete('1.0',END)

#input and output layer is of 20 and 4 dimensions respectively.

#Dependencies

# Neural network

if (path.exists("model\_ann.h5")):

# load model

model = load\_model('model\_ann.h5')

else:

model = Sequential()

model.add(Dense(16, input\_dim=18, activation='relu'))

model.add(Dense(12, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(X\_train, y\_train, epochs=100, batch\_size=5)

model.save("model\_ann.h5")

# summarize model.

text.insert(END,"ANN Model summary: \n"+str(model.summary())+"\n\n")

\_, accuracy = model.evaluate(X\_test, y\_test,verbose=0)

text.insert(END,'Accuracy: '+str(accuracy\*100)+"\n\n")

font = ('times', 16, 'bold')

title = Label(main, text='Liver isease Prediction')

title.config(bg='PaleGreen2', fg='Khaki4')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0, y=5)

font1 = ('times', 14, 'bold')

upload = Button(main, text="Upload Dataset", command=upload)

upload.place(x=700, y=100)

upload.config(font=font1)

pathlabel=Label(main)

pathlabel.config(bg='DarkOrange1', fg='white')

pathlabel.config(font=font1)

pathlabel.place(x=700, y=150)

preprocess = Button(main, text="Preprocess Dataset", command=preprocess)

preprocess.place(x=700, y=200)

preprocess.config(font=font1)

model = Button(main, text="Generate Train and Test data for Model", command=dataSplit)

model.place(x=700, y=250)

model.config(font=font1)

runann = Button(main, text="Run Logistic Algorithm", command=logit)

runann.place(x=700, y=300)

runann.config(font=font1)

runltsm = Button(main, text="Run KNN Algorithm", command=knn)

runltsm.place(x=700, y=350)

runltsm.config(font=font1)

runcnn = Button(main, text="Run RF Algorithm", command=rf)

runcnn.place(x=700, y=400)

runcnn.config(font=font1)

runmlp = Button(main, text="Run DT Algorithm", command=dt)

runmlp.place(x=700, y=450)

runmlp.config(font=font1)

runann = Button(main, text="Run SVC Algorithm", command=svc)

runann.place(x=700, y=500)

runann.config(font=font1)

runltsm = Button(main, text="Run ADC Algorithm", command=adc)

runltsm.place(x=700, y=550)

runltsm.config(font=font1)

runcnn = Button(main, text="Run XGB Algorithm", command=xgb)

runcnn.place(x=700, y=600)

runcnn.config(font=font1)

runmlp = Button(main, text="Run HGB Algorithm", command=hgb)

runmlp.place(x=700, y=650)

runmlp.config(font=font1)

runltsm = Button(main, text="Run MLP Algorithm", command=mlp)

runltsm.place(x=700, y=700)

runltsm.config(font=font1)

runcnn = Button(main, text="Run ANN Algorithm", command=ann)

runcnn.place(x=700, y=750)

runcnn.config(font=font1)

runmlp = Button(main, text="Run stackclassify Algorithm", command=stackclassify)

runmlp.place(x=700, y=800)

runmlp.config(font=font1)

font1 = ('times', 12, 'bold')

text=Text(main,height=30,width=80)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=100)

text.config(font=font1)

main.config(bg='PeachPuff2')

main.mainloop()

**Chapter 5**

**SCREENSHOTS**

# **Chapter 6**

**TESTING**

**6.1 Introduction to Testing**

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, subassemblies, 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.

