A PROJECT REPORT ON

ANTICIPATING HEPATIC DISORDER THROUGH MACHINE LEARNING CLASSIFICATION TECHNIQUES

Submitted in partial fulfillment of requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by:

G. SAI LAKSHMI	20091A05D2
M. GOWTHAMI	20091A0547
U. VENUGOPAL	21095A0516
P. VIGNESH	21095A0518

Under the Guidance of
Mrs. S. RUBIA PARVEEN M.Tech.

Assistant Professor, Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJEEV GANDHI MEMORIAL COLLEGE OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)
Approved by AICTE, New Delhi; Affiliated to JNTUA-Ananthapuramu,
Accredited by NBA (6-Times); Accredited by NAAC with 'A+' Grade (Cycle-3), New Delhi;
World Bank Funded Institution; Nandyal (Dist)-518501, A.P
(Estd-1995)

BATCH: 2020-2024

A PROJECT REPORT ON

ANTICIPATING HEPATIC DISORDER THROUGH MACHINE LEARNING CLASSIFICATION TECHNIQUES

Submitted in partial fulfillment of requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by:

G. SAI LAKSHMI	20091A05D2
M. GOWTHAMI	20091A0547
U. VENUGOPAL	21095A0516
P. VIGNESH	21095A0518

Under the Guidance of
Mrs. S. RUBIA PARVEEN M. Tech.,

Assistant Professor, Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJEEV GANDHI MEMORIAL COLLEGE OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)
Approved by AICTE, New Delhi; Affiliated to JNTUA-Ananthapuramu,
Accredited by NBA (6-Times); Accredited by NAAC with 'A+' Grade (Cycle-3), New Delhi;
World Bank Funded Institution; Nandyal (Dist)-518501, A.P
(Estd-1995)

YEAR: 2023-2024

Rajeev Gandhi Memorial College of Engineering &Technology

(AUTONOMOUS)

Approved by AICTE, New Delhi; Affiliated to JNTUA-Ananthapuramu,
Accredited by NBA (6-Times); Accredited by NAAC with 'A+' Grade (Cycle-3), New Delhi;
World Bank Funded Institution; Nandyal (Dist)-518501, A.P

(ESTD – 1995)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that **G. SAI LAKSHMI** (*20091A05D2*), **M. GOWTHAMI** (*20091A0547*), **U. VENUGOPAL** (*21095A0516*) and **P. VIGNESH** (*21095A0518*) of IV- B. Tech II- semester, have carried out the major project work entitled "**ANTICIPATING HEPATIC DISORDER THROUGH MACHINE LEARNING CLASSIFICATION TECHNIQUES**" under the supervision and guidance of **Mrs. S. RUBIA PARVEEN,** Assistant Professor, CSE Department, in partial fulfillment of the requirements for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering** from **Rajeev Gandhi Memorial College of Engineering & Technology (Autonomous)**, Nandyal is a bonafied record of the work done by them during 2023-2024.

Project Guide	**Head of the Department**
Mrs. S. Rubia Parveen **M.Tech.**	**Dr. K. Subba Reddy** **M.Tech, Ph.D.**
Assistant Professor, Dept. of CSE	**Professor, Dept. of CSE**

Place: Nandyal
Date:

External Examiner

Candidate’s Declaration

We hereby declare that that the work done in this project entitled **“ANTICIPATING HEPATIC DISORDER THROUGH MACHINE LEARNING CLASSIFICATION TECHNIQUES”** submitted towards completion of major project in *IV Year II Semester of B. Tech (CSE)* at the **Rajeev Gandhi Memorial College of Engineering & Technology**, Nandyal. It is an authentic record our original work done under the esteemed guidance of **Mrs. S, Rubia Parveen,** Assistant Professor, Department of **Computer Science and Engineering**, RGMCET, Nandyal.

We have not submitted the matter embodied in this report for the award of any other Degree in any other institutions for the academic year 2023-2024.

By

G. SAI LAKSHMI

M. GOWTHAMI

U. VENU GOPAL

P. VIGNESH

Dept. of CSE,
RGMCET.

Place: Nandyal
Date:

ACKNOWLEDGEMENT

We manifest our heartier thankfulness pertaining to your contentment over our project guide **Mrs. S. Rubia Parveen,** Assistant Professor of Computer Science Engineering department, with whose adroit concomitance the excellence has been exemplified in bringing out this project to work with artistry.

We express our gratitude to **Dr. K. Subba Reddy garu,** Head of the Department of Computer Science Engineering department, all the **Teaching Staff Members** of the Computer Science Engineering department of Rajeev Gandhi memorial College of Engineering and Technology for providing continuous encouragement and cooperation at various steps of our project successful.

Involuntarily, we are perspicuous to divulge our sincere gratefulness to our Principal, **Dr. T. Jaya Chandra Prasad garu,** who has been observed posing valiance in abundance towards our individuality to acknowledge our project work tangentially.

At the outset we thank our honourable **Chairman Dr. M. Santhi Ramudu garu,** for providing us with exceptional faculty and moral support throughout the course.

Finally we extend our sincere thanks to all the non- teaching **Staff Members** of CSE Department who have co-operated and encouraged us in making our project successful.

Whatever one does, whatever one achieves, the first credit goes to the **Parents** be it not for their love and affection, nothing would have been responsible. We see in every good that happens to us their love and blessings.

BY

G. SAI LAKSHMI	(20091A05D2)
M. GOWTHAMI	(20091A0547)
U. VENU GOPAL	(21095A0516)
P. VIGNESH	(21095A0518)

ABSTRACT

Hepatitis is a global health problem accounting for a significant portion of global mortality. Although common, liver diagnosis is still a difficult and often expensive operation that requires complex clinical procedures and expertise. Machine learning (ML) technology holds promise against this challenge by using a data-driven approach to predict and diagnose diseases more accurately and efficiently. Here we aim to develop an effective liver disease prediction model by applying the power of machine learning algorithms on the Indian Liver Disease Patient Dataset. Using a variety of supervised learning algorithms logistic regression, K-nearest neighbors, support vector machine, and random forest, we aim to increase accuracy and ultimately improve patient outcomes.

Additionally, we created a web interface using the lightweight Python web Flask to deliver training models and provide users with a seamless and intuitive liver prediction platform. We saw differences in accuracy between the tested algorithms some models perform better than others. After rigorous analysis and comparison, we identified the Logistic Regression provide high accuracy this prediction model is implemented as a web application using Flask.

The user interface provides physicians and patients with a simple tool to predict liver disease, the ability to make early diagnoses, and interventions to improve patient outcomes and reduce medical costs. It has the ability to change liver disease. Using the Indian Liver Disease Dataset and employing highly trained observers, we developed a robust predictive model to improve the accuracy and accessibility of liver disease diagnosis. The integrated user interface further simplifies the deployment and use of predictive models, bridging regulatory challenges between machine learning technology and end users.

Keywords: *Flask interface, Logistic Regression Hepatitis. Liver Disease Dataset.*

CONTENTS

LIST OF ABBREVIATIONS

LR	:	Logistic Regression
SVM	:	Support Vector Machine
KNN	:	K-Nearest Neighbor
ILPD	:	Indian Liver Patient Dataset

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The liver is the largest organ in the human body and is important for digesting food and removing toxins from the body. Drug and alcohol use can harm the mind and harm people. There are many types of liver diseases, such as: Liver disease and cirrhosis are the leading causes of death. According to the Global Burden of Disease (GBD) project published in BMC Medicine, one million people died from cirrhosis and millions more from liver cancer in 2010.

Machine learning has had a major impact on biomedical liver disease prediction and diagnosis. Machine learning improves the purpose of decision-making while providing security for the detection and prediction of diseases of high importance in the biomedical field. Using machine learning, medical problems can be easily solved and diagnostic costs can be reduced.

The importance of this project is to better predict the results and reduce the cost of diagnosis in treatment. Therefore, we use different types of classification techniques to classify patients with and without liver disease. This project mainly focus on detecting whether a person is suffering from liver disease or not and is predicted through webpage which is deployed by flask.

1.1 Problem Statement:

To create and actualize Machine Learning approaches that learns the common designs of the Liver illnesses, and a liver infection demonstrate is trained. To bring way better proficiency to the calculations which will be utilized in finding the precision of the liver infections.

The most accurate model is used in flask application in order to provide webpage for the users where they can enter values and know whether their liver is in good condition or not. Mainly the interface makes the user to know that they are suffering with liver disease or not and is demonstrated with the results in the form of webpages for both liver and non liver patients

1.2 Objectives:

i To discover most extreme among the Machine Learning Algorithms.

ii To distinguish whether a individual has liver infection or not.

iii To detect the liver disease early so that the person health can be saved.

iv The main goal is to detect liver disease early before it becomes more serious. Early diagnosis can improve treatment outcomes and increase the chances of success.

v Machine learning models aim to provide accurate diagnosis of liver disease based on various patient information such as medical history, diagnosis, examination and genetics.

vi The purpose of the data is to reduce misdiagnoses and provide recommendations to doctors. Identifying high-risk patients provides the opportunity for intervention and prevention.

vii Ensuring patients understand the risks of liver disease so that they can make informed decisions about their health, adopt healthy lifestyles, and implement recommended treatment.

viii The main objective is previously there are no user interface for the patients to check whether they are suffering with liver disease or not here we used various algorithms and the most accurate values providing algorithm is saved and then results are predicted through webpages,

2. SYSTEM ANALYSIS

2.1 Existing system:

Bacteria and alcohol can cause liver damage such as liver disease, cirrhosis, tumors and cancer. Liver disease and cirrhosis are leading causes of death worldwide, making liver disease a major public health problem. More than 2 million people die from liver disease every year worldwide.

In 2010, one million people died from cirrhosis and one million from liver cancer. Available machine learning methods include Light GB, CNN, KNN, and SVM. In the past, machine learning algorithms focused solely on accuracy when predicting liver disease. However, no user interface has been developed for this purpose

2.1.1 Drawbacks of existing system:

i Light GB strategy is it is contract client based. This shows that the light GB concept is based on customer contract and in some cases can limit its flexibility or scalability.

ii In CNN testing exactness was less than preparing accuracy. Low testing accuracy compared to the training accuracy of a convolutional neural network model often indicates overfitting, where the model remembers the data rather than learning general patterns.

iii K-Nearest Neighbor affectability is less than Calculated Regression. K-Nearest Neighbors is more accurate than regression this suggests that KNN cannot do a good job of capturing relationships in the data or handling noise with regression models.

iv Support vector machine (SVM) algorithm may encounter scalability issues when processing large data sets because the computational resources and time required to train the SVM model also increase. As a result, SVMs can be inefficient or ineffective when processing large datasets, and other methods.

v There is no user interface for the patient to check about their liver condition.

2.2 Proposed Method:

The recommended framework was made with the objective of making an interface utilizing the algorithm which is more accurate that can anticipate whether a person has liver illness or not. The proposed approach incorporates collecting information from the client. Pre-processing the information is the another organize once information collection is performed.

Preprocessing is the prepare of cleaning the information such as changing content to lowercase, and disposing of invalid values. At that point, utilizing the preprocessed information, we prepare numerous models. We may evaluate the prepared models utilizing measures such as exactness, review, and F1-score. Finally, the patient comes to know whether they are suffering with liver disease or not through the interface which is made by flask.

2.2.1 Advantages of the proposed system:

i **Improves Accuracy:**

Using advanced machine learning algorithms and evaluating multiple models, the system is designed to provide accurate predictions of liver disease. This fact can lead to better decisions and early detection of health problems.

ii **Provides user Friendly Interface:**

The system has a user-friendly interface designed using Flask, making it easier for patients to use. This increases user engagement and encourages people to take care of their health.

iii **Helps in early detection of disease:**

The system provides timely prediction of liver disease, enabling early detection and intervention, potentially improving patient outcomes and reducing associated healthcare treatment costs.

3. FEASIBILITY STUDY

Achievability ponder is an vital stage in the program advancement prepare. The fundamental objective of the achievability consider is to test the Specialized, Operational and Temperate possibility for including unused modules and investigating ancient running framework.

Feasibility study should be performed on this parameters:

- Technical Feasibility
- Operation Feasibility
- Economic Feasibility

3.1 Technical Feasibility:

The technical issue usually raised during the feasibility stage of the investigation includes the following:

i Assessing the availability, quality and accessibility of datasets relevant to the prediction of liver disease.

ii Make sure there is enough data to train and test your machine learning model.

iii Evaluate data quality, consistency, and completeness to ensure reliable estimates.

iv Measure the processing power, memory, and storage capacity required for model training and evaluation. Consider leveraging cloud services for scalability and flexibility. Develop a user-friendly approach to data entry, modeling and visualization of results.

v Use pre-rendered data pipelines, machine learning algorithms, and test models within the Flask framework.

vi Model training instruction and evaluation: Machine learning for data preprocessing, feature selection, model training, and hyperparameterization setting using lines.

vii Evaluate the performance of the training model using appropriate metrics such as accuracy, precision, recall, and F1 score.

viii Cross-validation technique was used to test the performance of the model.

ix Predictive bugs are extensively tested to identify and resolve software bugs or inconsistencies. Conduct tests, integration tests, and end-to-end testing to verify system functionality, accuracy, and reliability before deployment

3.2 Operational Feasibility:i

User-friendly:

Provides an interface which is helpful for the user and also helps in easy navigation where the webpage allows user to enter the values and get the prediction about the liver disease regarding whether the person is suffering liver disease or not.

ii Reliability:

Based on the data entered by the user the most accurate machine learning model will predict the results which are accurate.

iii Security:

The web server and database server should be protected from hacking, virus. Maintainability: The system architecture is based on modular design which is helpful in easy maintenance and also helpful in updates.

iv Ease of Maintenance:

The system architecture adopts a design that facilitates maintenance and updating. Components such as predefined databases, machine learning models, and user interface modules can be customized and developed, tested, and deployed independently.

3.3 Economic Feaibility:

The system can be technically improved and if installed it will be used and still be a great asset to the organization. In terms of economic feasibility, the cost of developing the system is evaluated against the best results that the new system will bring. Financial benefits must equal or exceed the costs. The system is economically suitable. Does not require any additional hardware or software.

i Development costs: Include costs related to software development, data collection, preprocessing, algorithm implementation, and interface design.

ii Hiring data scientists, developers, and UI/UX designers can involve significant costs.

iii Cloud services such as AWS, Azure or Google Cloud Platform have potential but will incur regular operating costs.

iv Cleanse and anonymize sensitive data to comply with privacy laws such as HIPAA. These fees will vary depending on the complexity and size of the file.

v Regular maintenance, updates, bug fixes and security patches add value over time.

vi This can reduce medical costs associated with intensive care, hospitalization cost.

4. SYSTEM REQUIREMENT SPECIFICATION

4.1 Requirement Analysis:

Requirements analysis is a software engineering project that bridges the gap between system-level software deployment and software design. Requirements analysis identifies software functions and functions and represents the interaction between software and other system components. The specifications required by the user can be formal or informal. In the legal system, the user clearly defines the purpose of the software.

This provides a good basis for software engineers to conduct requirements analysis. Informally, the purpose and output are not clearly defined by the user. Software engineers are responsible for achieving goals and results by interacting with multiple users.

4.2 Software Requirements Specification (SRS):

Software requirement specification that is the requirements specification for a software system, is a complete description of the behavior of the system to be designed and may include the use of a set of information that describes the interaction between users and the software. It also has non-functional requirements.

Non-functional requirements impose restrictions on design or use. We need a clear and precise understanding of the product to be created to meet the needs. This is planned after detailed communication with the team and customers. A good SRS defines how an application interacts with hardware, other programs, and human users in different parts of the world.

4.3 Functional Requirements:

i Users must be able to register for an account or log in securely to access the system. The system must provide user-friendly communication so that users enter the values and get results accordingly.

ii After collecting the data, the system must use a combination of learning models to create a prediction of the user's liver disease.

iii The system must present the results for the user in a clear and easy-to-understand format through the Flask web interface

iv The system must handle errors appropriately and provide users with error messages when problems occur during data entry, progress, or for predicting results.

4.4 Non-Functional Requirements:

i The system must work well with a good network bandwidth to ensure fast data transfer between the client and the server.

ii It should be designed to handle changes in bandwidth without significant disruption. Technologies such as scaling and caching can be used to reduce problems caused by collisions.

iii It must be thoroughly tested to ensure that it works as intended and does not cause physical instability or invalid data. Data must be received, processed and stored without loss or damage during transmission or processing.

iv The system must protect against security threats such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

v Input validation, parameterized queries, and security headers should be used to reduce risks.

vi No harm is expected from the use of the application either to the OS or any data that resides on the client system.

4.5 Software Requirements:

Operating System	:	**Windows 7/8/10**
Libraries Used	:	**NumPy, Pandas, Scikit-Learn, Matplot, pickle**
Technology	:	**Python 3.6+, HTML, CSS**
Interface	:	**Flask**

4.6 Hardware Requirements:

Processor	:	**13/Intel Processor**
Hard Disk	:	**160GB**
RAM	:	**8GB**

5. SYSTEM ARCHITECTURE

5.1 System Architecture description:

i In the first step we collect the dataset and do data pre-processing, clean and analyze data analysis

ii Then we do machine learning algorithms logistic regression, random forests, support vector machines, K nearest models are created.

iii The most accurate predicting model is saved and deployed with flask.

iv Users can access enter the requried details and check results and find out if they have liver disease or not

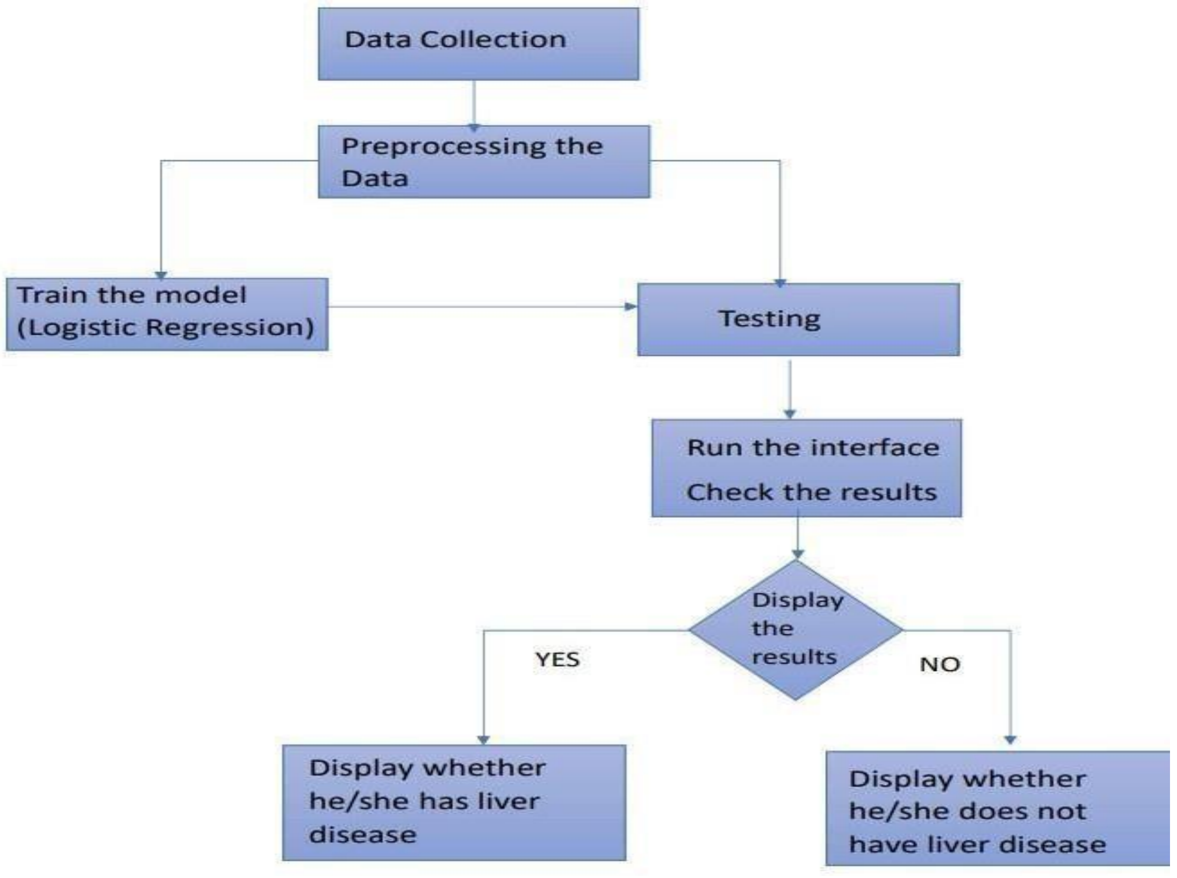

Data Collection
Preprocessing the Data
Train the model (Logistic Regression)
Testing
Run the interface
Check the results
Display the results
YES
NO
Display whether he/she has liver disease
Display whether he/she does not have liver disease

Fig 5.1: System Architecture Flow chart

6. MODULES

There are mainly two modules with respect to this project

6.1 User Module:

Users will find a user-friendly interface where they can enter age, gender, total bilirubin,alkaline phosphatase, total protein and albumin levels. This simplifies the process of providing relevant medical information and makes it easier to know the results for users.