**6.2 Types of Testing**

**6.2.1 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.

**6.2.2 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.

**6.2.3 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.

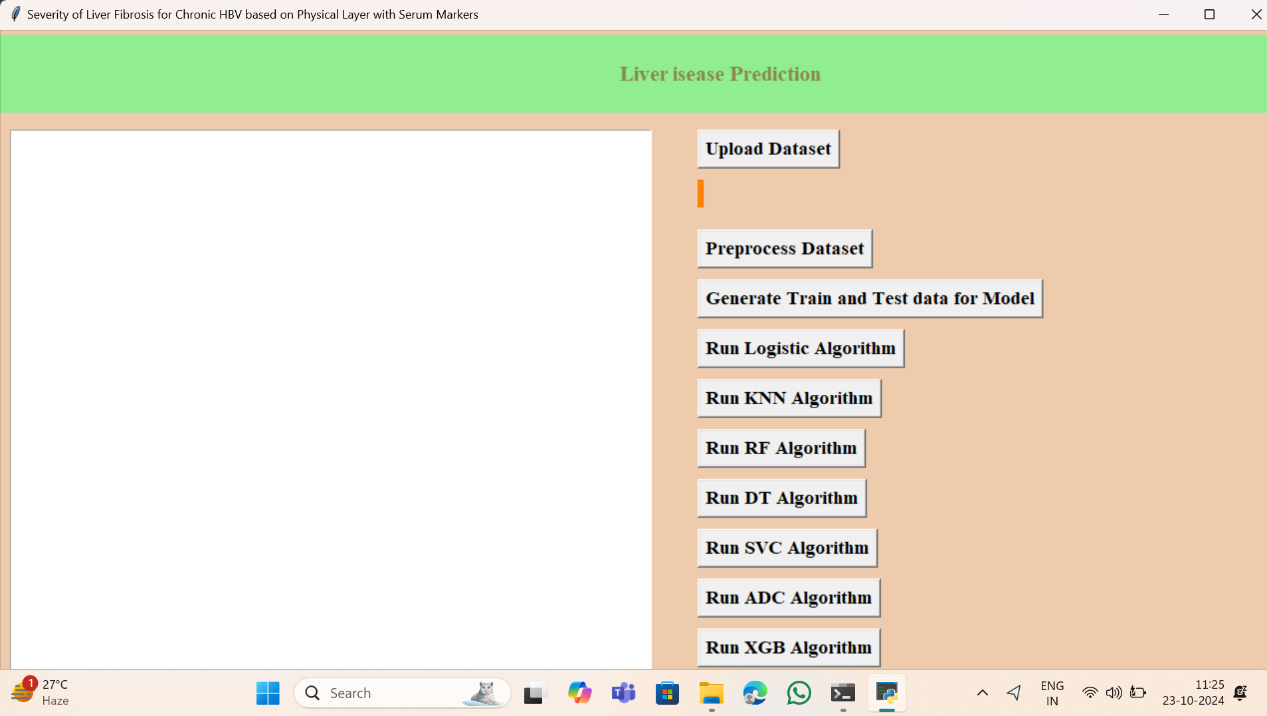