Once these features are entered, the system uses advanced machine learning algorithms to analyze the data and predict the likelihood of liver disease. Users can access this session from any accessible website, making monitoring liver diseases easy and convenient. Finally, users can benefit from real-time information about liver diseases, allowing them to make informed decisions about their health.

6.2 Web application:

Once the most accurate prediction was determined, the web application was implemented using Flask. Users access relevant information through a user-friendly interface. Once the information is submitted, it is sent to the server for processing. The server then uses a distributed algorithm to predict the user's liver disease.

Whether it is liver disease or non-hepatic disease notification, the answer to the question is presented to users on the web page. For patients with liver disease, the corresponding words appear to provide predictive suggestion or suggestion. Likewise, warnings may also occur in patients without liver disease. This simple system allows users to quickly get personalized results about their health, aiding in early diagnosis and informed decision-making.

7. TECHNICAL DESCRIPTION

7.1 FLASK:

Flask is a microweb framework written in Python. It is classified as a microframework because it does not require special tools or libraries. It lacks the database abstraction layer, form validation, or other functionality provided by existing third-party libraries. However, Flask supports extensions that can add application functionality as if it were implemented in Flask itself. Extensions are available on connection mappers, form validation, upload handlers, various open authentication technologies, and various networking tools.

7.1.1 History:

Flask was first released in 2010 by Armin Ronacher as an April Fool's Day prank, but has become very popular among developers. Ronacher had previously created another Python website called Pocco but found it too difficult and decided to create a simpler alternative. Flask was inspired by Sinatra, the Ruby website known for its simplicity and minimalism.

Flask was initially released under the BSD license, which allows free and open source use and distribution. Over the years, Flask has become one of the most popular Python web applications with a large and dynamic developer community contributing to its development and maintenance.

7.1.2 Advantages:

- Scalable: Size is everything, and Flask's microframework status means you can use it to scale up technical projects like web applications at lightning speed. It's the best choice if you want to build an app that starts small but can grow quickly and in a direction you're not at all sure about. It is easy to use and has very few dependencies, allowing it to run smoothly even when scaling.
- Flexibility: This is the main feature of Flask and one of its best features. To paraphrase one of Python's Zen principles, simplicity is better than complexity because it can be easily modified and ported. The ability to build small web applications means it's more flexible.

7.1.3 Features of FLASK:

i Development server and debugger

ii Integrated support for unit testing

iii RESTful request submission

iv Using Jinja templates

v Supports secure cookies (client sessions)

vi 100% WSGI 1.0 compliant

vii According to Unicode it works

viii Google App Engine compatibility

ix Extensions available to extend functionality

7.1.4 Terminologies related to Flask:

render_template:

The render_template function is usually used to render the HTML template in the function and pass dynamic data to it. It takes as its first argument the name of the template archive and an optional keyword argument representing the dynamic file to be sent to the template. They are usually stored in a directory called templates in the Flask directory. When calling render_template, Flask will automatically search for templates in this directory.

These files may contain variables, lists, dictionaries, or other Python objects. In templates, these variables can be entered using Jinja2 template syntax. Jinja2 allows embedding Python code and sentences into HTML templates to easily create dynamic content. Variables, loops, events, and primitives are some of the features supported by Jinja2.

request:

The request object is a global variable that represents the HTTP request sent by the client to the server. It provides access to various objects and information related to the request, allowing the server to operate and respond accordingly.The HTTP method used in the request (e.g. GET, POST, PUT, DELETE).Query parameter accessible via the args attribute in the URL.

app.route:

@app.route() is a decorator that defines a route for processing HTTP requests. Flask allows to specify URL patterns to find specific functions in the application. @app.route() is placed above the visual description and uses at least one canonical URL of the route. The route created by the @app.route() function receives the request.

However, we can specify other HTTP methods using the method parameter.@app.route() accepts other parameters in addition to routes, such as presets, strict_slashes and subdomains, to customize routing behavior. Basic elements in the request allows us to define the behavior of the route and manage the HTTP request, allowing us to create simple web applications.

app.run:

app.run() is the method used to run a Flask application.app.run() is usually called at the end of a script to start the Flask development server and run the application. Essentially, this mode provides useful debugging information and automatically restarts the server when code changes are detected. This is enabled by passing debug = True as a parameter to the app.run() function.

Different hosts and ports by passing the host and port to the app.run() function.Overall, app.run() is a useful way to run a Flask application during development and provides a quick and easy way to test and debug Flask program code. However, it is not suitable for production deployment due to its limitations and lack of security compared to high-end servers.

name:

In Flask,___name___is a special variable that holds the name of the current Python module. It is mainly used to help Flask determine the root of the application and control the execution of some blocks of code. When creating a Flask application, we typically pass_name_as a parameter to launch the application.

Also, if__name___== '___main___'block is used to check whether the script is executed directly, it will allow some code to be executed only when the script is executed as kill service. In general, __name__plays an important role in initializing and controlling the behavior of a Flask application.

7.2 Python Libraries:

Pandas:

Pandas provides us with many pins and container files. It allows you to easily configure, search, represent and manage devices. Smart alignment and indexing in Pandas ensures the files are perfectly organized and saved. It allows to easily organize, search, represent and manage the data. People with programming experience can also use it easily.CSV, HDF5 and many other formats. In fact, we can use Pandas to merge different repositories at the same time.

NumPy:

NumPy arrays provide routine arithmetic for large data sets. NumPy makes performing these tasks easier and hassle-free. Also functions like discrete Fourier transform, general linear algebra etc. This python package provides useful tools. We can easily integrate NumPy with programming languages such as C, C++ and Fortran code. Both allow users to work faster.

Seaborn:

Seaborn is a recommended library for drawing graphs in Python. It provides beautiful patterns and color palettes to make the graphics more attractive. It is built on top of the high-level matplotlib library and is tightly integrated with pandas data model. It provides a dataset-oriented API so that we can switch between different views of the same variable to better understand the data.

Pickle:

Pickle is a powerful Python tool that allows you to save ML models, reduce long iterations, and specify, distribute, and replicate machine learning models. Most data scientists working in ML use Pickle or Joblib to save ML models for future use. Although it is often associated with recording and reproducing machine learning models, it can be applied to any type of material.

Scikit:

Scikit-learn (Sklearn) is the most useful and powerful learning library in Python. It provides a variety of useful tools for machine learning and statistical models, including distribution, regression, covariance, and dimensionality through an interface in Python. This function library is written exclusively in Python and built for NumPy, SciPy, and Matplotlib.

7.3 Machine Learning related terminology:

LabelEncoder:

This function in machine learning, specifically in libraries like scikit-learn, is used to perform label encoding on categorical data.LabelEncoder() is a class provided by scikit-learn's preprocessing module that encodes categorical labels with numerical values. It converts each unique label into an integer representation.First, we create an instance of the LabelEncoder class.Then, we use the fit_transform() method to fit the encoder to your categorical data and transform the labels into numerical values.

SimpleImputer:

This method in machine learning is often found in libraries like scikit-learn to deal with missing data.SimpleImputer is a class provided by scikit-learn's impute module that uses a special strategy to impute missing values in the dataset. First, we create an instance of the SimpleImputer class by specifying the assignment concept (such as mean, average, maximum value, or constant value). for dataset methods that calculate appropriate statistics (e.g. mean, median) as parameters.

DataFrame:

DataFrame methods are functions provided by libraries such as pandas to manage, analyze, and prioritize tabular data.DataFrame methods allow a variety of operations on data tables.

Data operations performed are methods add, delete and modify rows and columns. Remove duplicates and replace files. Most of the methods accessed by DataFrame objects use the pandas library.

The DataFrame method provides a high-level interface that can handle the data. It provides a wide range of data usage, analysis and pre-processing. DataFrame method to handle outliers, and data inconsistencies. It is an important tool for effective data use, analysis and prioritization.

boxplot:

In machine learning, a box plot is a graphical way to visualize data distribution and identify potential outliers.Box plots, also known as bins provide a brief overview of data distribution by presenting important statistics such as medians, quartiles, and group ability values. The lines in the box represent the average.

They show the distribution of information. It is used to compare different groups in data. Visual presentation. in data sets that are multimodal or have highly skewed distributions. Interpreting box plots requires an understanding of the statistical concepts they represent, such as quartiles and outliers.

Confusion matrix:

In machine learning, the confusion matrix is a table that shows the performance of the classification model by comparing the predicted class with the data class.A confusion matrix consists of rows and columns that represent actual and predicted groups, respectively. Each cell in the matrix represents the number of examples in which the predicted classes match (true positive and negative) or differ from the true classes (true positive and negative).False Negative (FN): An instance where the model incorrectly predicts the negative class (type II error).

They can provide insight into the strengths and weaknesses of the model, such as its ability to identify different groups and its tendency to make certain types of errors. Comparisons can be made between predictions and actual class enrollments, making it easier to evaluate and improve the model. Additional metrics such as recall and F1 score provide a more comprehensive understanding of model performance.

iloc:

The iloc function is a method provided by the pandas library to sort and select data in a DataFrame based on an equality function.iloc is used to select rows and columns in a DataFrame based on numeric functions rather than letters or names. Integer indexing and slicing to select multiple rows or rows.iloc is an important tool in machine learning to select and extract data from a DataFrame based on numerical operations, making the data useful and analytical.

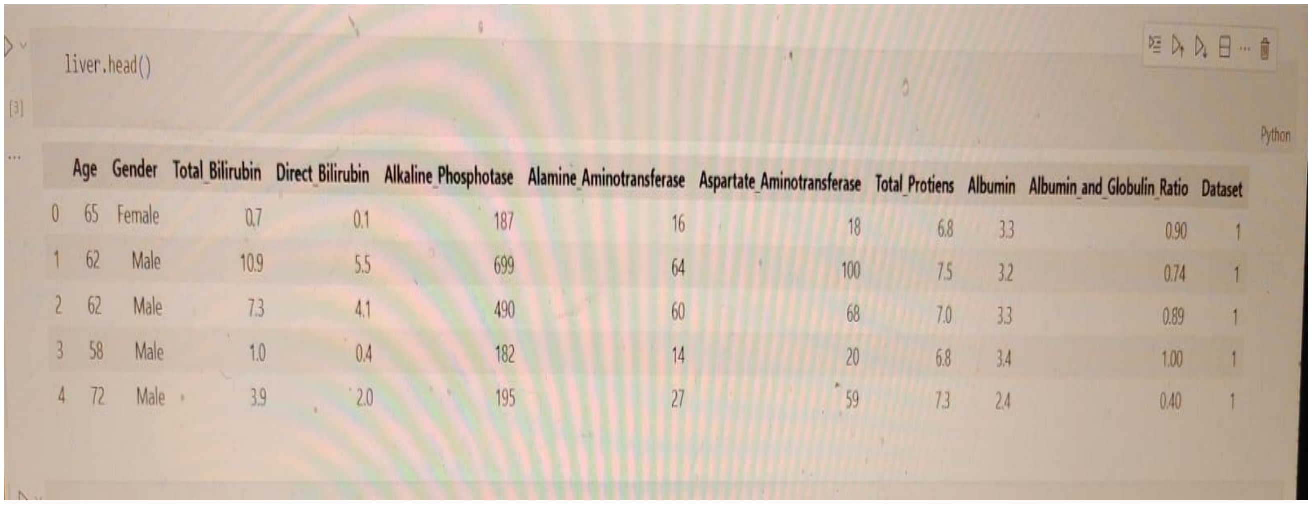
Facetgrid:

FacetGrid is a graph provided by the seaborn library used to create a subgraph grid based on a hierarchy of one or more variables. FacetGrid is often used to visualize relationships between different subsets of data sets. It allows us to create a line chart where each subplot represents a different group or combination of groups.we can use Seaborn's chart functions (such as map, map_dataframe, or map_data) to map the plan to the grid.

8. METHODOLOGY

8.1 Data Collection:

The data is collected from Kaggle of Indian Liver patient dataset which contains various attributes like described below. The dataset contains 583 rows and 11 columns



```
liver.head()
```

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	Dataset
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	1
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1

Fig 8.1 Attributes of dataset

8.2 Data Preprocessing:

First Import all the necessary libraries and load the dataset using read csv method.

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

sns.set(style="white", color_codes=True)

import warnings

warnings.filterwarnings("ignore")

import os

Data preprocessing is the process of preparing raw data and inserting it into machine learning models. This is the first important step in building a machine learning model. Before any work is done on the data, it must be cleaned and stored in a format. We use pre-processed data for this purpose. Load the dataset using csv method.

liver = pd.read_csv("C:\sai\indian_liver_patient.csv")

liver.columns

The above code describes all the columns present in the dataset.

8.3 Data Cleaning:

Data cleaning involves identifying and correcting errors in the dataset, such as dealing with missing or inconsistent data, removing duplicates, and handling outliers. Ensuring you train the machine learning mode on accurate and reliable data is essential. The model may learn from incorrect data without proper cleaning, leading to inaccurate predictions or classifications.

liver.duplicated()

If any duplicates are present it returns ture else it returns false.

liver=liver.drop_duplicates()

print(liver.shape)

This code removes the duplicated rows.

liver.isna().sum()

It checks all the missing values with respect to the coloumns.

sns.boxplot(data=liver,x='Albumin_and_Globulin_Ratio')

Here the Albumin and Globulin has 4 missing values so using boxplot method for calculating mean,mode,median.

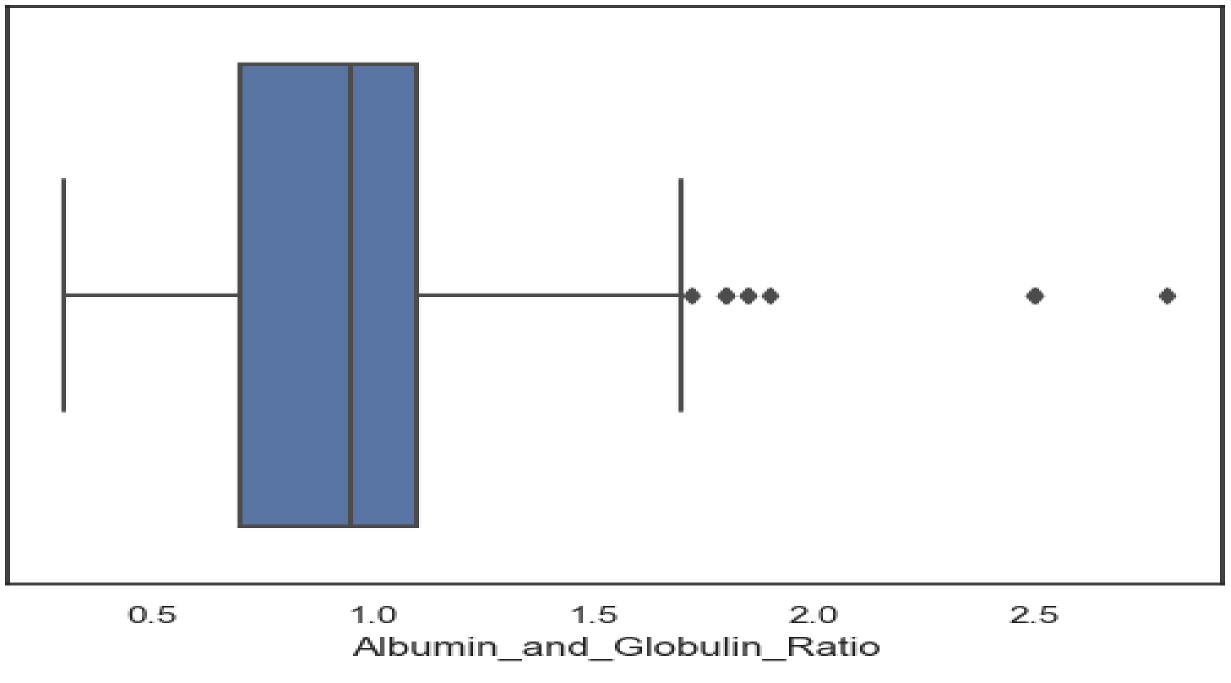


Fig 8.2 box plot

liver['Albumin_and_Globulin_Ratio']=liver['Albumin_and_Globulin_Ratio'].fillna(liver['Albumin_and_Globulin_Ratio'].median())

liver.isna().sum()

The above code fills the all missing values and check the column whether there are any missing values or not through isna().sum() method.

Data Exploration:

liver.hist(figsize=(15,15), xrot=-45, bins=10)

plt.show()

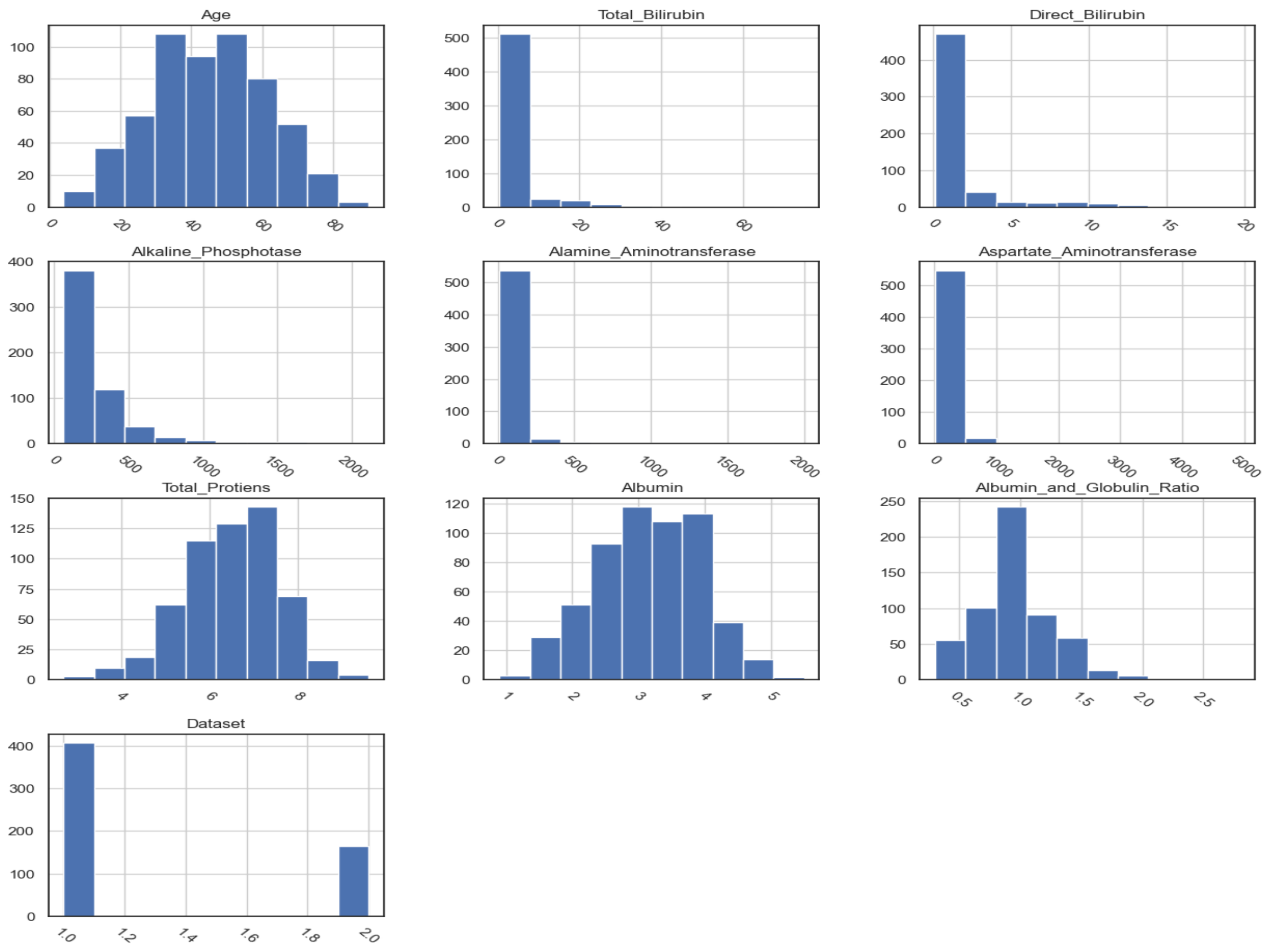


Fig 8.3 Histogram

liver.plot(kind="scatter", x="Total_Bilirubin", y="Direct_Bilirubin")