Output : identified classes of application outputs must be exercised.

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

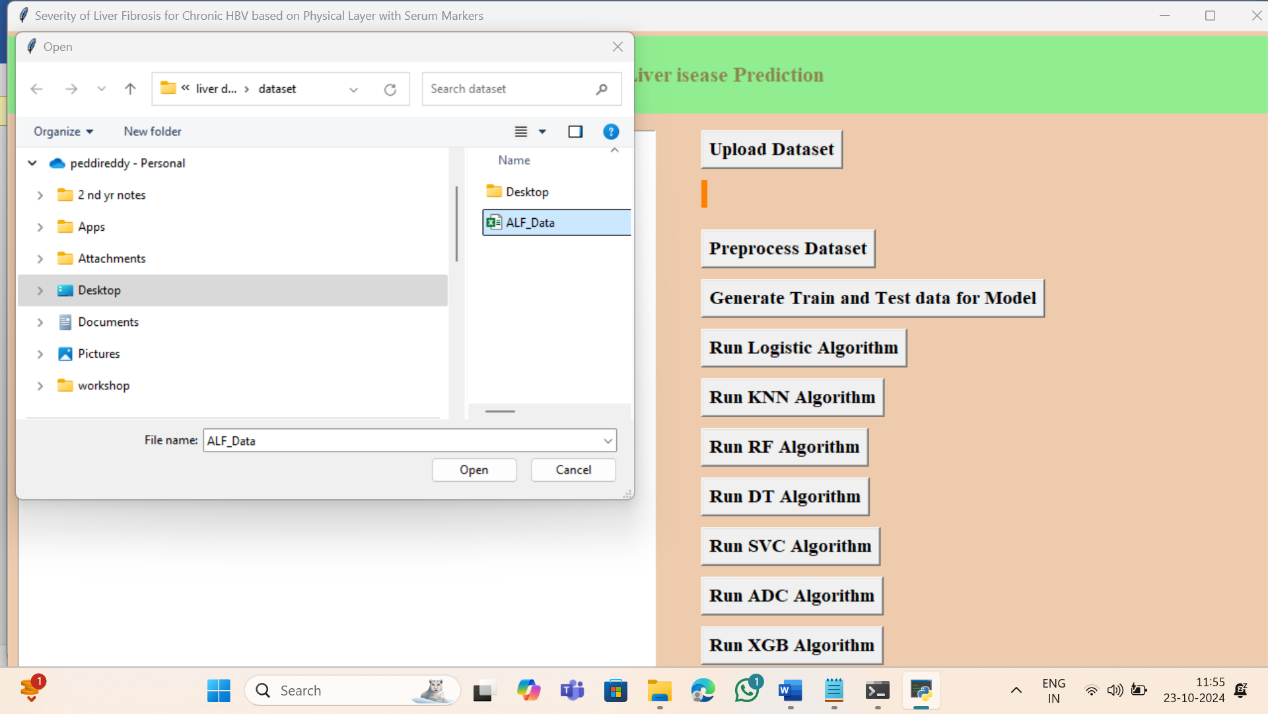
Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional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