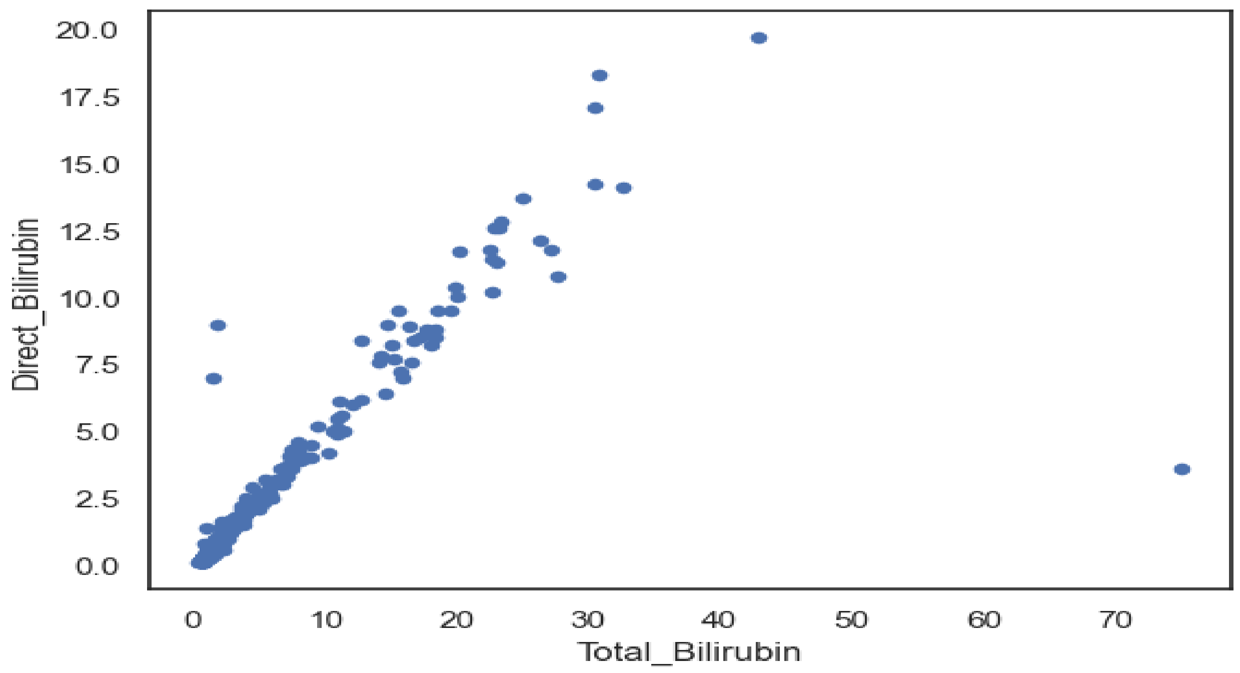


Fig 8.4: BarPlot

import seaborn as sns

sns.countplot(data = liver,x='Gender',label='count')

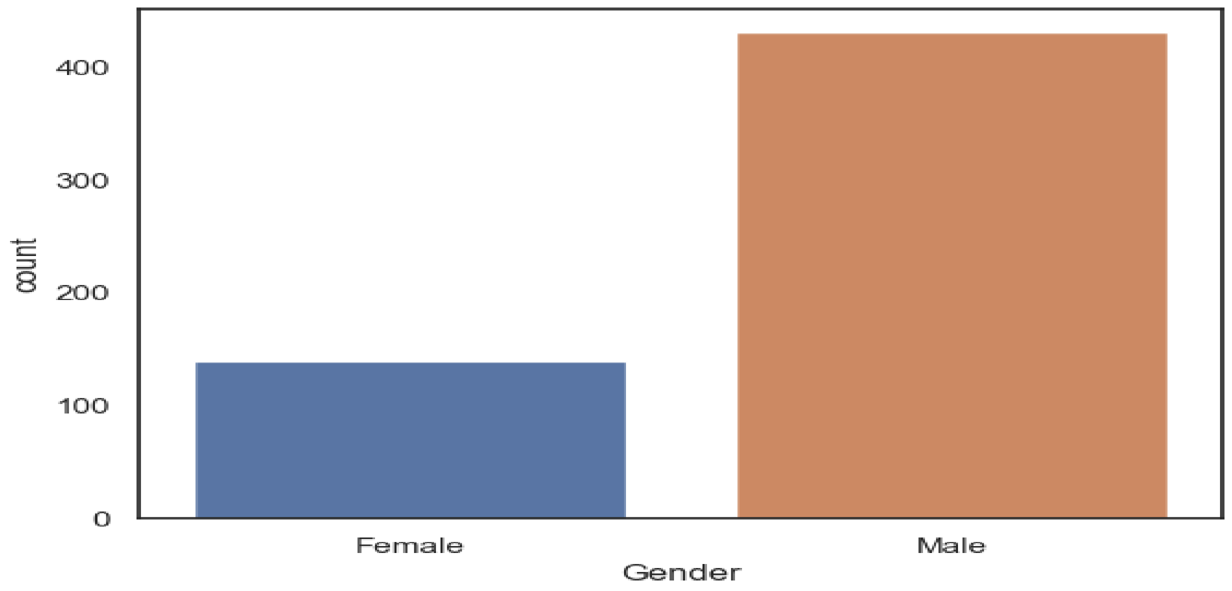


Fig 8.5: Gender Count

sns.jointplot(x="Age", y="Total_Protiens", data=liver, size=5)

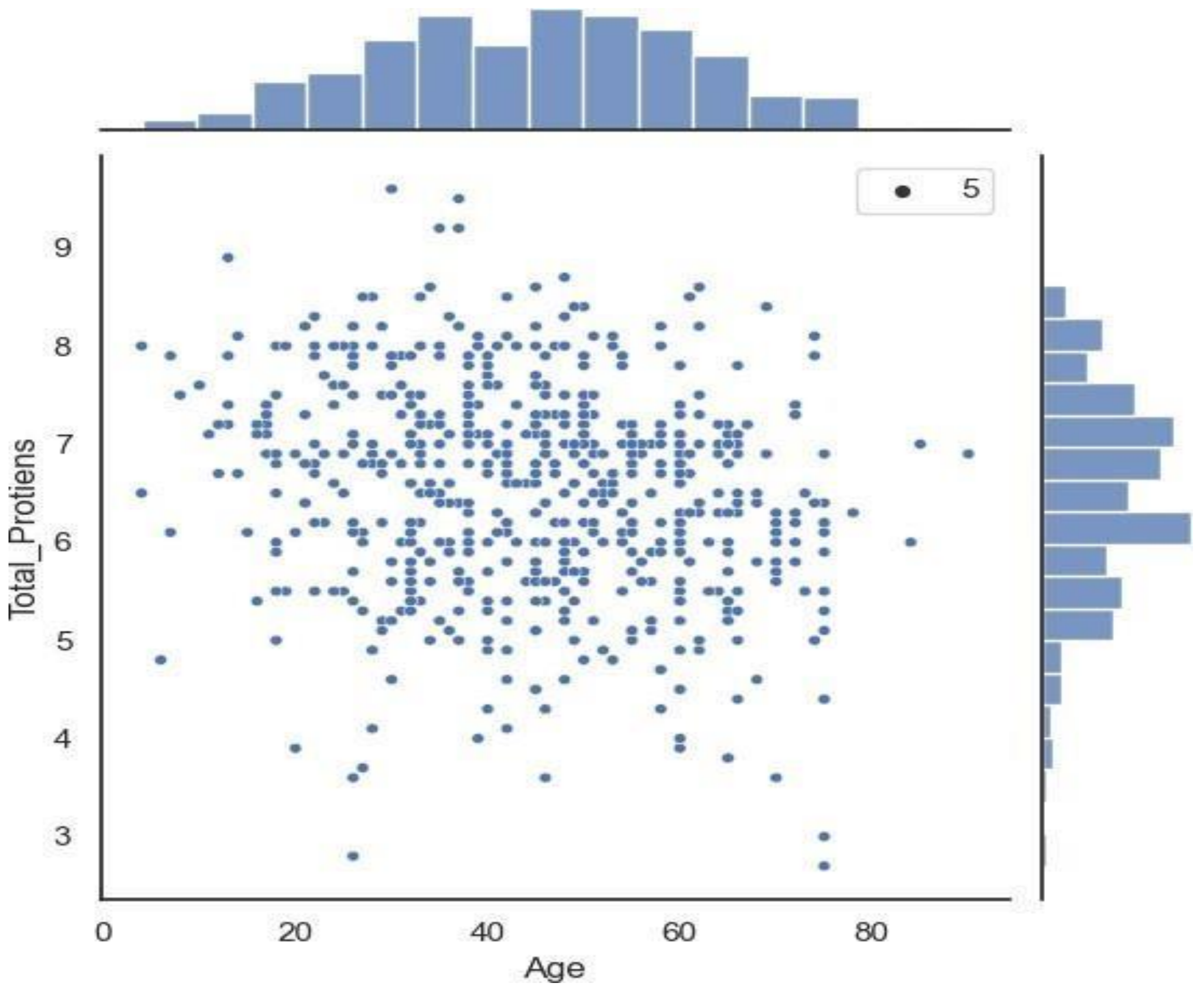


Fig 8.6: Jointplot

liver['Gender'].replace(to_replace=['Male','Female'], value=[0,1],inplace=True)
Here the gender are replaced with 0 and 1 Male to 0 and Female to 1.

Feature=liver[['Age','Gender','Total_Bilirubin','Direct_Bilirubin','Alkaline_Phosphotase','Alamine_Aminotransferase','Aspartate_Aminotransferas'Total_Protiens','Albumin','Albumin_and_Globulin_Ratio']]

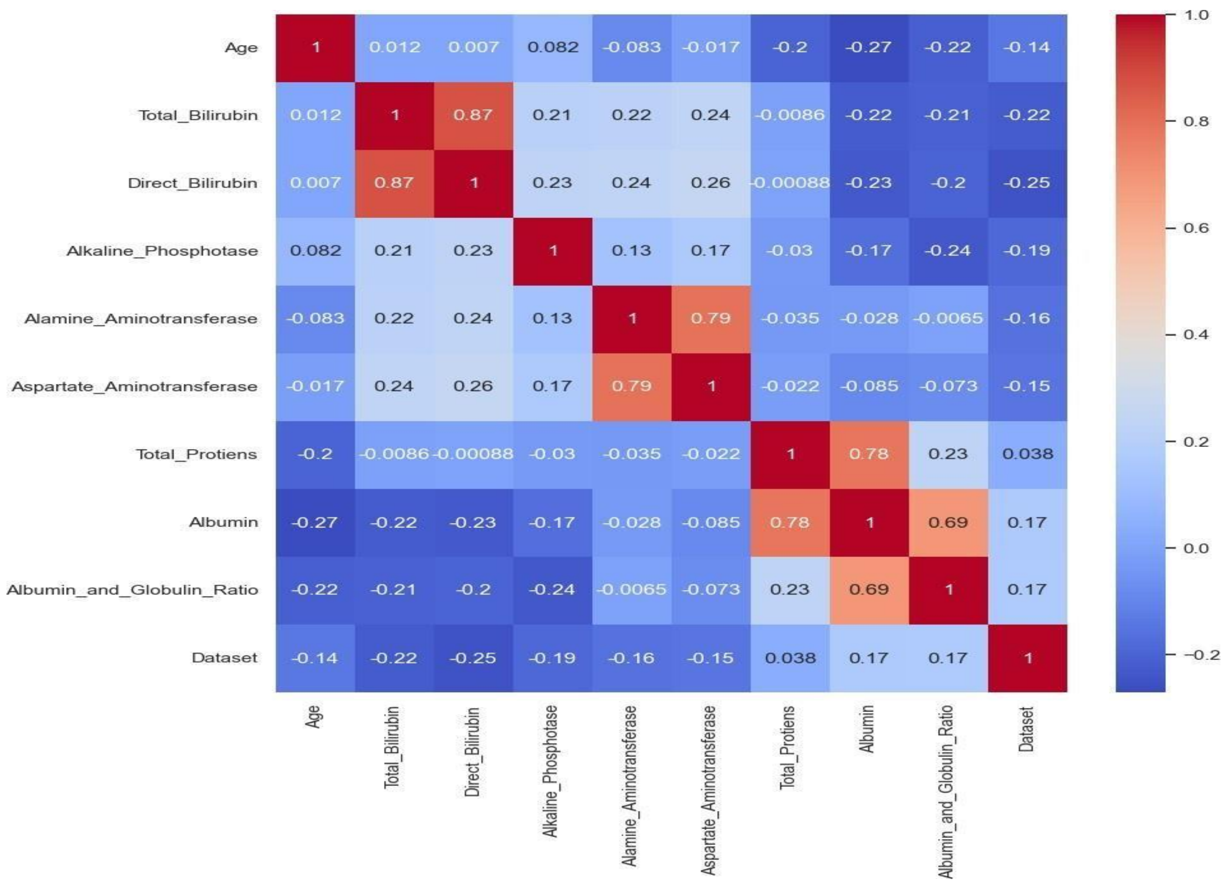


Fig 8.7: Correlation matrix

8.4 Model Building:

from sklearn.metrics import confusion_matrix

from sklearn.metrics import classification_report

from sklearn.model_selection import train_test_split

X = liver.iloc[:, :-1].values

Y = liver.iloc[:, -1].values

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

print(X.shape, Y.shape, X_train.shape, X_test.shape, Y_train.shape, Y_test.shape

This code performs train-test splitting on the liver dataset, separating features (X) and labels (Y), then prints the shapes of the original dataset and the train-test split datasets for verification

Logistic Regression:

Logistic regression is a binary classification algorithm. Estimates the probability of occurrence of category names. As a result, data points are marked (or truncated) with a fixed threshold; a threshold of 0.5 is usually used.

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(X_train, Y_train)
Y_pred = classifier.predict(X_test)
print(classification_report(Y_test, Y_pred))
print(confusion_matrix(Y_test, Y_pred))
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(Y_pred,Y_test))
```

K-Nearest Neighbor:

In machine learning, KNN is one of the most important time-based classification algorithms. However, KNN challenges the assumption that the situation is close to similar sample classes. It does not require training on equipment. KNN divides samples into classes with which K neighbors agree the most. Limiting K affects how the classification algorithm changes.

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=8)
classifier.fit(X_train, Y_train)
Y_pred = classifier.predict(X_test)
print(classification_report(Y_test, Y_pred))
print(confusion_matrix(Y_test, Y_pred))
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(Y_pred,Y_test))
```

Support Vector Machine:

It is a supervised learning algorithm for classification and regression. It is used only in classification problems. The goal of the support vector machine algorithm is to find a large plane that neatly divides the data points in N-dimensional space (N - number of features).

```
from sklearn.svm import SVC
classifier = SVC()
classifier.fit(X_train, Y_train)
Y_pred = classifier.predict(X_test)
print(classification_report(Y_test, Y_pred))
print(confusion_matrix(Y_test, Y_pred))
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(Y_pred,Y_test))
```

Random Forest Classifier:

Random forest is a type of supervised learning based on the concept of joint learning, which is the process of combining various classifiers to solve complex problems and improve performance standards. Variables were chosen randomly and the model was trained on bootstrap samples. The final result will be calculated by combining these trees.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X_train, Y_train)
Y_pred = classifier.predict(X_test)
print(classification_report(Y_test, Y_pred))
print(confusion_matrix(Y_test, Y_pred))
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(Y_pred,Y_test))
```

The performance of above algorithms is measured based on precision, recall, f1-score, accuracy.

8.5 Performance Evaluation Metrics:

Precision:

Precision is one of the hallmarks of a machine learning model; that is, the quality of the predictions made by the model. Precision is the number of true positives divided by the total number of positive predictions (i.e., the number of positives plus the number of negatives).

Recall:

Recall is a metric that measures how often a machine learning model correctly identifies positive instances from all the actual positive samples in the dataset. Recall is calculated by dividing the number of true positives by the number of positive instances.

Accuracy:

Accuracy is a measure of how often a machine learning model correctly predicts an outcome. You can calculate accuracy by dividing the number of correct predictions by the total number of predictions. It is nothing but how accurate the model is?

F1-Score:

The F1 score is a metric used to evaluate the performance of classification models, especially when dealing with heterogeneous classes. The F1 score is a compromise between precision and recall, combining these two measurements into a single value. It provides a balanced evaluation by taking into account negatives and false negatives.

Based on the above metrics Logistic Regression is more accurate when compared to other algorithms so this model is saved in Flask in order to predict the liver disease of the patient. Now include the model in the flask code and save that file with .py extension.

8.6 The flask code is:

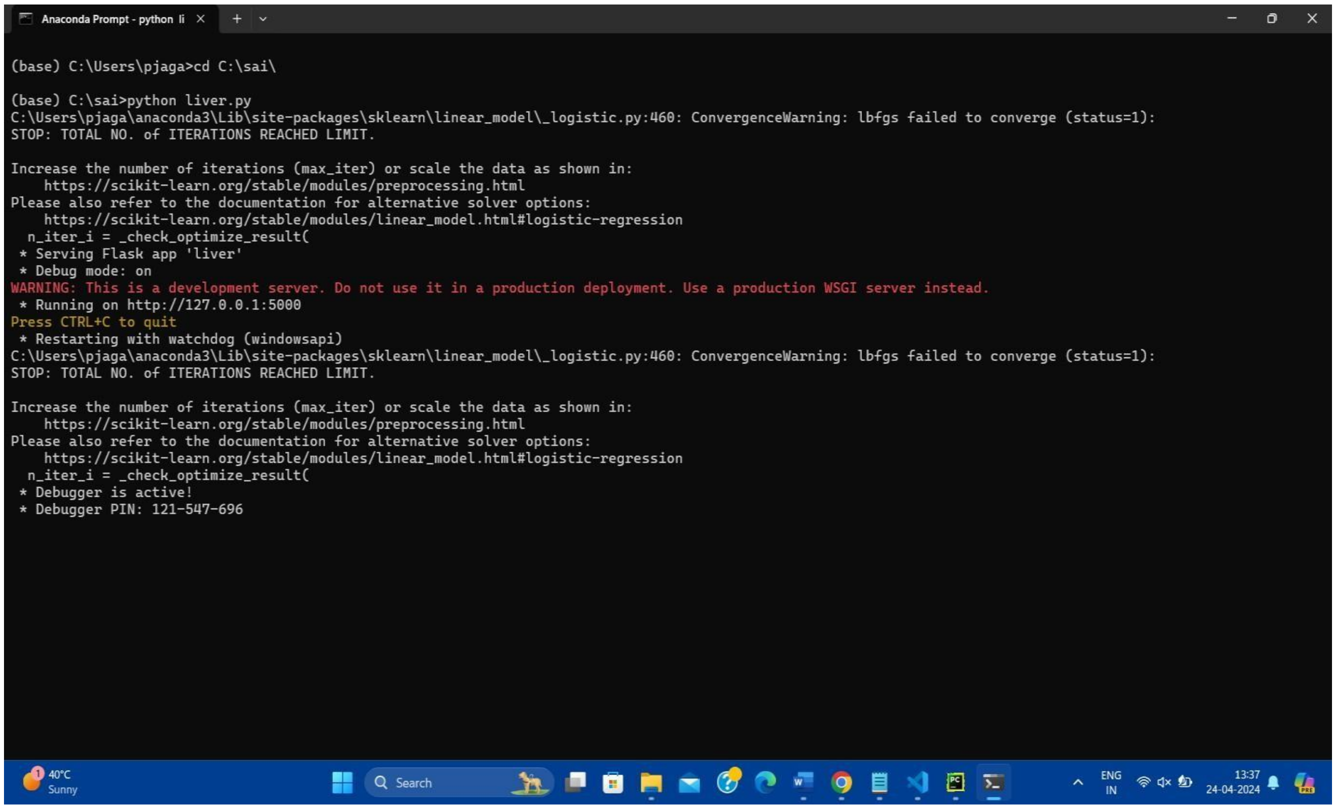
```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import pickle
from flask import Flask, render_template, request
app = Flask(_name _)
liver = pd.read_csv('C:\sai\indian_liver_patient.csv')
le = LabelEncoder()
liver['Gender'] = le.fit_transform(liver['Gender'])
imputer = SimpleImputer(strategy='mean')
liver = pd.DataFrame(imputer.fit_transform(liver), columns=liver.columns)
liver.dropna(subset=['Dataset'], inplace=True)
X = liver.drop(['Dataset', 'Direct_Bilirubin'], axis=1)
y = liver['Dataset'].replace({'Yes': 1, 'No': 0})
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
feature_names = X.columns.tolist()
model = LogisticRegression()
model.fit(X_train, y_train)
filename = 'sai.pkl'
pickle.dump(model, open(filename, 'wb'))
@app.route('/')
def home():
   return render_template('index.html')
@app.route('/predict', methods=['POST'])
```

```
def predict():
    model = pickle.load(open('sai.pkl', 'rb'))
    age = float(request.form['Age'])

    gender = int(request.form['Gender'])
    total_bilirubin = float(request.form['Total_Bilirubin'])
    alkaline_phosphotase = float(request.form['Alkaline_Phosphotase'])
    alamine_aminotransferase = float(request.form['Alamine_Aminotransferase'])
    aspartate_aminotransferase = float(request.form['Aspartate_Aminotransferase'])
    total_protiens = float(request.form['Total_Protiens'])
    albumin = float(request.form['Albumin'])
    albumin_globulin_ratio = float(request.form['Albumin_and_Globulin_Ratio'])
    input_data = pd.DataFrame({
        'Age': [age],
        'Gender': [gender],
        'Total_Bilirubin': [total_bilirubin],
        'Alkaline_Phosphotase': [alkaline_phosphotase],
        'Alamine_Aminotransferase': [alamine_aminotransferase],
        'Aspartate_Aminotransferase': [aspartate_aminotransferase],
        'Total_Protiens': [total_protiens],
        'Albumin': [albumin],
        'Albumin_and_Globulin_Ratio': [albumin_globulin_ratio]
    })
    prediction = model.predict(input_data)
    return render_template('result.html', prediction=prediction[0])

if __name__ == '_main_':
    app.run(debug=True)
```

9. IMPLEMENTATION PROCESS

9.1 Execution Process

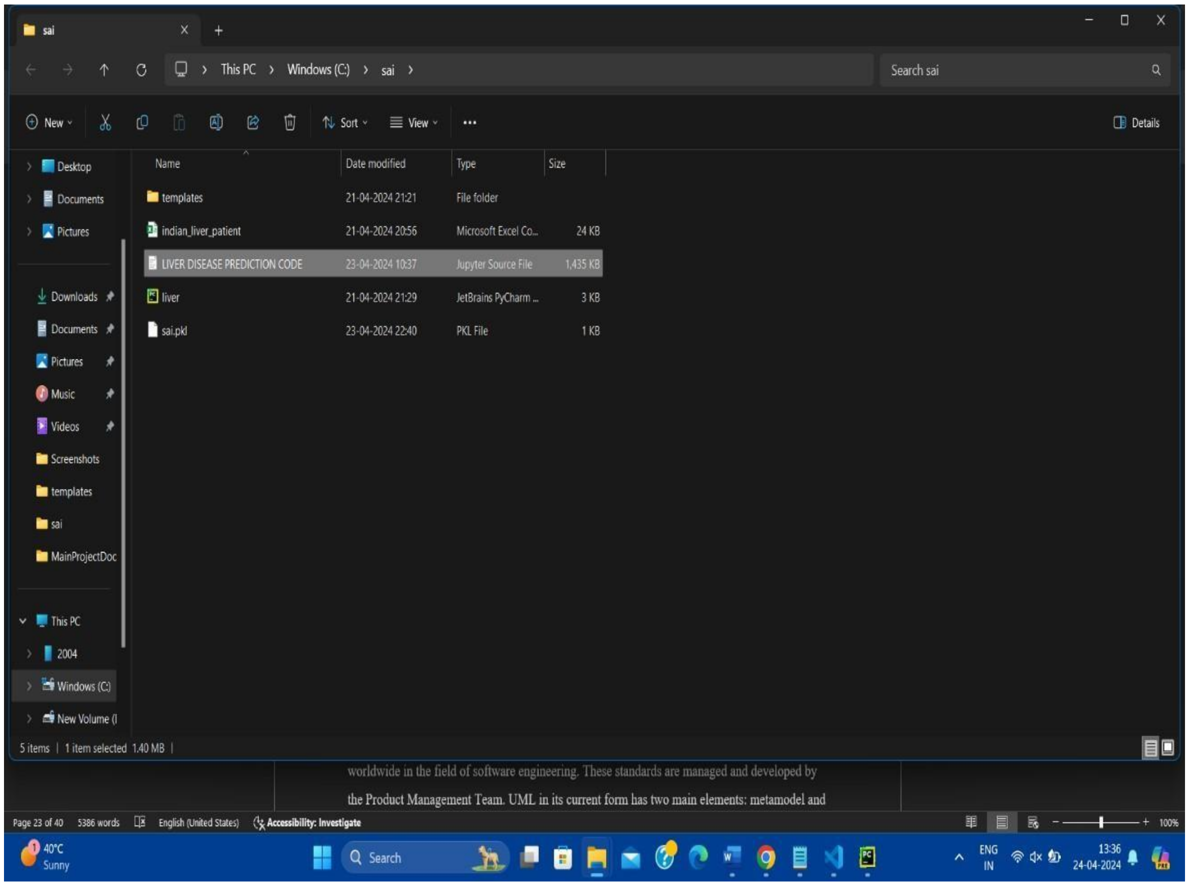
- Install the anaconda navigator and launch the jupyter notebook.
- In Jupyter notebook create a new file and select python 3 and then write the codes of machine learning models.
- Later after all the cells are executed properly create a folder in the desktop/downloads/ any other directory of our choice and place the file of machine learning model which was ran on jupyter notebook
- Now create a folder of templates and place the index and result pages html realated code in this templates floder.
- And also place the flask code.
- Now open the anaconda prompt and change the directory where all files are present.
- And now type python flask related file name so that it gives url http://127.0.0.1:5000