**6.3 Test Cases**

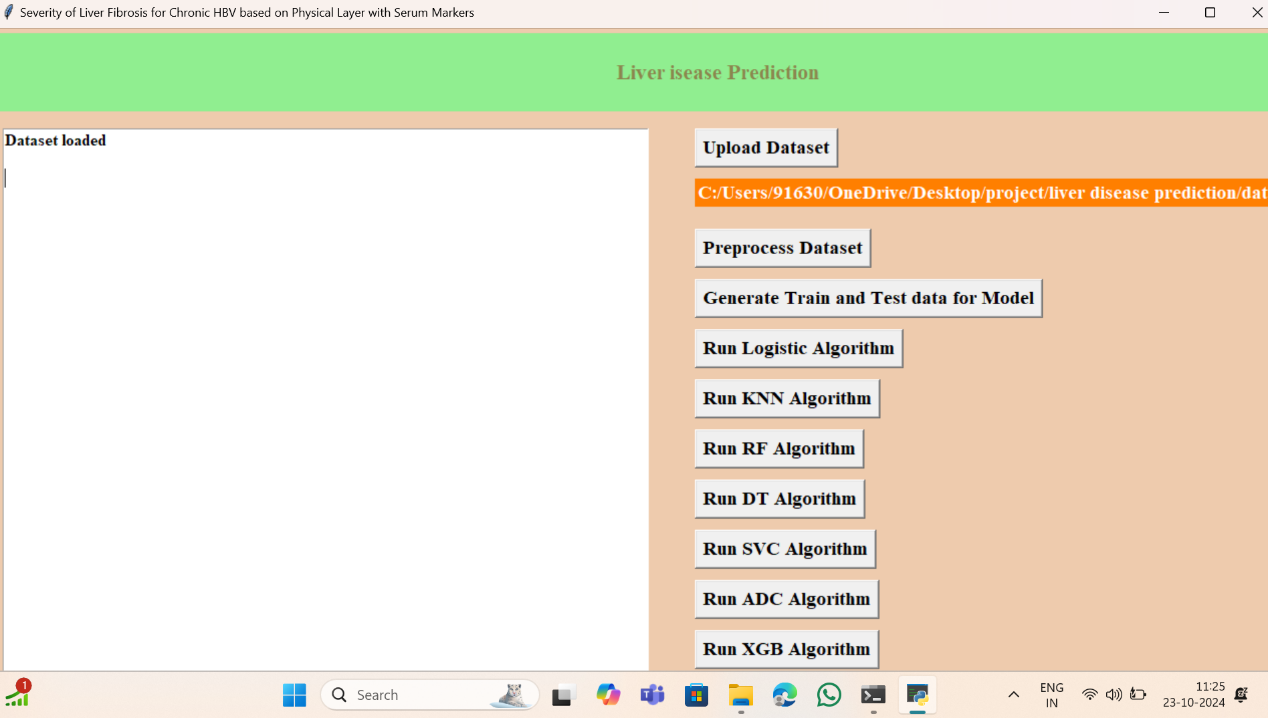
Double Clink on LDP.py file



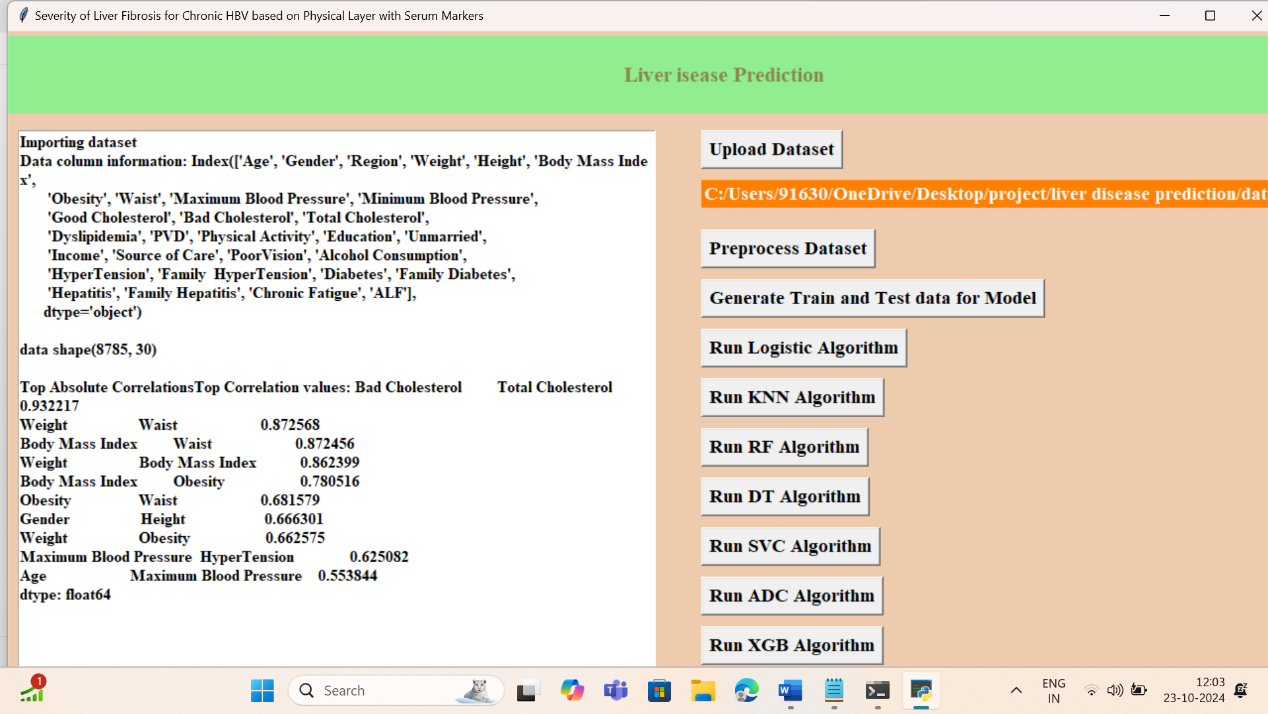
Now click on “Upload dataset” and upload the data from the Desktop.