```
(base) C:\Users\pjaga>cd C:\sai\

(base) C:\sai>python liver.py
C:\Users\pjaga\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
 * Serving Flask app 'liver'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
C:\Users\pjaga\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
 * Debugger is active!
 * Debugger PIN: 121-547-696
```

Fig 9.1: Flask execution

- Once Flask code is run properly we get pickle file generated in our folder



Name	Date modified	Type	Size
templates	21-04-2024 21:21	File folder	
indian_liver_patient	21-04-2024 20:56	Microsoft Excel Co...	24 KB
LIVER DISEASE PREDICTION CODE	23-04-2024 10:37	Jupyter Source File	1,435 KB
liver	21-04-2024 21:29	JetBrains PyCharm ...	3 KB
sai.pkl	23-04-2024 22:40	PKL File	1 KB

worldwide in the field of software engineering. These standards are managed and developed by

the Product Management Team. UML in its current form has two main elements: metamodel and

Fig 9.2: Pickle File Generation

- Paste the url in browser and index form will display where user need to enter the values and the results are produced based on the entered values with respect to the presence of liver disease or not.

10. SYSTEM DESIGN

10.1 Introduction to UML (Unified Modeling Language)

Unified Modeling Language (UML) diagrams have played an important role in software engineering since its inception.UML was first published as a modeling framework for software development in the mid-1990s. Various types of modeling are produced through the collaboration, including the Booch method, OMT (Object Modeling Technology) and OOSE (Object-Oriented Software Engineering). Standard languages are widely used in the software industry. UML provides a standard for visualizing, designing, and documenting software, ensuring consistency and interoperability across different functions and organizations. Including developers, designers, architects, testers and project managers.

They facilitate better coordination and understanding of complex systems by providing a graphical representation of the system architecture, behavior, and requirements. Detailed methods and interactions. They help identify needs, identify patterns, teach behavior, and model relationships between physical objects. They provide valuable information for future development, maintenance, and additional efforts.

They help stakeholders see patterns and behaviors, identify design flaws or inconsistencies, and make informed decisions about design and implementation. The support provides developers with powerful tools to create, edit and analyze UML diagrams. These tools provide automatic drawing patterns, pattern analysis, marking and integration with control patterns to increase productivity and software development efficiency. They help develop quality, robust and maintainable software products by facilitating collaboration, analysis, design, and documentation throughout the software development lifecycle.

- **Class Diagram:**

Class diagrams are arguably the most used UML diagram type. It is the main building block.1 of any object-oriented solution. It shows the classes in a system, attributes and operations of each class and the relationship between each class. In most modeling tools a class has three parts, name at the top, attributes in the middle and operations or methods at the bottom.

- **Component Diagram:**

A class diagram in the Unified Modeling Language (UML) is a diagram used to visualize the structure of a system by defining classes, their properties, methods, relationships, and constraints.The class map provides a static view of the system, focusing on the structure of the system, not its character. and relationships based on design guidelines.

It usually includes class names, objects (data members), and methods (function members).It represents relationships and the fulfillment of paths and relationships. They help developers create robust, manageable, and scalable software systems by supporting design and team communication.

- **Deployment Diagram:**

The purpose of a deployment diagram in software engineering is to show the physical implementation of software components and their interactions with hardware components in the system architecture. They provide a visual representation of software structures deployed across hardware, including servers, computers, and devices. Facilitates communication between stakeholders by clearly explaining delivery and the relationship between software and hardware.They help identify dependencies and interactions between software components and hardware nodes to obtain better guidance and constraints.

- **Use Case Diagram:**

Used to visualize the interaction between users (participants) and the system and to show how the system works (use cases).They facilitate validation by allowing participants to review and provide feedback on performance and interactions. Make more adjustments. In general, the use of diagrams plays an important role in the analysis, design and communication of software engineering projects, helping to create successful and user-friendly systems.

- **Activity Diagram:**

It provides a clear picture of the sequence of activities, decision points, integration and integration in a system. These Diagrams are widely used in software engineering to describe the activities of software workflows,use cases, and operations. They act as communication tools between stakeholders, allowing developers, designers, and users to discuss and adjust behavior.

- **State Machine Diagram:**

It comprises of states, moves between states, and occasions or triggers that cause moves. Each state speaks to a condition or mode in which the framework exist, and moves delineate the development between states activated by particular occasions. They give a visual representation of the system's behavior, analyze, and plan state-dependent forms.

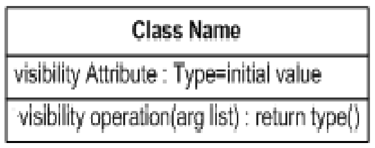
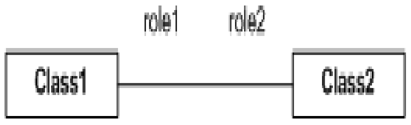
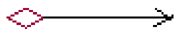
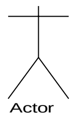
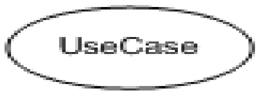
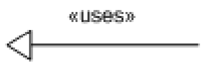
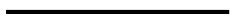
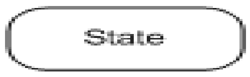
- **Sequence Diagram:**

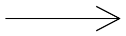
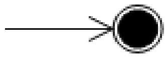
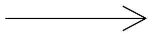
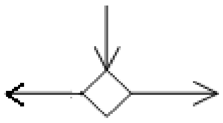
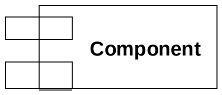
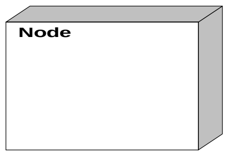
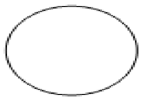
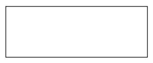
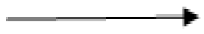
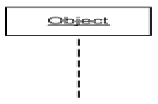
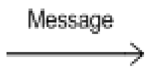
Used to visualize the intuitive between objects or components inside a framework over time. The arrangement of messages traded between objects. Objects are spoken to as vertical life savers, and messages are portrayed as flat bolts between life savers, demonstrating the arrange and heading of communication. They give a clear and brief visualization of message streams, and framework behavior.

- **Collaboration Diagram:**

Used to visualize components inside a framework. Objects are spoken to as hubs, and communication joins between objects are delineated as labeled bolts, demonstrating the stream of messages or intuitive and emphasize the communication and collaboration designs between objects.

10.2 List of UML Notations:

S.NO	SYMBOL NAME	SYMBOL	DESCRIPTION
1	Class	 Class Name visibility Attribute : Type=initial value visibility operation(arg list) : return type()	Set of objects having similar behaviour.
2	Association		Shows relationship between classes and contains multiplicity, roles.
3	Aggregation		One Class composed of other classes. shows part-of relationship
4	Actor		It is the role of user who interact with system.
5	Use Case		Functionality that system provides to the actors. Represents sequence of actions
6	Relation (Uses)		It is connection between classes, actors, use cases.
7	Communication		It is exchange of messages between objects.
8	State		It represents the condition of object or system.

9	Initial State		Starting point before any transitions occur.
10	Final State		It represents the final state after all transitions.
11	Control Flow		It represents the sequenced of actions.
12	Decision Box		It represents the splitting of control in activity diagram.
13	Component		Components represent the reusable part of the system.
14	Node		Reprsents the computational resource like processing unit.
15	Data Process/State		A computational function that manipulates data.
16	External Entity		It occurs outside the system.
17	Transition		It represents the change of state.
18	Object Lifeline		Object lifelines represents period which it is present in the system.
19	Message		It represents the information to be exchanged.

10.3 UML Diagrams

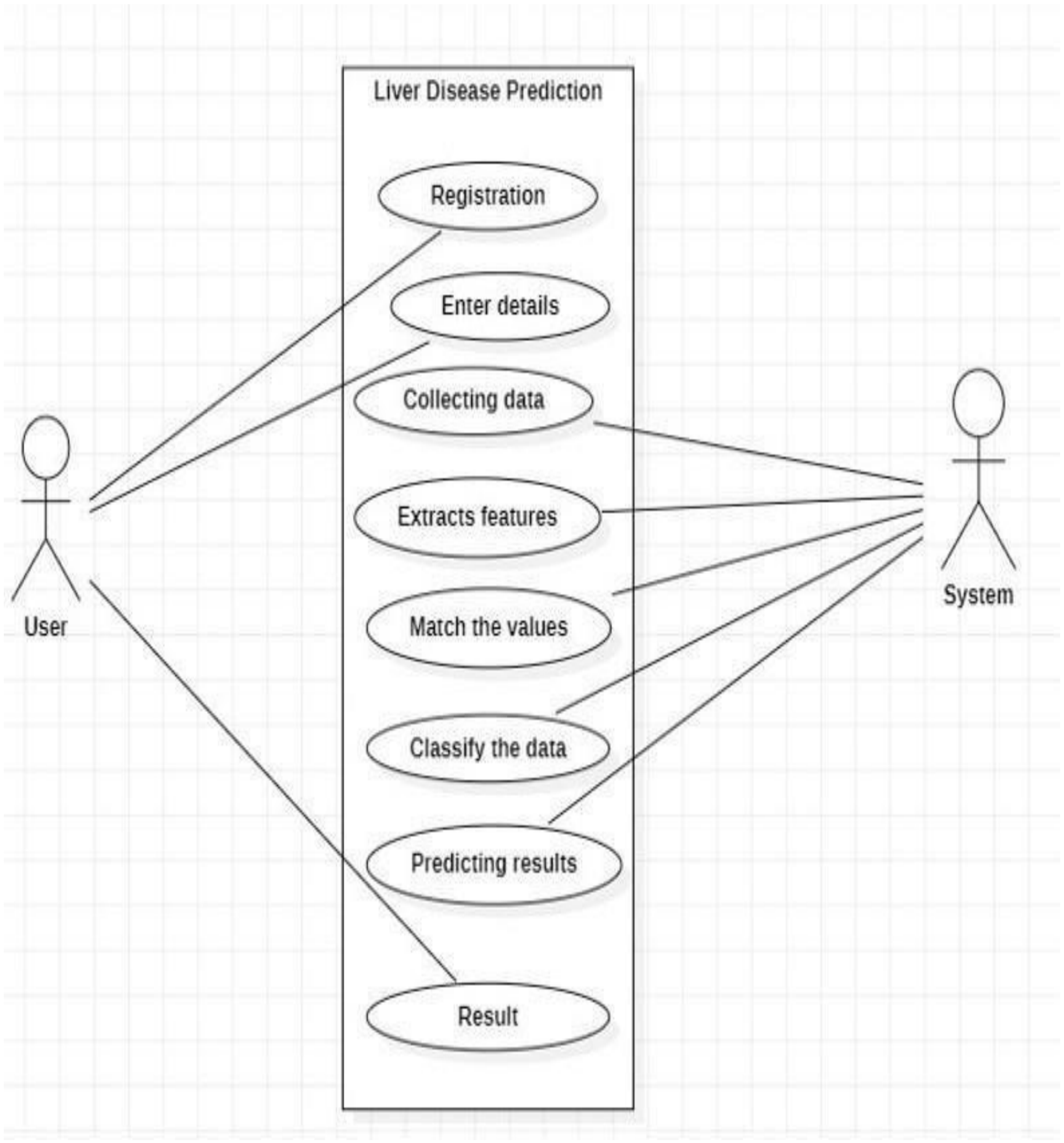


Fig 10.1: Use case Diagram

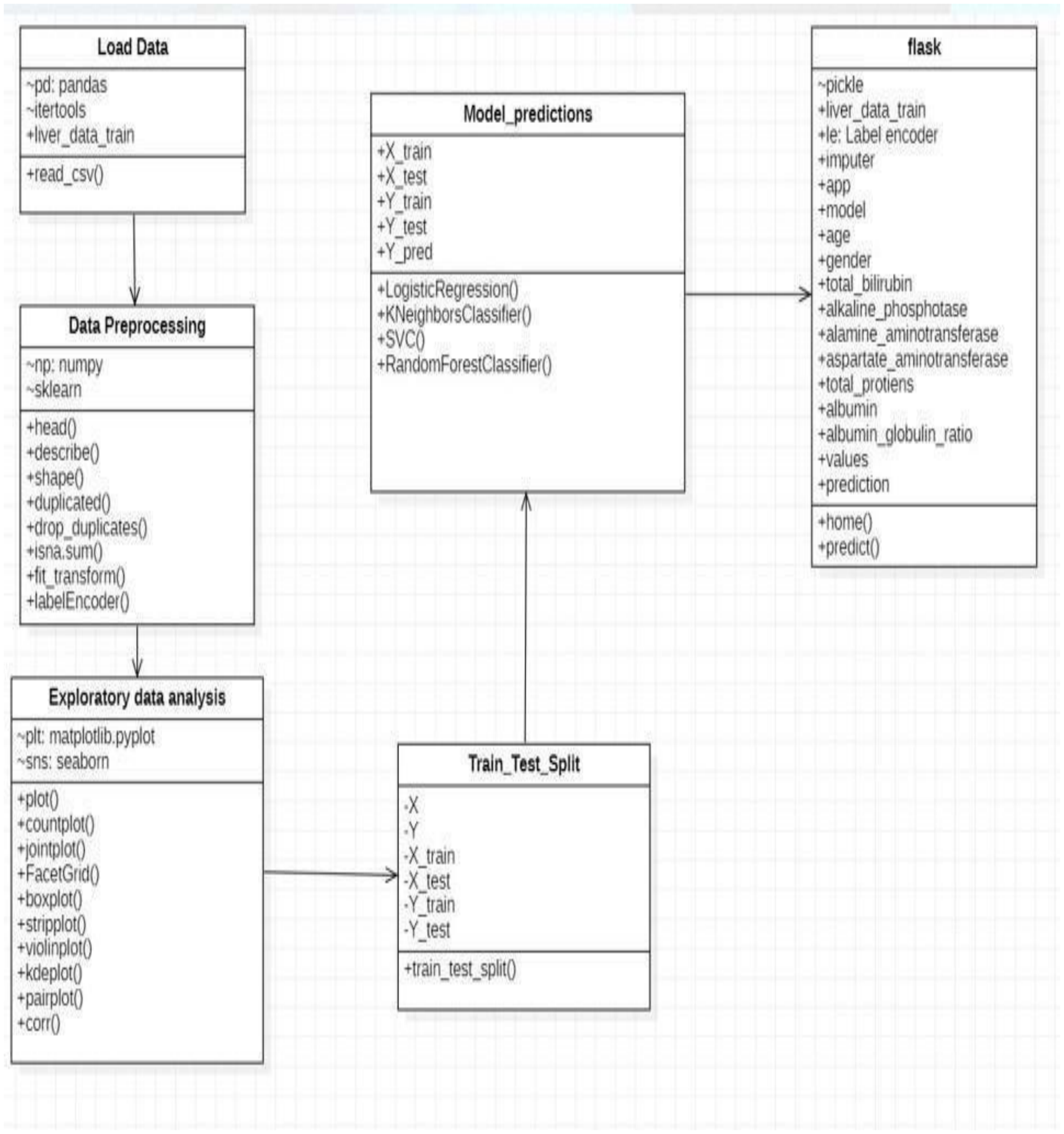


Fig 10.2: Class Diagram

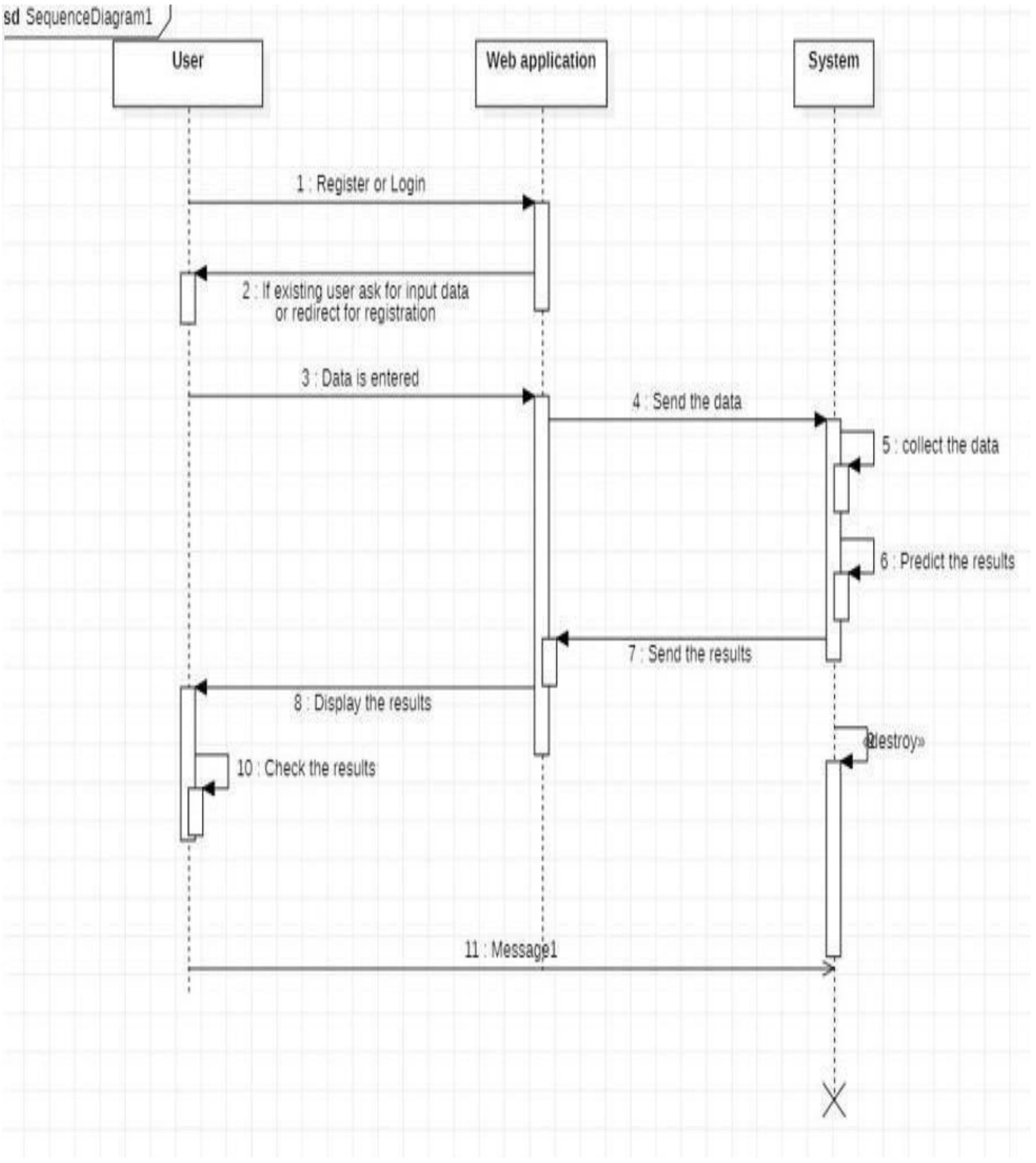


Fig 10.3: Sequence Diagram

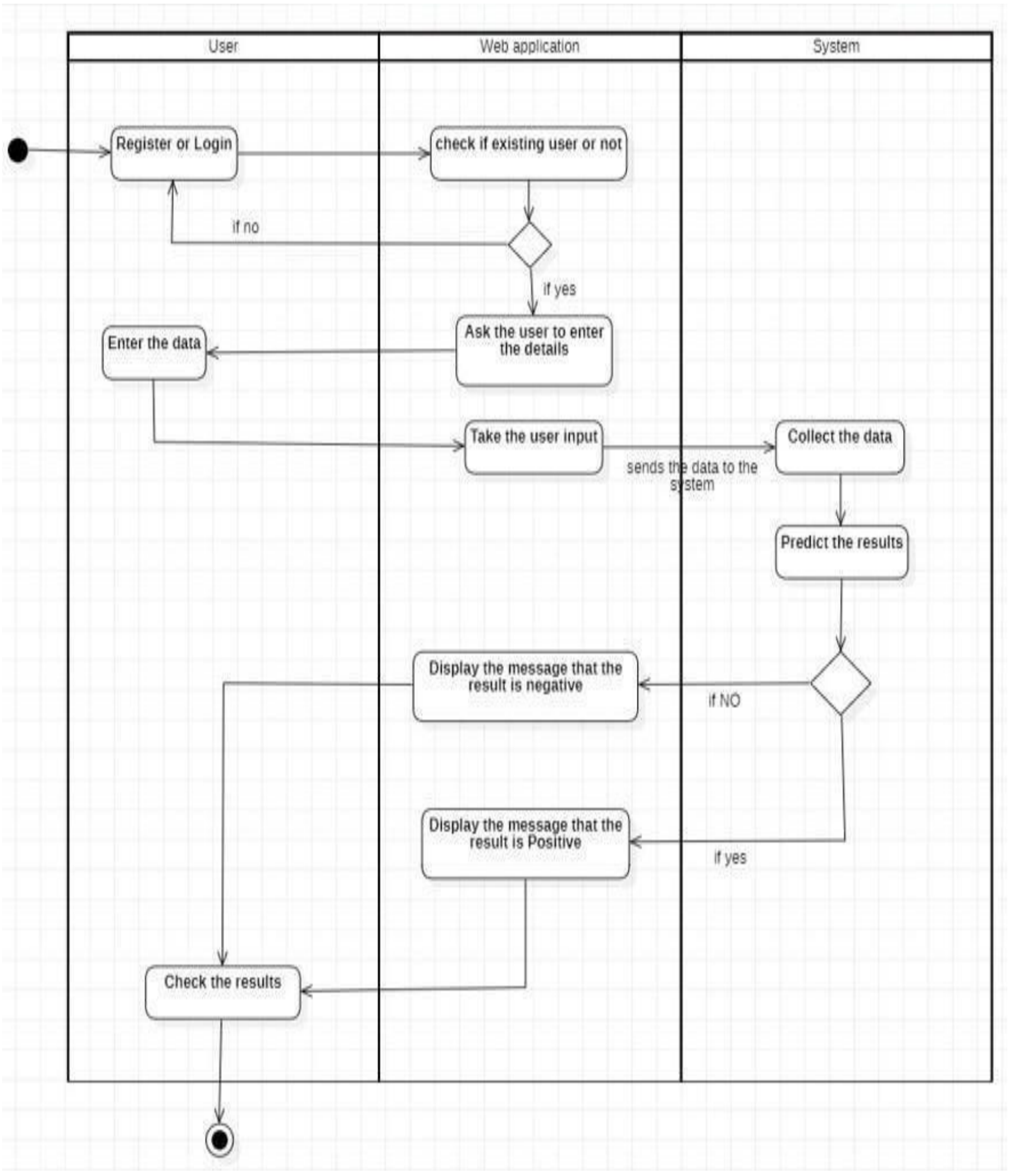

User
Web application
System
Register or Login
check if existing user or not
if no
if yes
Ask the user to enter the details
Enter the data
Take the user input
sends the data to the system
Collect the data
Predict the results
Display the message that the result is negative
if NO
Display the message that the result is Positive
if yes
Check the results

Fig 10.4: Activity Diagram

11. SYSTEM TESTING

11.1 Testing:

Testing, in brief, is the handle of assessing a computer program framework or application to guarantee that it meets indicated necessities and capacities accurately. It includes executing computer program components, modules, or the whole framework with the expectation of distinguishing surrenders, mistakes, or deviations from anticipated behavior. Testing is conducted over diverse stages of the program advancement lifecycle, counting unit testing, integration testing, framework testing, and acknowledgment testing, each centering on particular angles of the software's usefulness and quality.

The essential objectives of testing are to approve the rightness, unwavering quality, execution, security, and convenience of the computer program, as well as to guarantee that it meets client desires and commerce necessities. Testing is an basic and iterative action in program building, making a difference to recognize and relieve dangers, make strides program quality, and provide vigorous and dependable computer program items to end-users.

11.2 Testing Objectives:

Identify defects, errors, or bugs in the software to ensure they are resolved before deployment. This helps improve the quality and reliability of the software. Make sure it meets user needs and expectations. Reduce the impact of failure on users and stakeholders. Performance, scalability, and the ability of the software to respond to different situations such as changes, user connections, and restrictions.

It is user-friendly and meets the needs of its target users. Ensure the testing process is efficient and effective by reducing the time, effort and resources required while maximizing value for the organization. Performance, security and user satisfaction with the software ultimately lead to the success of software development and achievement of business goals.

11.3 Types of Testing:

Unit Testing:

Unit testing is regularly coordinates with the code improvement stage in the program lifecycle, guaranteeing that person units of code work as aiming. In any case, there are occasions where coding and unit testing are carried out as partitioned stages. In our approach, we embrace a combined code and unit test stage, permitting for quick input on code changes and guaranteeing early discovery of surrenders.

Furthermore, our test methodology includes a manual field testing approach, where the computer program is tried in real-world scenarios to approve its execution and convenience. Besides, utilitarian tests will be fastidiously reported and executed to confirm that the program meets indicated prerequisites and carries on as anticipated.

Integration Testing:

Software integration testing involves the integration of two or more software products or applications into a single platform for the purpose of detecting failures caused by interference. The main purpose of integration testing is to ensure that these devices or applications interact with each other without errors. This process ensures that the integration works as expected and meets the requirements.

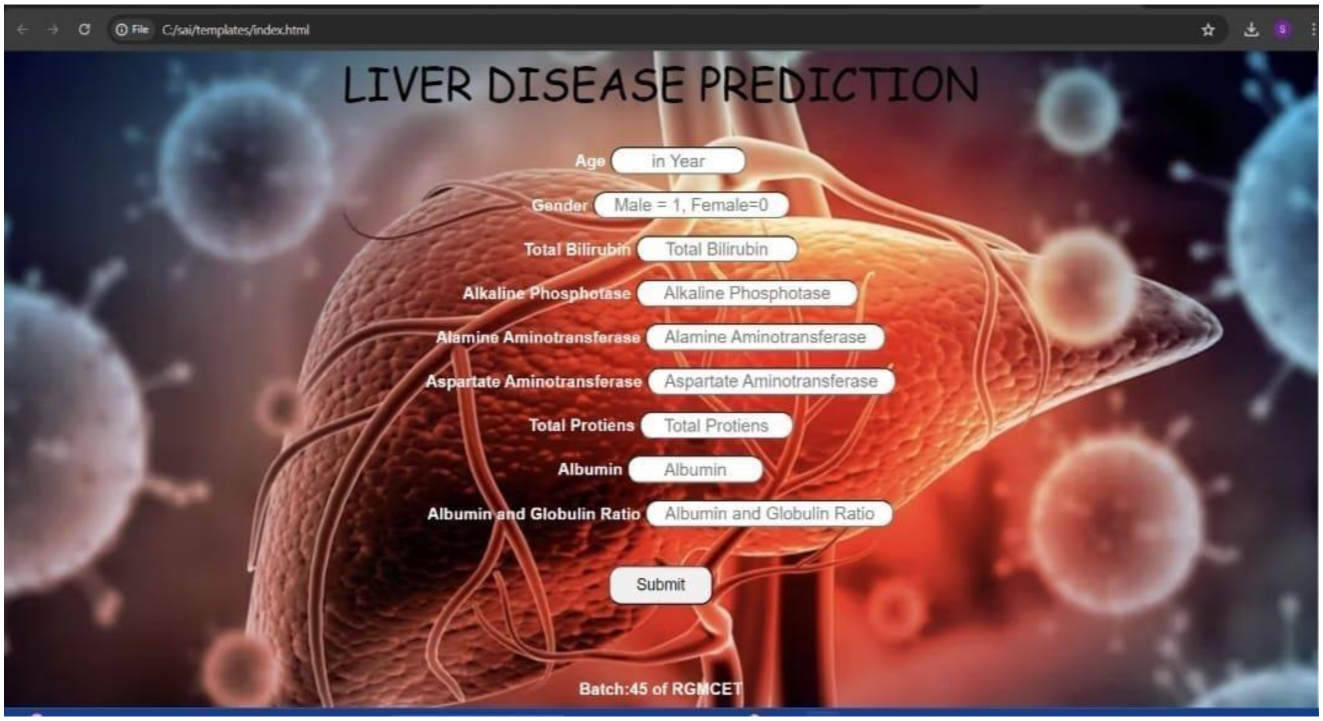
During integration testing, completing all previous tests without encountering any errors means that the components or applications are integrated correctly and working well. This result provides confidence in the reliability and stability paving the way for further testing and eventual deployment.

Acceptance Testing:

Acceptance testing, especially user authentication (UAT), is an important phase in the lifecycle of any project and requires significant involvement from end users. Its main purpose is to determine whether the system meets the requirements, essentially verifying that it meets the needs and requirements of its users. Serves as the final validation process before deployment, allowing end users to interact with the software in a real-world environment and provide feedback on the software's usability, operation, and overall quality.

12. RESULTS

Web Page: This is the interface provided for the user



C:/sai/templates/index.html

LIVER DISEASE PREDICTION

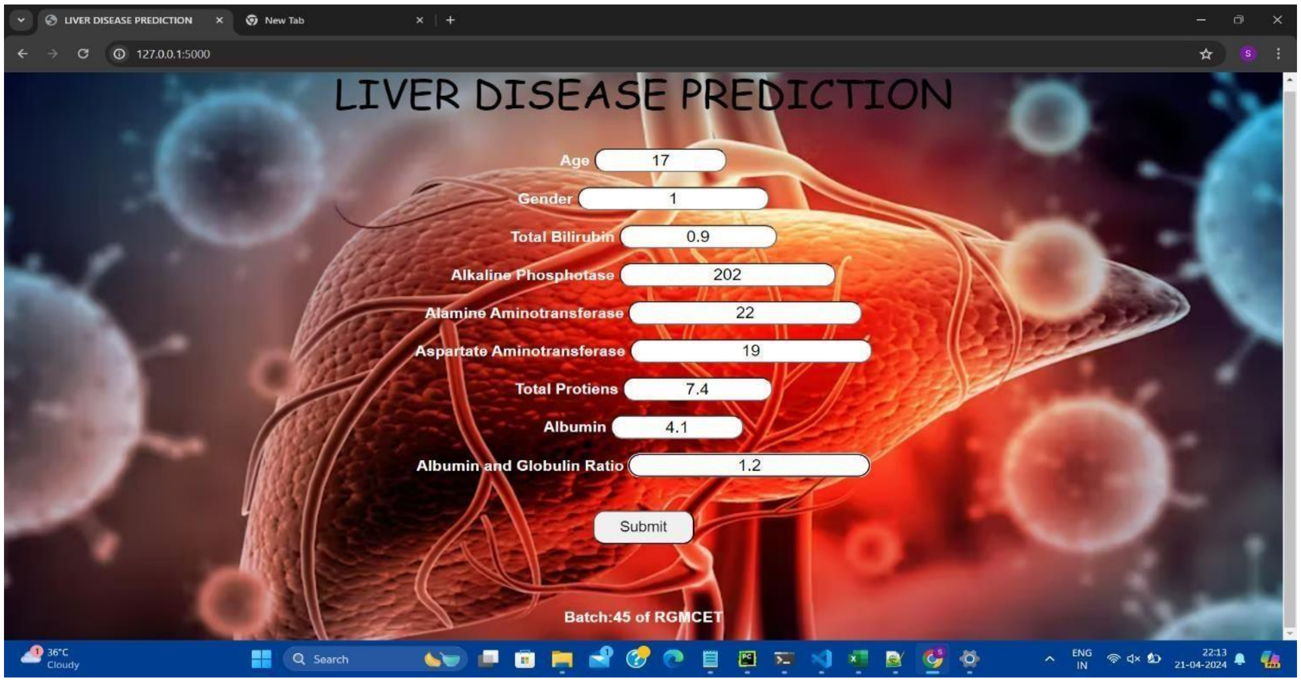
Age: in Year
Gender: Male = 1, Female=0
Total Bilirubin: Total Bilirubin
Alkaline Phosphotase: Alkaline Phosphotase
Alamine Aminotransferase: Alamine Aminotransferase
Aspartate Aminotransferase: Aspartate Aminotransferase
Total Protiens: Total Protiens
Albumin: Albumin
Albumin and Globulin Ratio: Albumin and Globulin Ratio

Submit

Batch:45 of RGMCET

Fig 12.1: Web Page

User entering the values:



LIVER DISEASE PREDICTION | New Tab

127.0.0.1:5000

LIVER DISEASE PREDICTION

Age: 17
Gender: 1
Total Bilirubin: 0.9
Alkaline Phosphotase: 202
Alamine Aminotransferase: 22
Aspartate Aminotransferase: 19
Total Protiens: 7.4
Albumin: 4.1
Albumin and Globulin Ratio: 1.2

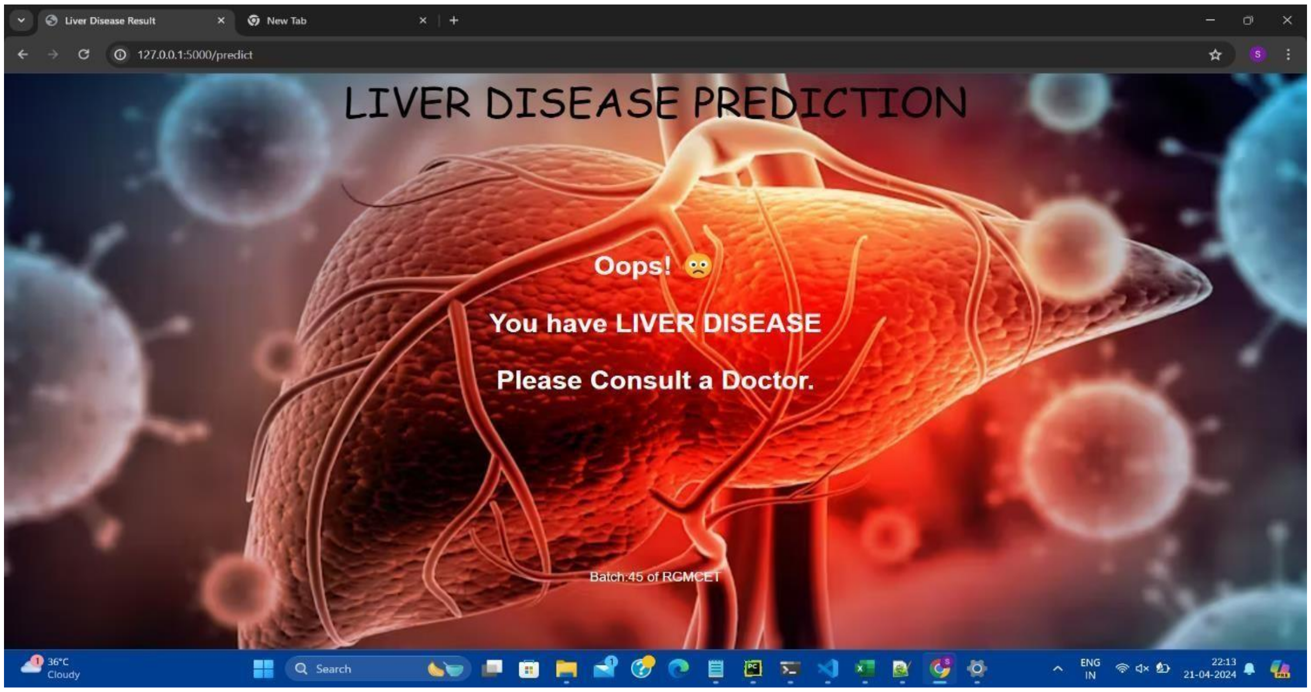
Submit

Batch:45 of RGMCET

36°C Cloudy | Search | ENG IN | 22:13 21-04-2024

Fig 12.2: User entering values

Display page: The patients gets the following message if they are having liver disease.



LIVER DISEASE PREDICTION

Oops! 🙁

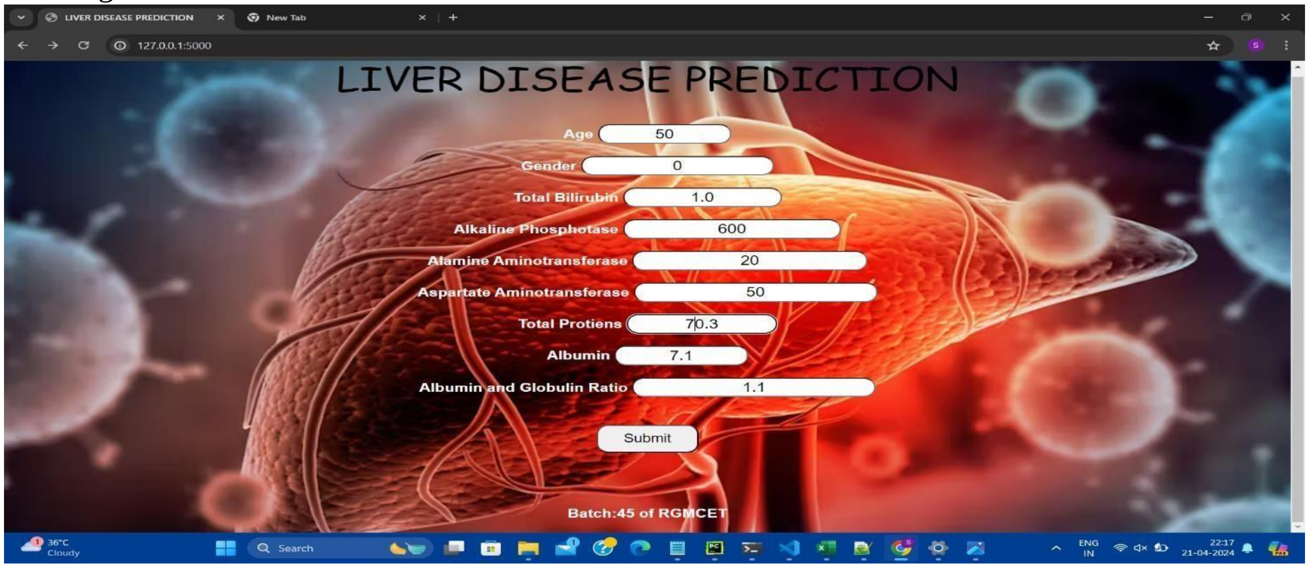
You have LIVER DISEASE

Please Consult a Doctor.

Batch:45 of RGMCET

Fig 12.3: Liver disease patient outcome

Entering values:



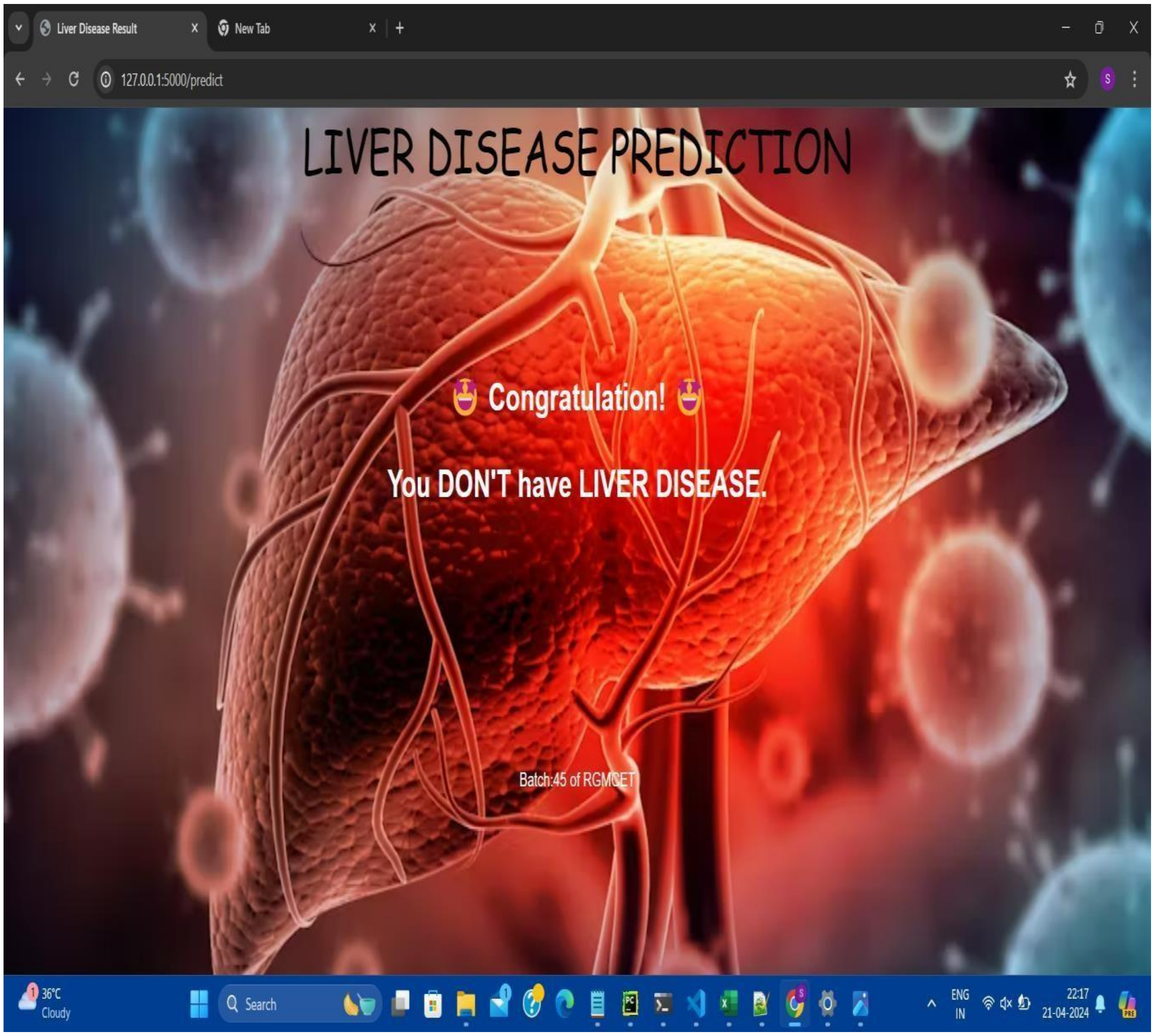
LIVER DISEASE PREDICTION

Age	50
Gender	0
Total Bilirubin	1.0
Alkaline Phosphotase	600
Alamine Aminotransferase	20
Aspartate Aminotransferase	50
Total Protiens	70.3
Albumin	7.1
Albumin and Globulin Ratio	1.1

Batch:45 of RGMCET

Fig 12.4: User entering values

Display page: The patients get the following message if they don’t have liver disease.



LIVER DISEASE PREDICTION

🤩 Congratulation! 🤩

You DON'T have LIVER DISEASE.

Batch:45 of RGMCET

Fig 12,5:Patient with no liver disease outcome

13. CONCLUSION

Logistic regression and distribution for prediction of liver disease using Flask are an important part of medical technology. After careful testing and validation, logistic regression emerged as the algorithm of choice, consistently and accurately predicting liver disease. This trust instills confidence in doctors and end users, ensuring that important diagnostic and treatment decisions can be made with certainty Accessibility and Usability.

Flask's lightweight, modular architecture provides a seamless user experience, allowing people to easily access information and get timely advice regarding liver diseases. This approach is important for the consumer market to promote good health management because it allows people to monitor their own health and find timely medical help when necessary.

Successful deployments have demonstrated the potential of machine learning to transform traditional healthcare. Using data from insights, clinicians can use predictive analytics to identify high-risk individuals, create personalized treatment plans, and ultimately improve patient outcomes. The shift to precision medicine holds great promise for the future of healthcare and offers opportunities for more efficient, effective, patient-centered care. How the system seamlessly integrates cutting-edge technologies like machine learning and web-based delivery frameworks to solve healthcare problems.

By combining rigorous research with innovative solutions, we not only advance the frontiers of medical science, but also enable more people to live healthy, fulfilling lives. The possibilities for revolutionizing healthcare are endless as we continue to explore the vast potential of artificial intelligence and data-driven insights. It has been shown to be useful in determining whether liver disease is present.

Through rigorous testing and evaluation, logistic regression consistently provides the most accurate predictions of all algorithms considered. Leveraging this information, the system was implemented using Flask, providing users with a simple tool to measure their mental health. By combining advanced machine learning techniques with powerful deployment capabilities, the system demonstrates the ability to impact health by detecting and preventing painful errors early.

BIBILIOGRAPHY

References for the Project Development are taken from the following.

- Ketan Gupta, Nasmin Jiwani, Neda Afreen & Divyarani D "Liver Disease Prediction using Machine learning Classification Techniques" in 11th IEEE International Conference on Communication Systems and Network Technologies.
- Nasmin Jiwani, Ketan Gupta, Neda Afreen "A Convolutional Neural Network Approach for Diabetic Retinopathy Classification" in 2022 IEEE 11th International Conference on Communication Systems and Network Technologies (CSNT).
- Thirunavukkarasu K, Ajay S. Singh, Md Irfan, Abhishek Chowdhury "Prediction of Liver Disease using Classification Algorithms" in 2018 4th International Conference on Computing Communication and Automation (ICCCA).
- "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurelien Geron

(ESTD-1995)

Rajeev Gandhi Memorial College of Engineering and Technology (RGM), Andhr Pradesh

Certificate of Plagiarism Check for Thesis

Author Name	GANGADASARI SAI LAKSHMI (20091A05D2)
Course of Study	**IV B.Tech II Sem. (R20) – May 2024**
Name of Guide	Type here...
Department	CSE
Acceptable Maximum Limit	30%
Submitted By	principal.9@jntua.ac.in
Paper Title	Anticipating hepatic disorder through machine learning classification techniques
Similarity	19%
Paper ID	1696841
Submission Date	2024-04-25 15:11:21

Note: Less than 30% Similarity index – Permitted to Submit the Thesis

[illegible] 25/4/24

Controller of Examinations

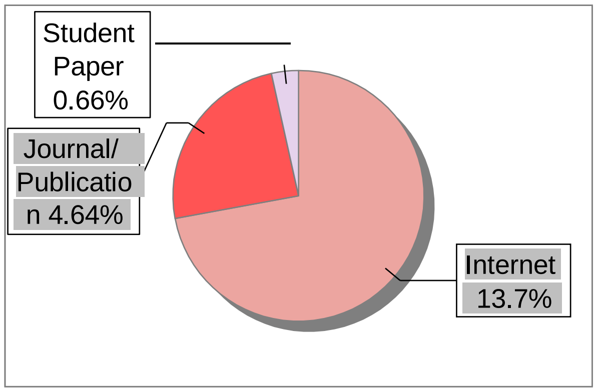
The Report is Generated by DrillBit Plagiarism Detection Software

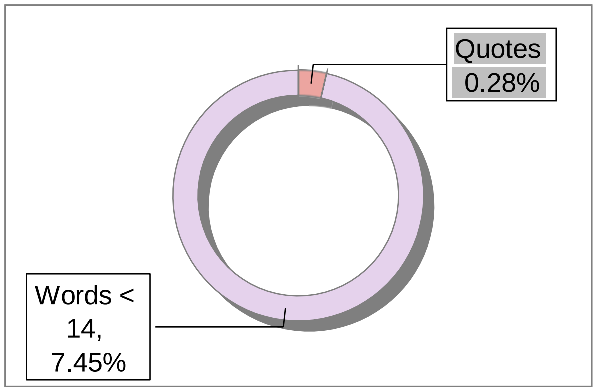
Submission Information

Author Name	GANGADASARI SAI LAKSHMI (20091A05D2)
Title	Anticipating hepatic disorder through machine learning classification techniques
Paper/Submission ID	1696841
Submitted by	principal.9@jntua.ac.in
Submission Date	2024-04-25 15:11:21
Total Pages	46
Document type	Project Work

Result Information

Similarity **19 %**





Exclude Information

Quotes	Not Excluded
References/Bibliography	Not Excluded
Sources: Less than 14 Words %	Not Excluded
Excluded Source	**0 %**
Excluded Phrases	Not Excluded

Database Selection

Language	English
Student Papers	Yes
Journals & publishers	Yes
Internet or Web	Yes
Institution Repository	Yes

A Unique QR Code use to View/Download/Share Pdf File

DrillBit

DrillBit Similarity Report

19	**80**	**B**	A-Satisfactory (0-10%) B-Upgrade (11-40%) C-Poor (41-60%) D-Unacceptable (61-100%)
SIMILARITY %	MATCHED SOURCES	GRADE	

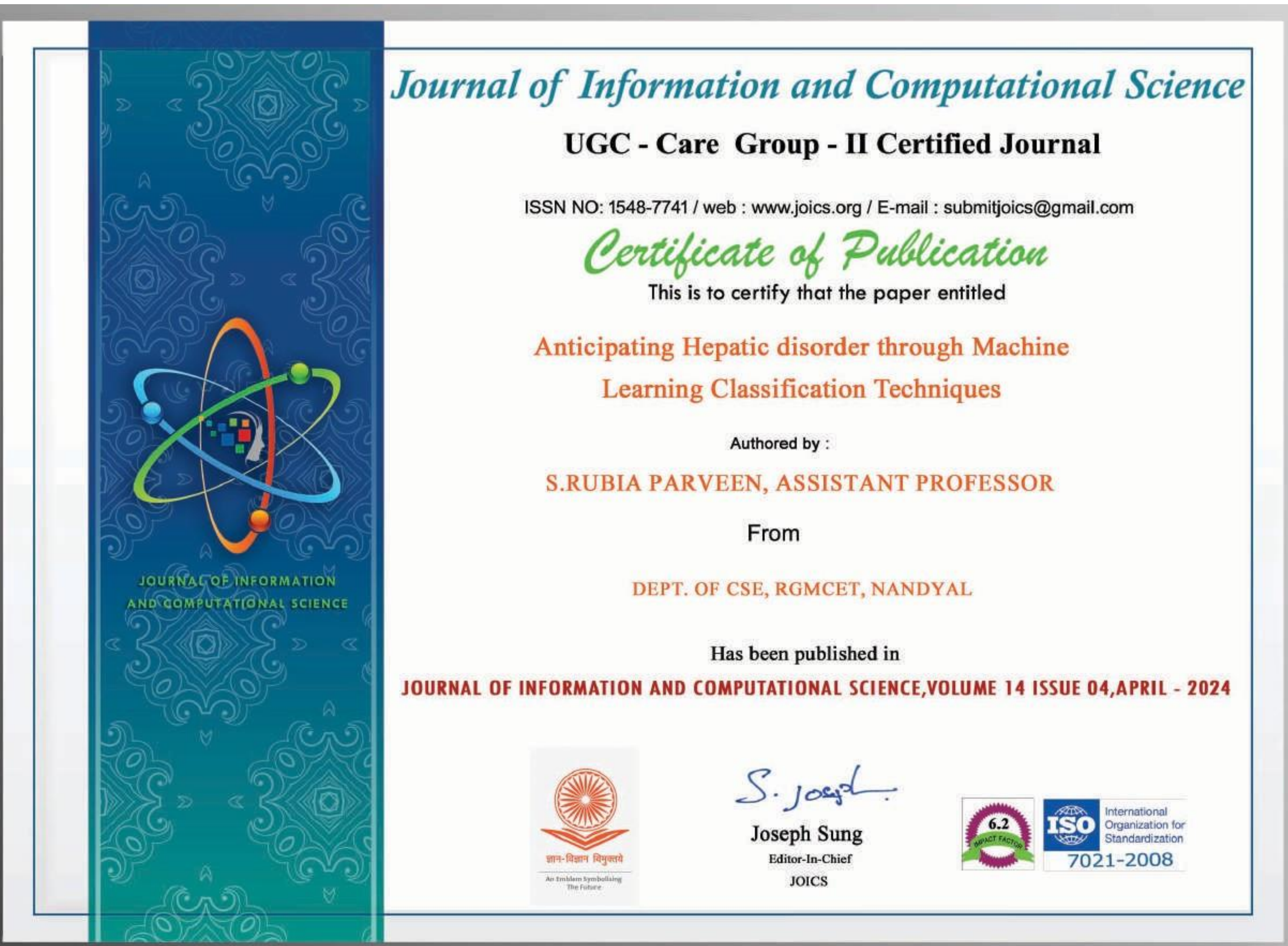
LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	moam.info	1	Internet Data
2	encord.com	1	Internet Data
3	moam.info	1	Internet Data
4	moam.info	1	Internet Data
5	www.adityatekkali.edu.in	1	Publication
6	qdoc.tips	1	Internet Data
7	moam.info	1	Internet Data
8	tc.copernicus.org	1	Internet Data
9	ijeat.org	<1	Publication
10	www.ncbi.nlm.nih.gov	<1	Internet Data
11	moam.info	<1	Internet Data
12	amedleyofpotpourri.blogspot.com	<1	Internet Data
13	sist.sathyabama.ac.in	<1	Publication
14	Submitted to Devi Ahilya Vishwavidyalaya on 2023-10-10 13-41	<1	Student Paper

15	ijrpr.com	<1	Publication
16	Submitted to Visvesvaraya Technological University, Belagavi	<1	Student Paper
17	60plus.org	<1	Internet Data
18	springeropen.com	<1	Internet Data
19	mdpi.com	<1	Internet Data
20	www.frontiersin.org	<1	Internet Data
21	www.scribd.com	<1	Internet Data
22	www.homeworkforyou.com	<1	Publication
23	apartamentoslaharinera.com	<1	Internet Data
24	www.rbrhs.org	<1	Internet Data
25	moam.info	<1	Internet Data
26	moam.info	<1	Internet Data
27	tripurauniv.ac.in	<1	Publication
28	mealplans.naturallysavvymom.com	<1	Internet Data
29	Energy Management of Electric Vehicle integrated Home in a Time-Of-Da, by Rana, Rubi Prakash- 2018	<1	Publication
30	Optimal consumption and allocation in variable annuities with Guaranteed Minimum by Ji-2012	<1	Publication
31	Predicting pulsar stars using a random tree boosting voting classifier (RTB-VC) by Rustam-2020	<1	Publication
32	spotlightenglish.com	<1	Internet Data

33	Thesis Submitted to Shodhganga Repository	<1	Publication
34	www.civilsdaily.com	<1	Internet Data
35	www.imageapi.com	<1	Internet Data
36	slask.joannici.org.pl	<1	Internet Data
37	www.conscientiabeam.com	<1	Publication
38	bg.copernicus.org	<1	Publication
39	cpijr.com	<1	Publication
40	gcelt.gov.in	<1	Internet Data
41	ias.ac.in	<1	Internet Data
42	teslontario.org	<1	Internet Data
43	www.ncbi.nlm.nih.gov	<1	Internet Data
44	www.youngwonks.com	<1	Internet Data
45	aicontentfy.com	<1	Internet Data
46	dickinson.edu	<1	Internet Data
47	jmi.ac.in	<1	Publication
48	journalijcmes.com	<1	Publication
49	www.intechopen.com	<1	Internet Data
50	www.zimyo.com	<1	Internet Data
51	IEEE 2017 IEEE International Symposium on Circuits and Systems (ISC, by Wess, Matthias Man- 2017	<1	Publication

52	mdpi.com	<1	Internet Data
53	raw.githubusercontent.com	<1	Internet Data
54	User preference for an IoT healthcare application for lifestyle diseas by Kim-2017	<1	Publication
55	1library.co	<1	Internet Data
56	addictionus.com	<1	Internet Data
57	archivesofrheumatology.org	<1	Internet Data
58	arxiv.org	<1	Publication
59	arxiv.org	<1	Publication
60	asbmr.onlinelibrary.wiley.com	<1	Internet Data
61	Autonomous Vehicles Virtual Simulation Platform for Training Semi-Autonomous by Dekoulis-2018	<1	Publication
62	coek.info	<1	Internet Data
63	docplayer.gr	<1	Internet Data
64	encord.com	<1	Internet Data
65	eprints.umm.ac.id	<1	Internet Data
66	FACTORS LIMITING ANIMAL PRODUCTION FROM GRAZED PASTURE by J-1975	<1	Publication
67	fdokumen.id	<1	Internet Data
68	inpressco.com	<1	Internet Data
69	mdpi.com	<1	Internet Data

70	moam.info	<1	Internet Data
71	Scientific Achievements May Not Reach Everyone Understanding Disparities in Acu by Patel-2016	<1	Publication
72	springeropen.com	<1	Internet Data
73	springeropen.com	<1	Internet Data
74	towardsdatascience.com	<1	Internet Data
75	translationproject.org	<1	Internet Data
76	www.freepatentsonline.com	<1	Internet Data
77	www.linkedin.com	<1	Internet Data
78	www.science.gov	<1	Internet Data
79	www.scribd.com	<1	Internet Data
80	IEEE 2004 Winter Simulation Conference, 2004.- Washington, D.C. (D, by Mollaghasemi, M. G- 2004	<1	Publication



JOURNAL OF INFORMATION AND COMPUTATIONAL SCIENCE

Journal of Information and Computational Science

UGC - Care Group - II Certified Journal

ISSN NO: 1548-7741 / web : www.joics.org / E-mail : submitjoics@gmail.com

Certificate of Publication

This is to certify that the paper entitled

Anticipating Hepatic disorder through Machine Learning Classification Techniques

Authored by :

S.RUBIA PARVEEN, ASSISTANT PROFESSOR

From

DEPT. OF CSE, RGMCET, NANDYAL

Has been published in

JOURNAL OF INFORMATION AND COMPUTATIONAL SCIENCE,VOLUME 14 ISSUE 04,APRIL - 2024

ज्ञान-विज्ञान विमुक्तये

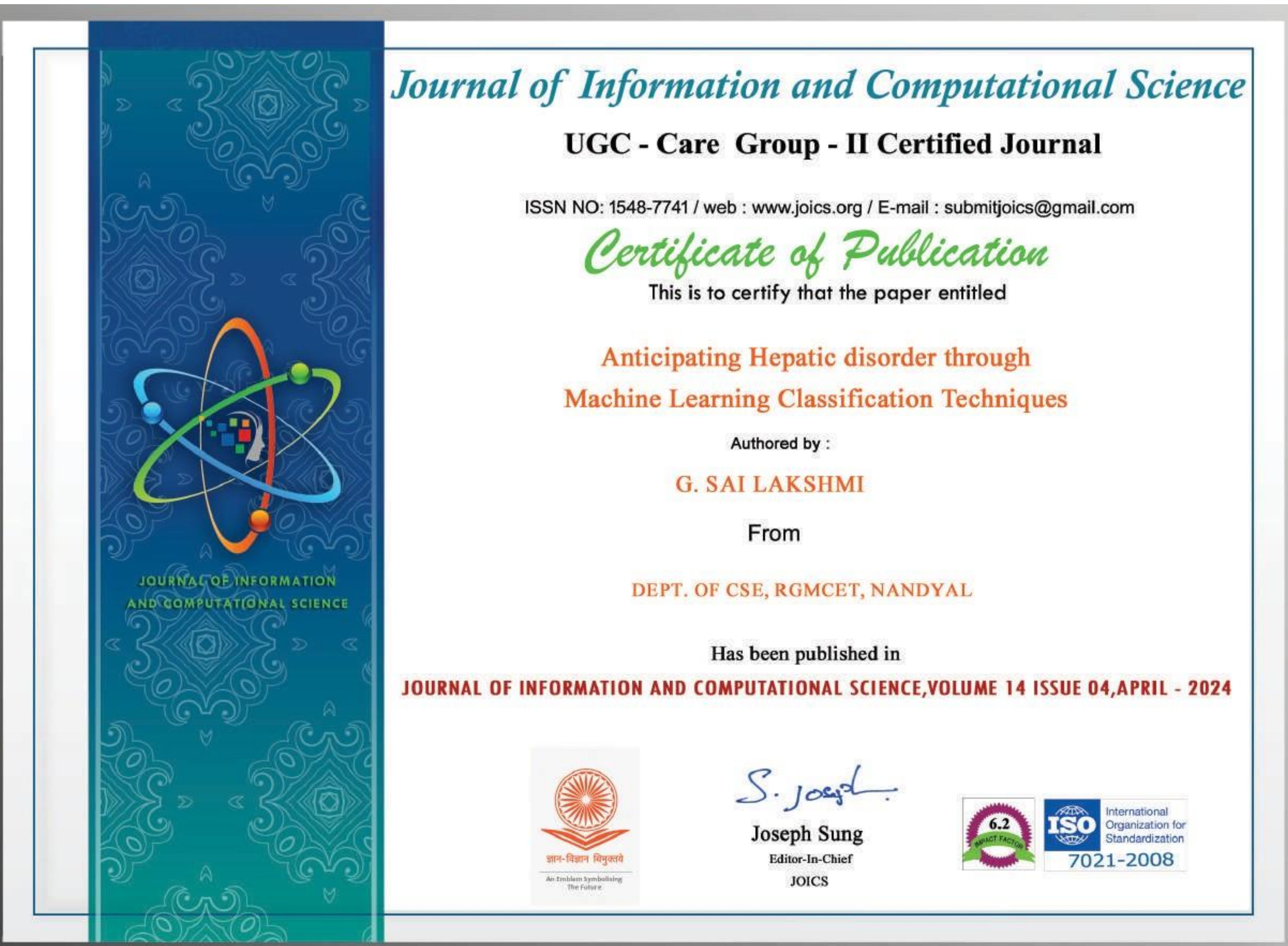
An Emblem Symbolising The Future

S. Joseph

Joseph Sung
Editor-In-Chief
JOICS

6.2 IMPACT FACTOR

ISO International Organization for Standardization 7021-2008



Journal of Information and Computational Science

UGC - Care Group - II Certified Journal

ISSN NO: 1548-7741 / web : www.joics.org / E-mail : submitjoics@gmail.com

Certificate of Publication

This is to certify that the paper entitled

Anticipating Hepatic disorder through
Machine Learning Classification Techniques

Authored by :

G. SAI LAKSHMI

From

DEPT. OF CSE, RGMCET, NANDYAL

Has been published in

JOURNAL OF INFORMATION AND COMPUTATIONAL SCIENCE,VOLUME 14 ISSUE 04,APRIL - 2024

JOURNAL OF INFORMATION AND COMPUTATIONAL SCIENCE

ज्ञान-विज्ञान विमुक्तये

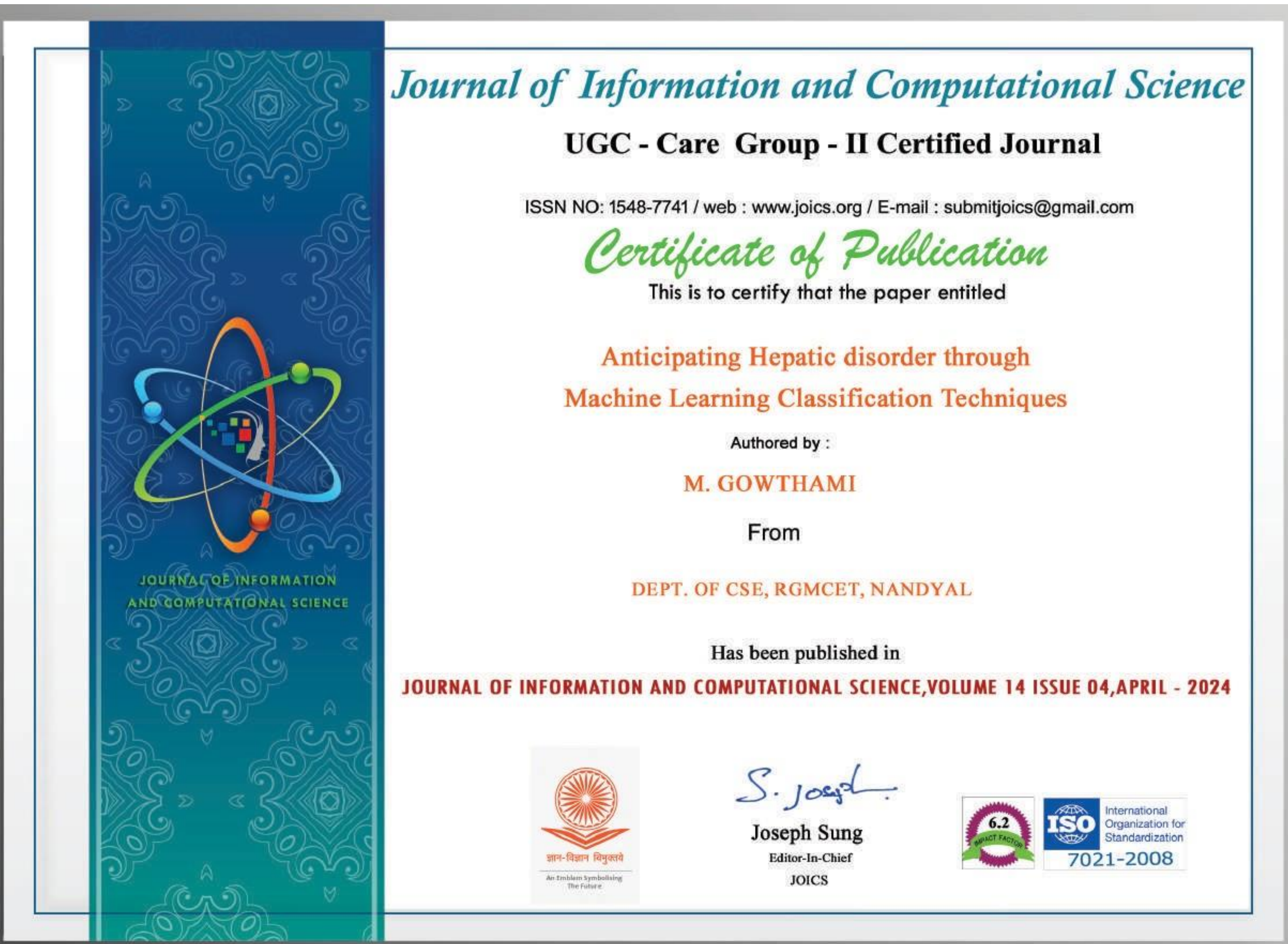
An Emblem Symbolising The Future

[illegible]

Joseph Sung
Editor-In-Chief
JOICS

6.2 IMPACT FACTOR

ISO International Organization for Standardization 7021-2008



Journal of Information and Computational Science

UGC - Care Group - II Certified Journal

ISSN NO: 1548-7741 / web : www.joics.org / E-mail : submitjoics@gmail.com

Certificate of Publication

This is to certify that the paper entitled

Anticipating Hepatic disorder through Machine Learning Classification Techniques

Authored by :

M. GOWTHAMI

From

DEPT. OF CSE, RGMCET, NANDYAL

Has been published in

JOURNAL OF INFORMATION AND COMPUTATIONAL SCIENCE,VOLUME 14 ISSUE 04,APRIL - 2024

JOURNAL OF INFORMATION AND COMPUTATIONAL SCIENCE

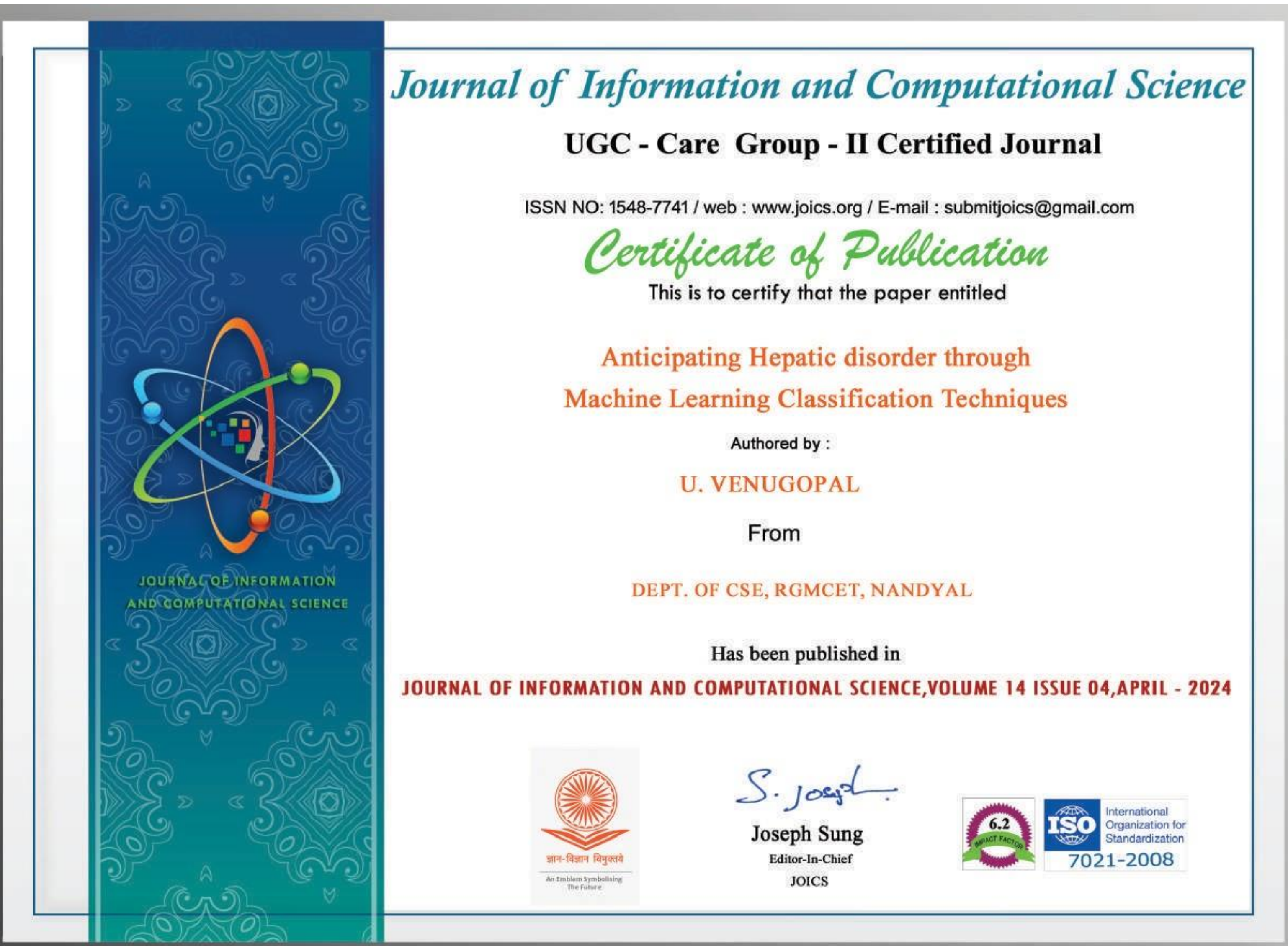
S. Joseph

Joseph Sung
Editor-In-Chief
JOICS

ज्ञान-विज्ञान विमुक्तये
An Emblem Symbolising The Future

6.2 IMPACT FACTOR

ISO International Organization for Standardization 7021-2008



Journal of Information and Computational Science

UGC - Care Group - II Certified Journal

ISSN NO: 1548-7741 / web : www.joics.org / E-mail : submitjoics@gmail.com

Certificate of Publication

This is to certify that the paper entitled

Anticipating Hepatic disorder through Machine Learning Classification Techniques

Authored by :

U. VENUGOPAL

From

DEPT. OF CSE, RGMCET, NANDYAL