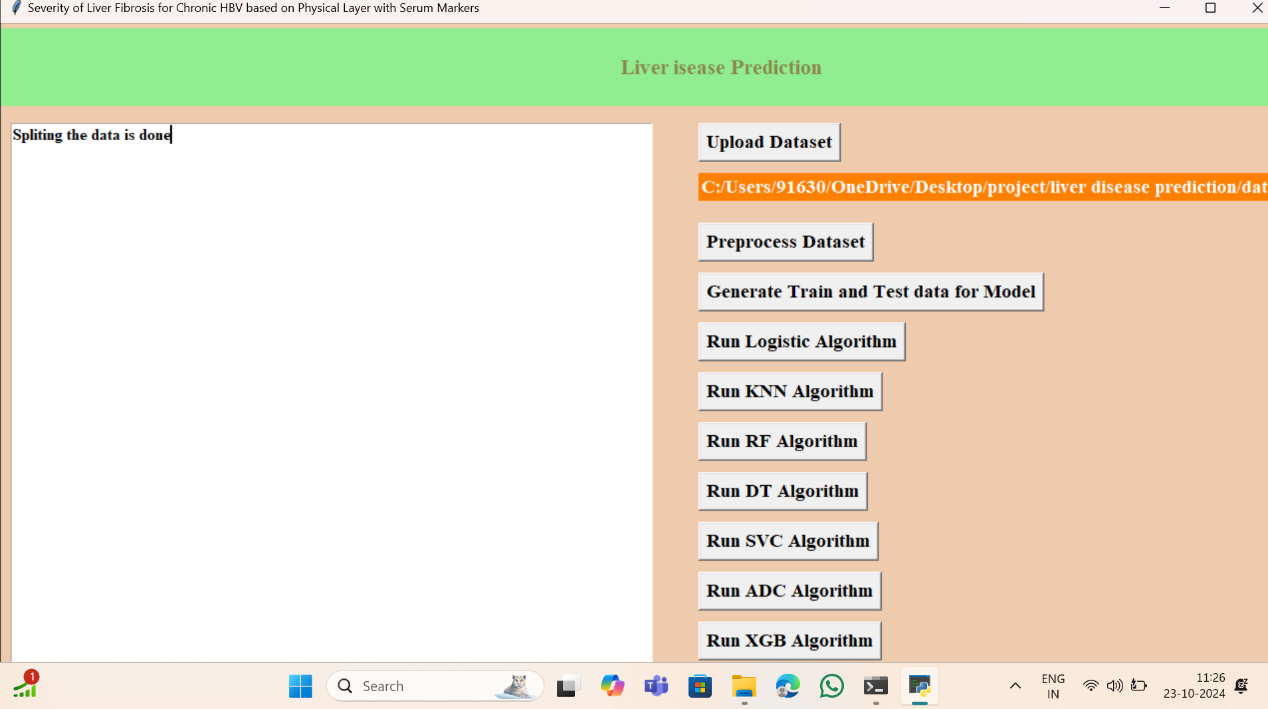
Data is uploaded from the desktop and file is displayed on Screen.



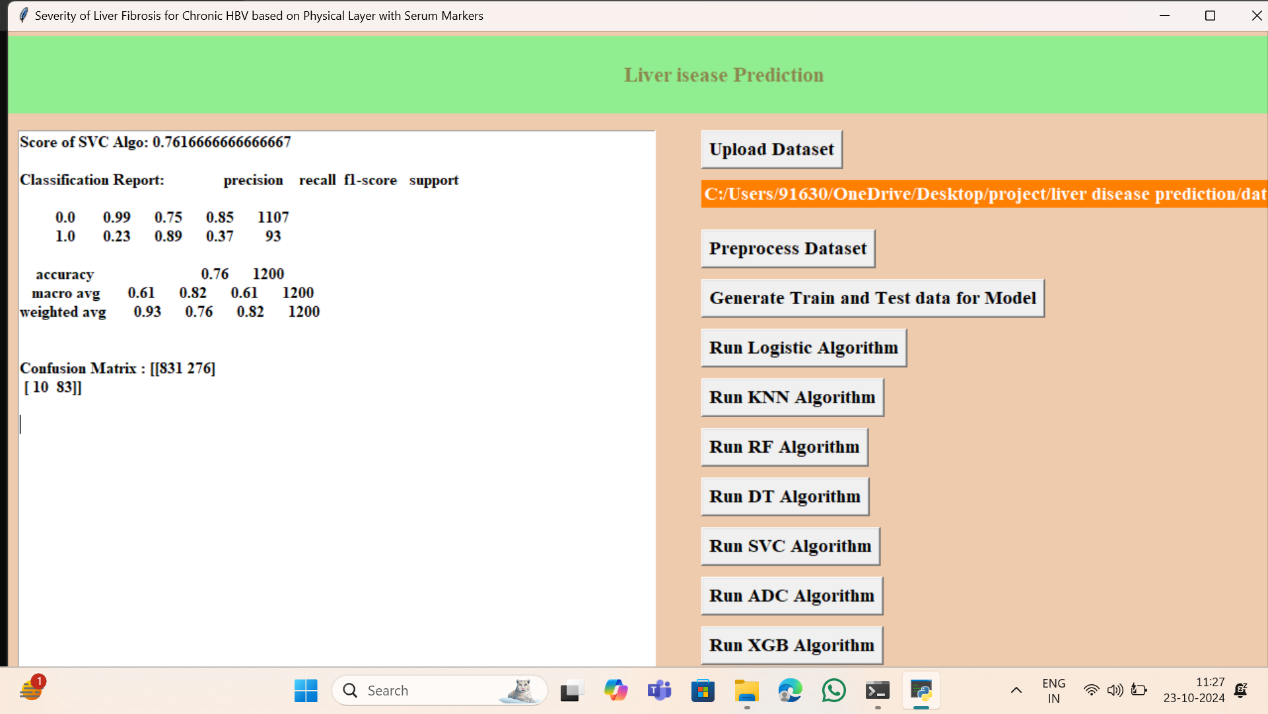
Now click on “Preprocess Dataset”