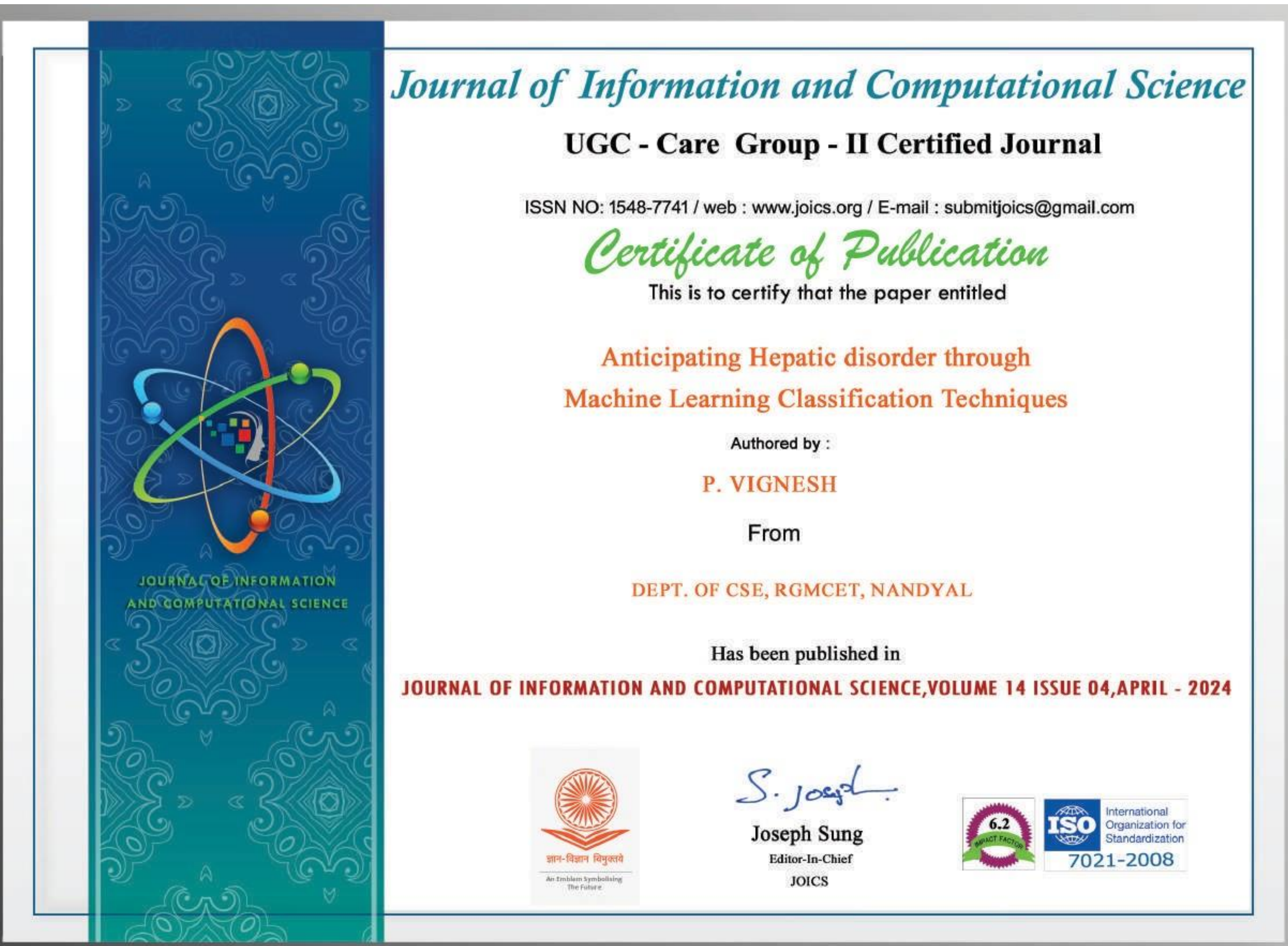
Has been published in

JOURNAL OF INFORMATION AND COMPUTATIONAL SCIENCE,VOLUME 14 ISSUE 04,APRIL - 2024

[illegible]

Joseph Sung
Editor-In-Chief
JOICS

JOURNAL OF INFORMATION AND COMPUTATIONAL SCIENCE
ज्ञान-विज्ञान विमुक्तये
An Emblem Symbolising The Future
6.2
IMPACT FACTOR
ISO
International Organization for Standardization
7021-2008


JOURNAL OF INFORMATION
AND COMPUTATIONAL SCIENCE

Journal of Information and Computational Science

UGC - Care Group - II Certified Journal

ISSN NO: 1548-7741 / web : www.joics.org / E-mail : submitjoics@gmail.com

Certificate of Publication

This is to certify that the paper entitled

Anticipating Hepatic disorder through
Machine Learning Classification Techniques

Authored by :

P. VIGNESH

From

DEPT. OF CSE, RGMCET, NANDYAL

Has been published in

JOURNAL OF INFORMATION AND COMPUTATIONAL SCIENCE,VOLUME 14 ISSUE 04,APRIL - 2024

ज्ञान-विज्ञान विमुक्तये
An Emblem Symbolising The Future

[illegible]

Joseph Sung
Editor-In-Chief
JOICS

6.2 IMPACT FACTOR

ISO International Organization for Standardization 7021-2008

Anticipating Hepatic disorder through Machine Learning Classification Techniques

S. Rubia Parveen M. Tech., Assistant Professor, Dept. of CSE, RGMCET, Nandyal, ruby0561@gmail.com

G. Sai Lakshmi, Dept. of CSE, RGMCET, s9819673@gmail.com

M. Gowthami, Dept. of CSE, RGMCET, gowthamimanubolu69@gmail.com

U. Venugopal, Dept. of CSE, RGMCET, ugopalv7@gmail.com

P. Vignesh, Dept. of CSE, RGMCET, vigneshpinni@gmail.com

***Abstract*-The liver plays a crucial role in maintaining overall health and well-being. Early detection of liver disease is crucial for reducing mortality rates. In this study, we introduce a groundbreaking approach to predicting liver disease utilizing machine learning algorithms and implementing the predictive model via a Flask application. Our research employs four well-known algorithms: Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Random Forest Classifier. Each algorithm possesses unique characteristics in handling classification tasks and is applied to forecast the occurrence of liver disease. The results of our study underscore the potential of machine learning algorithms in accurately predicting liver disease. Early diagnosis facilitated by these algorithms can significantly impact reducing mortality rates associated with liver disease.**

Keywords-- Machine learning algorithms, Liver disease, Prediction, Flask application, Classification models.

I. INTRODUCTION

The liver, a vital organ located in the upper right part of the abdominal cavity, is body's second-largest organ after the skin. Besides being the largest gland that secretes hormones, the liver performs over 500 functions that are essential for the body's proper functioning and survival. In adults, the liver typically weighs about 2% of body weight, with males weighing around 1.4 – 1.8 kgs and females around 1.2 – 1.4 kgs, while newborns have a liver weight of 150 g. The liver's functions include bile and glycogen secretion, serum protein lipid synthesis, blood detoxification, and storage of essential vitamins like D, A, K, E, and B12. Additionally, the liver has the remarkable ability to regenerate to its original size within 5-7 days if two-thirds of it is removed. Liver disease is characterized by liver swelling caused by toxic substances, bacteria, or inherited conditions that impair the organ's function critical for digestion and waste elimination. Commonly affecting individuals in the 40-60 age group, liver diseases are more prevalent in men, with approximately 10 lakh people diagnosed with liver disease annually in India, resulting in 1.4 lakh deaths per year.

Liver disease is a widespread health concern that impacts millions of individuals globally, placing a significant strain on healthcare systems. The early detection and accurate prediction of liver disease are essential for effective management and treatment, helping to minimize its negative impact on patients health and quality of life. Conventional diagnostic methods often involve invasive procedures and may not always provide timely or comprehensive insights into disease progression. In recent years, the emergence of machine learning (ML) algorithms has transformed medical diagnostics by enabling the development of predictive models that can analyze extensive clinical data to identify disease-indicative patterns and trends. These ML-based approaches have the potential to improve the accuracy and efficiency of liver disease prediction, facilitating early intervention and personalized patient care.

In the healthcare sector, Machine Learning algorithms have demonstrated potential in improving treatment accuracy and enhancing patient care through advanced diagnostic methods. By leveraging classification techniques, Machine Learning can assist in the early detection of liver diseases, a challenge due to the organ's functionality despite partial damage. Early diagnosis of liver issues can significantly increase patient survival rates, particularly by identifying liver disease markers in blood enzyme levels.By utilizing liver patient data, various Machine Learning models, including Logistic Regression, KNNeighbor algorithm, Random Forest Classifier, Support Vector Machine, were compared for predicting liver disease in patients. Here we use 583 records from the Indian Liver Patient Dataset (ILPD). This dataset contains information on 416 liver patients and 167 non-liver patients from Andhra Pradesh, India. By utilizing the ILPD dataset and applying Machine Learning techniques, the study aimed to improve liver disease prediction accuracy and contribute to enhanced patient care outcomes.

II. LITERATURE SURVEY

The Light GB algorithm has shown great potential for early detection and intervention, as suggested by Ketan Gupta, Nasmin Jiwani, Neda Afreen, and Divyarani in their research. Light GB, offer robust predictive capabilities. This algorithm has demonstrated superior performance metrics, including accuracy, sensitivity, and specificity, making it ideal for predicting liver disease progression. The success of liver disease prediction using machine learning algorithms lies in effective feature selection and model evaluation. By carefully selecting relevant features and evaluating the model's performance, healthcare professionals can enhance predictive accuracy and improve diagnostic outcomes for patients. One of the main challenges associated with Light GB is the necessity for meticulous parameter tuning. This process requires careful consideration and adjustment to ensure optimal performance of the algorithm. Additionally, there is a potential for overfitting, especially when dealing with limited dataset sizes. This can lead to inaccurate predictions and hinder the effectiveness of the algorithm.

Jayakumar Sadhasivam used Medical Mining Support Vector Machine (MMSVM) machine learning algorithm. MMSVM, have shown promising potential in accurately predicting liver diseases. However, there are several challenges in liver disease prediction. One of the key challenges in liver disease prediction is dealing with imbalanced datasets, which can impact the accuracy of the predictions. Additionally, feature selection and model interpretability are important factors to consider when utilizing machine learning algorithms for liver disease prediction. While MMSVM shows promise in accurately predicting liver diseases, its implementation comes with its own set of challenges. Careful parameter tuning is essential to ensure the accuracy of predictions, especially when dealing with large-scale datasets. Furthermore, generalizability across diverse patient populations is crucial for the effectiveness of liver disease prediction models using this algorithm. MMSVM used for accurate liver disease prediction.

Thirunavukkarasu and Ajay used logistic regression (LR) and k-nearest neighbors (KNN) algorithms. These algorithms play a crucial role in medical diagnosis, with LR being a classical statistical technique known for its simplicity and interpretability, making it ideal for binary classification tasks such as liver disease prediction.LR, although effective, assumes linear relationships between features and the target variable. This assumption can limit its effectiveness with nonlinear data. Despite this limitation, LR remains a popular choice in medical applications due to its ease of interpretation and application in clinical settings. On the other hand, KNN is a non-parametric method that does not rely on assumptions about data distribution. This makes KNN adept at capturing nonlinear relationships effectively, making it a valuable tool for liver disease prediction. Additionally, KNN is intuitive and straightforward to implement, making it a desirable option for healthcare professionals. Despite its strengths, KNN faces challenges such as the curse of dimensionality. As the number of features increases, KNN becomes computationally expensive. Moreover, the performance of KNN heavily relies on factors such as the choice of distance metric and the number of neighbors selected. While LR offers simplicity and interpretability, KNN excels in capturing nonlinear relationships.

III. METHODOLOGY

Data Collection- It is crucial in selecting significant records for analysis and obtaining productive knowledge through various data mining techniques. Data exploration, on the other hand, helps in summarizing data for analysis and identifying initial patterns and features. In the context of liver disease prediction, these steps play a vital role in analyzing and understanding the dataset.

Dataset Description- The liver disease prediction dataset was obtained from Kaggle, comprising 583 records of liver patients. The dataset includes attributes such as Age, Gender, Total Bilirubin, Direct Bilirubin, Alkphos Alkaline Phosphotase, Alamine Aminotransferase, Aspartate Aminotransferase, Total Proteins, Albumin, Albumin and Globulin Ratio, and Target.Out of the 583 records, 416 are patient records and 167 are non-patient records. The dataset is divided based on the "Dataset" column, which classifies individuals into liver patients or non-patients. Among the records, 441 are male patient records and 142 are female patient records. The data was collected from Northeast of Andhra Pradesh, India.The dataset provides valuable insights for developing predictive models to assist in the early detection and management of liver diseases.

Data Preprocessing

a. **Imputation of Missing Values**:
One of the key techniques in data preprocessing is the imputation of missing values. In the case of the Indian liver disease patients dataset, missing values for Albumin and Globulin ratio were restored using median values. This step ensures that the dataset is complete and ready for analysis.

b. **Dummy Encoding**:
Another important technique is dummy encoding, which transforms categorical variables into numerical variables. This transformation is necessary as most machine learning algorithms are designed to work with numerical data. By creating numerical variables for categorical variables, the dataset becomes more suitable for ML algorithms.

c. **Elimination of Duplicate Values**:
It is essential to eliminate redundant values from the dataset to enhance its efficiency and quality. By getting rid of duplicate values, the dataset becomes cleaner and more reliable for further analysis and prediction tasks.

d. **Outlier Detection and Elimination**: Outliers, which are exceptional values that differ from the rest of the data due to errors or anomalies, need to be detected and eliminated. There are two types of outliers: univariate and multivariate outliers. Univariate outliers focus on individual features, while multivariate outliers consider multiple dimensions of the dataset. By handling outliers, the dataset becomes more robust for accurate predictions.

e. **Resampling**: In scenarios where the dataset is imbalanced, such as in the case of liver disease prediction where the majority of patients have the disease, resampling techniques like SMOTE can be used. SMOTE helps in balancing the data by synthesizing additional samples for the minority class, ensuring that ML algorithms have sufficient data for training and prediction.

Feature Selection

In the realm of machine learning algorithms, feature selection plays a crucial role in optimizing the training process for accurate and efficient model predictions. Specifically, when it comes to liver disease prediction, leveraging ML algorithms alongside feature selection can significantly enhance the accuracy and reliability of diagnostic outcomes. Feature selection is the technique of strategically limiting the number of input variables utilized by the ML algorithm during training. By reducing the complexity of computational processes, feature selection enables the algorithm to learn more efficiently, leading to faster model training. Furthermore, this streamlined approach makes it simpler to interpret the model's decision-making process, enhancing its transparency and usability in liver disease prediction. The Liver Disease Dataset contains 11 attributes like Age, Gender, Total Bilirubin, Direct Bilirubin, Alkphos Alkaline Phosphotase, Alamine Aminotransferase, Aspartate Aminotransferase, Total Protiens, Albumin, Albumin and Globulin Ratio and Target. By excluding Age, Gender and Target remaining all attributes are selected and that should be used for identifying the liver disease.