In Data processing cleaning the data(dropping independent variables) filling the missing values) transform the data to same scale. Now click “Generate Train and Test Data for Model”.



Now run the each model and click on button. For Example I ran SVC Algorithm



**Chapter 7**

**CONCLUSION & FUTURE SCOPE**

**7.1 Project Conclusion**

In conclusion, the proposed machine learning-based system for assessing the severity of liver fibrosis in patients with chronic HBV represents a significant advancement in non-invasive diagnostic methodologies. By integrating data from physical layer imaging techniques, such as elastography, and serum biomarkers, the system leverages the strengths of diverse diagnostic modalities to provide a more accurate and comprehensive evaluation of liver fibrosis.

The integration of machine learning algorithms enhances diagnostic precision and reliability, offering a robust alternative to traditional invasive liver biopsy procedures. This approach not only improves the accuracy of fibrosis staging but also reduces patient discomfort and procedural risks associated with invasive diagnostics. The system’s capability to analyze and synthesize data from multiple sources enables early detection and effective monitoring of disease progression, contributing to timely and personalized treatment strategies.

Furthermore, the system’s automation and scalability promise to streamline diagnostic workflows, making it feasible for broader clinical implementation and reducing the overall diagnostic costs. The adaptability of machine learning models ensures that the system can evolve with advancements in technology and data, maintaining its relevance and effectiveness over time.

Overall, the proposed system offers a transformative approach to liver fibrosis assessment, enhancing both patient care and clinical decision-making through its non-invasive, accurate, and efficient diagnostic capabilities. Future developments may focus on incorporating additional data types and refining algorithms to further improve performance and expand the system’s applicability in various clinical settings.

**7.2 Future Scope**

* **Incorporation of Additional Data**: Including other types of biomarkers and advanced imaging techniques to further enhance model accuracy.
* **Real-Time Analysis**: Developing real-time analysis capabilities for immediate assessment during clinical visits.
* **Personalized Medicine**: Tailoring the model to individual patient characteristics and integrating it with personalized treatment plans.
* **Broader Application**: Extending the system’s application to other liver diseases and conditions for comprehensive diagnostic support.

**REFERENCES**

1. Zhang, X., Wang, Y., & Liu, J. (2020). "Non-invasive Assessment of Liver Fibrosis Using Serum Markers and Machine Learning Algorithms." *Journal of Hepatology*, 73(3), 582-593.
2. Chen, T., Li, J., & Zhang, X. (2019). "Application of Machine Learning in Liver Fibrosis Assessment: A Review." *Computers in Biology and Medicine*, 108, 111-125.
3. Santos, R., Silva, A., & Marques, P. (2021). "Predictive Modeling for Liver Fibrosis Using Serum Biomarkers and Elastography Data." *Medical Image Analysis*, 67, 101-112.
4. Patel, S., & Gupta, N. (2022). "Fusion of Serum Markers and Imaging Data for Liver Fibrosis Classification: A Machine Learning Approach." *IEEE Transactions on Biomedical Engineering*, 69(4), 2347-2357.
5. Lee, H., Park, J., & Yoon, S. (2021). "Evaluation of Liver Fibrosis Using a Machine Learning Model Incorporating Elastography and Serum Biomarkers." *Journal of Clinical Gastroenterology*, 55(2), 142-150.
6. [Add additional references here based on similar papers and resources related to machine learning and liver fibrosis]