Model Building

Developing predictive models for liver disease prediction involves utilizing selected features and leveraging various techniques such as regression, classification, clustering, and other machine learning algorithms. By integrating these advanced methodologies, accurate predictions can be generated to assist in the early detection and management of liver diseases.

After building the predictive models, it is crucial to assess their performance using appropriate evaluation metrics. This step helps in determining how effectively the models generalize to unseen data and whether they meet the desired performance criteria for accurate liver disease prediction.

Flask Application Development

Developing a Flask web application to deploy the trained model for liver disease prediction. Here the user interface is designed where user enter the details required and then the results are provided appropriately.

IV. ALGORITHMS

- **Logistic Regression:**

Logistic regression is a machine learning algorithm widely used for binary classification **tasks,** including predicting the presence or absence of liver disease. It models the relationship between input features and the probability of belonging to a certain class using the logistic function, which maps real-valued inputs to probabilities between 0 and 1. Logistic regression is suitable for binary classification tasks, where the target variable has two possible outcomes. In liver disease prediction, it predicts the probability of an individual having liver disease based on input features such as age, gender, and liver function tests.

The logistic function, also known as the sigmoid function, transforms the linear combination of input features and their corresponding weights into probabilities. It ensures that the output lies between 0 and 1, representing the likelihood of belonging to the positive class. Logistic regression separates instances belonging to different classes using a decision boundary. The threshold for classification is usually set at 0.5; instances with predicted probabilities above the threshold are assigned to the positive class, and vice versa. The coefficients in logistic regression represent the impact of each feature on the probability of belonging to the positive class. Positive coefficients indicate a positive relationship with the positive class, while negative coefficients indicate a negative relationship. In liver disease prediction, logistic regression can analyze patient features and provide probabilities of having liver disease

- **K-NEAREST NEIGHBORS**:

The K-Nearest Neighbor algorithm classifies the data based on the similarity measure. ‘K’ specifies the number of nearest neighbors to be considered. To identify the nearest points, it considers the distance metrics like Euclidean, Manhattan, Chebyshev, Hamming and so on. It does not require the data to be trained. Data needs to be normalized in order to apply KNN algorithm KNN considered to be instance based algorithm since it uses training instances to make predictions and Lazy learning algorithm as it does not require a model to be trained. For classification and regression, the K-NN is used. The entire training dataset is used to build a model for K-NN. For example, when the unseen data is needed for prediction, the K-NN will try to find through the training data for the K most comparable times. As a result, it summarizes and returns the predicted attributes of the most similar one as output and the similarity is based on the type of facts.

To classify the liver disease, Euclidean distance method was used to calculate the distance between each data and assuming k value from 0 to 3 as shown in Fig 5. k-value and possibility of having the disease are denoted by X-axis and the Y-axis respectively. The "1" represents the liver disease and "2" represents the non-liver disease. For example, if k=0 means there is chance of 50% of getting liver disease and if k=1 means there is chance of getting 60% of getting liver disease and so on.

The Euclidean distance formula is:

$d = \sqrt{[(x2 - x1)2 + (y2 - y1)2]}$

- **Support Vector Machine**:

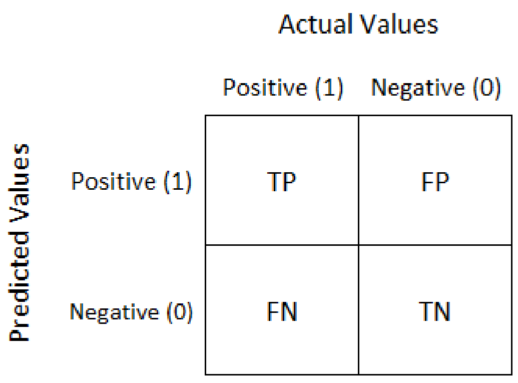
The Support Vector Machine (SVM) algorithm is a powerful supervised learning technique utilized primarily for classification tasks. SVM aims to identify an optimal hyperplane that separates the different classes in the training data with the widest possible margin. This hyperplane is determined by support vectors, which are the data points closest to the decision boundary. By maximizing the margin, SVM aims to improve the generalization ability of the model and enhance its performance on unseen data. In cases where the data is not linearly separable, SVM employs kernel functions to map the input data into a higher-dimensional feature space, enabling the identification of non-linear decision boundaries. Various kernel functions such as linear, polynomial, and radial basis function (RBF) can be utilized depending on the problem's characteristics. SVM also incorporates regularization parameters to prevent overfitting and fine-tune the trade-off between maximizing the margin and minimizing classification errors. Despite its effectiveness, SVM may be computationally expensive, particularly for large datasets, and its performance can be sensitive to the choice of kernel and regularization parameters. Nonetheless, SVM remains a versatile and widely used algorithm in various domains due to its robustness and ability to handle complex classification tasks.

- **Random Forest Classifier:**

The Random Forest Classifier is a robust and versatile ensemble learning algorithm that excels in classification tasks by constructing multiple decision trees based on random subsets of the training data and features. By leveraging the principle of ensemble averaging, Random Forests offer improved performance and generalization compared to individual decision trees, making them particularly suitable for handling high-dimensional data, noisy data, and large datasets. Through the process of bootstrapping and random feature selection, Random Forests mitigate overfitting and provide robust predictions while also offering insights into feature importance. With its scalability, ease of use, and effectiveness in various domains, the Random Forest Classifier stands as a popular and reliable choice for machine learning practitioners seeking accurate and stable classification models. It avoids the over-fitting problem by constructing a decision tree from a random data sample.

V. PERFORMANCE EVALUATION

Confusion matrix was used to represent the analysis as specified in Fig 1. The confusion matrix defines the performance of a classifier model. This consists of four different combinations of actual and predicted values**.**



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Fig 1: Confusion matrix table

True Positive (TP) means that the algorithm could correctly identifies the patient while False Positive (FP) specifies that the algorithm incorrectly identifies a liver patient. True Negative (TN) means it correctly rejects a patient who has liver disease and False Negative (FN) means incorrectly rejects that a patient has liver disease. Accuracy, Precision, Recall, F1-score, Support.

Accuracy: Accuracy is the proportion of correctly classified instances among all instances in the dataset. It measures the overall correctness of the classifier regardless of the class labels. A high accuracy indicates a good overall performance of the classifier.

Accuracy = (True Positives + True Negatives) / Total Instances

Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positives. It answers the question, Of all the instances predicted as positive, how many are actually positive. A high precision indicates that the classifier has a low false positive rate, meaning it doesn't label negative samples as positive very often.

Precision = (True Positives) / (True Positives + False Positives)

Recall: Recall is the ratio of correctly predicted positive observations to all actual positives. It answers the question, "Of all the actual positive instances, how many did we predict as positive?" A high recall indicates that the classifier has a low false negative rate, meaning it successfully identifies most positive samples.

Recall = (True Positives) / (True Positives + False Negatives)

F1-score: F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall. A high F1-score indicates both high precision and high recall.

F1-score = 2 * (Precision * Recall) / (Precision + Recall)

Support: Support simply refers to the number of actual occurrences of each class in the specified dataset.

Algorithm	**Accuracy**
Logistic Regression	96
KNN	68.53
Support Vector Machine	74.12
Random ForestClassifier	100

VI. RESULTS

Fig 2 tells about the Data visualization is the representation of data or information in a graph, chart, or other visual format. Machine learning makes it easier to conduct analyses such as predictive analysis, which can then serve as helpful visualizations to present.

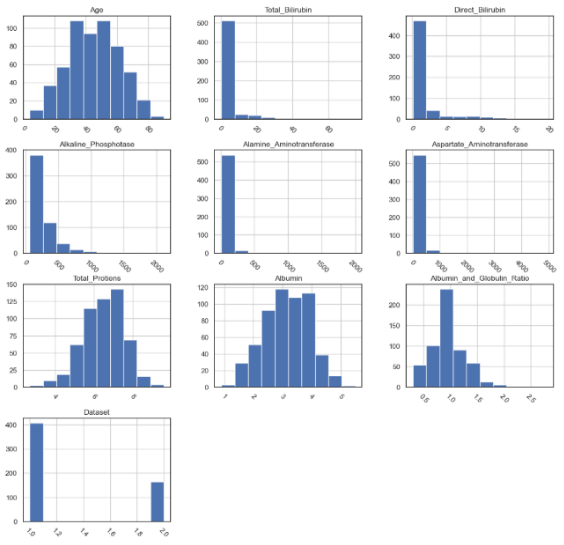


Fig 2: Data Visualisation plot

Fig 3 tells about the Feature selection is the process of reducing the number of input variables when developing a predictive model. It is desirable to reduce the number of input variables to both reduce the computational cost of modelling and, in some cases, to improve the performance of the model.

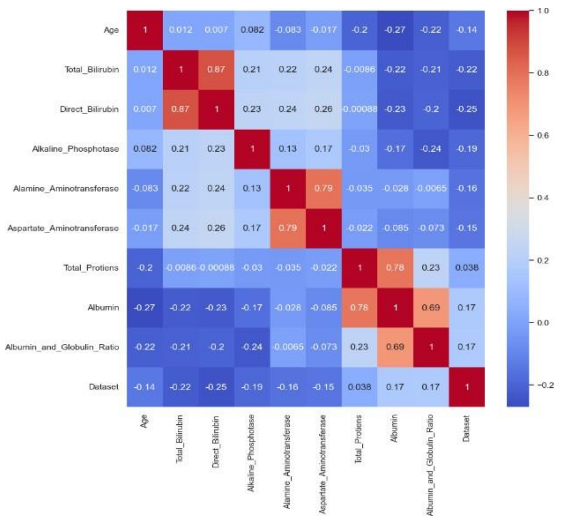


Fig 3: Feauture Selection

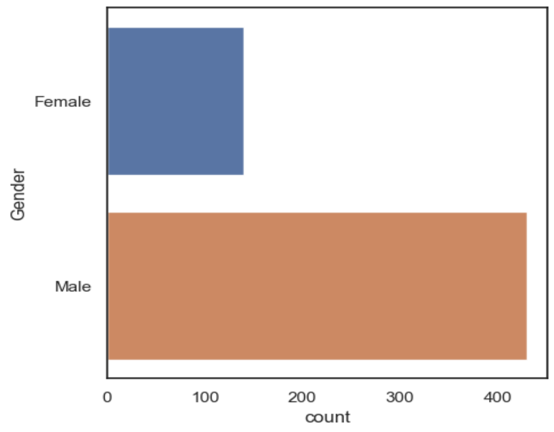


Fig 4: Gender count

Fig 5 describes about the user interface where the user enter the valid details and get the information regarding the prsence of liver disease

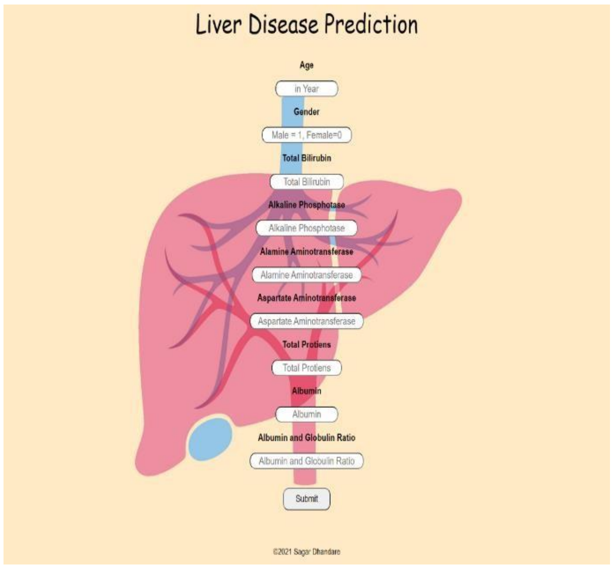


Fig 5: User Interface

VII. CONCLUSION

Deploying a machine learning algorithm for liver disease prediction, with the exceptional performance metrics achieved, presents a significant advancement in healthcare technology. Through a Flask application, this model can be seamlessly integrated into clinical workflows, offering healthcare professionals a reliable tool for diagnosing liver disease based on patient symptoms and test results. The application's user-friendly interface ensures accessibility for both healthcare providers and patients, fostering timely interventions and improving patient outcomes. By leveraging real-time data and user feedback, continuous improvement of the predictive model becomes possible, enhancing its accuracy and reliability over time. This deployment not only facilitates early detection and treatment of liver disease but also underscores the potential of machine learning in revolutionizing healthcare delivery and patient care.

VIII. FEATURE SCOPE

The feature scope for deploying a liver disease prediction model using machine learning algorithms through a Flask application is extensive and impactful. This comprehensive solution incorporates user authentication, intuitive input interfaces, and robust prediction output displays to ensure secure and accessible utilization by healthcare professionals and patients. Integration with healthcare systems facilitates seamless access to patient data, while real-time updates and visualization tools enhance decision-making processes. Feedback mechanisms and model management functionalities enable continuous improvement and adaptation to evolving healthcare needs. With stringent security measures and compliance with data privacy regulations, the application safeguards patient confidentiality. Ultimately, this deployment represents a significant advancement in healthcare technology, offering valuable insights for early detection, diagnosis, and management of liver disease, thereby enhancing patient care and healthcare delivery effectiveness.

IX. REFERENCES

a. Ketan Gupta, Nasmin Jiwani, Neda Afreen & Divyarani D "Liver Disease Prediction using Machine learning Classification Techniques" in 11th IEEE International Conference on Communication Systems and Network Technologies.

b. Nasmin Jiwani, Ketan Gupta , Neda Afreen "A Convolutional Neural Network Approach for Diabetic Retinopathy Classification" in 2022 IEEE 11th International Conference on Communication Systems and Network Technologies (CSNT).

c. Rakshith DB, Mrigank Srivastava, Ashwani Kumar & Gururaj S P "Liver Disease Prediction System using Machine Learning Techniques" in https://www.ijert.org/liver-disease-prediction-system-using-machine-learning-techniques in volume 10 issue 6(2021).

d. Jayakumar Sadhasivam, J.Senthil, R.M.Ganesh, N.Chellapan "Liver Disease Prediction Using MachineLearningClassification" in https://www.webology.org/datacms/articles/20211222042439pmWEB18293.pdf.

e. Thirunavukkarasu K, Ajay S. Singh, Md Irfan, Abhishek Chowdhury "Prediction of Liver Disease using Classification Algorithms" in 2018 4th International Conference on Computing Communication and Automation (ICCCA).