A

Mini Project

On

COMPARATIVE ANALYSIS OF LIVER DISEASE PREDICTION USING MACHINE LEARNING

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

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

CH. Rushmitha (217R1A05M0)

D. Rishik Kumar(217R1A0M2)

A. Murali (227R5A0526)

Under the Guidance of

D. Sandhya Rani

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi)

Recognized Under Section 2(f) & 12(B) of the UGCAct.1956,

Kandlakoya (V), Medchal Road, Hyderabad-501401.

2021-25

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**COMPARATIVE ANALYSIS OF LIVER DISEASE PREDICTION MACHINE LEARNING"** being submitted by **CH. Rushmitha (217R1A05M0), D. Rishik Kumar(217R1A0M2) and A. Murali(227R5A0526)** in partial fulfillment of the requirements for the award of the degree of B. Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2024-25.

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

Mrs. D. Sandhya Rani
(Assistant Professor)
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. N. Bhaskar
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on _____

ACKNOWLEDGEMENT

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

We take this opportunity to express my profound gratitude and deep regard to our guide **Mrs. D. Sandhya Rani,** Assistant Professor for her exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by her shall carry us a long way in the journey of life on which we are about to embark. We also take this opportunity to express a deep sense of gratitude to Project Review Committee (PRC) Coordinators: **Dr. J. Narasimha Rao, Dr. K. Maheshwari, Mr. K. Ranjith Reddy, Mrs. K. Shilpa** for their cordial support, valuable information and guidance, which helped usin completing this task through various stages.

We are also thankful to **Dr. N. Bhaskar,** Head of the Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

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

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

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

CH. RUSHMITHA(217R1A05M0)
D. RISHIK KUMAR(217R1A05M2)
A. MURALI(227R5A0526)

ABSTRACT

The diagnosis of liver diseases is a critical area of medical research and practice, necessitating accurate and timely identification to improve patient outcomes. Leveraging advancements in machine learning (ML), this study explores the potential of ML techniques to enhance the diagnosis of liver diseases. Various ML algorithms, including decision trees, support vector machines, and neural networks, are applied to datasets comprising clinical and biochemical data of patients. The study demonstrates that ML models can effectively identify patterns and correlations within complex medical datasets, surpassing traditional diagnostic methods in accuracy and speed. Additionally, feature selection techniques are employed to determine the most significant predictors of liver disease, aiding in the development of more focused and efficient diagnostic tools. The results indicate that ML-based diagnostic systems can significantly improve the early detection and classification of liver diseases, leading to better patient management and treatment strategies. This research underscores the importance of integrating ML into clinical practice, paving the way for more personalized and precise healthcare solutions.

LIST OF FIGURES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1. INTRODUCTION

1.INTRODUCTION

1.1 PROJECT SCOPE

The scope of the project focuses on developing a machine learning-based system for predicting liver disease, particularly liver fibrosis, using patient data, including serum markers and demographic information. The project involves collecting and preprocessing datasets, implementing various machine learning algorithms and evaluating their performance using metrics like accuracy, precision, recall, and confusion matrix. It includes creating a user-friendly interface for easy dataset uploads, model training, and results display. The system aims to assist healthcare professionals in predicting liver disease severity and can be enhanced in the future with deeper data integration and advanced AI techniques for real-time clinical use.

1.2 PROJECT PURPOSE

The purpose of this project is to develop an efficient, machine learning-based system to improve the prediction and diagnosis of liver diseases, particularly liver fibrosis. By leveraging diverse algorithms and patient data such as serum markers and physical attributes, the system aims to enhance diagnostic accuracy, enable early detection, and reduce reliance on invasive diagnostic procedures. The ultimate goal is to provide healthcare professionals with a reliable tool for personalized patient care, facilitating better decision-making and improving overall patient outcomes in liver disease management.

1.3 PROJECT FEATURES

The project features a comprehensive machine learning system designed for liver disease diagnosis. Key features include data preprocessing capabilities, allowing seamless handling of patient data from multiple sources. It incorporates various machine learning algorithms, such as logistic regression, KNN, decision trees, and advanced models like XGBoost and neural networks (ANN, MLP), to predict liver disease severity. The system also supports model evaluation through accuracy, precision, recall, and confusion matrices.

2.SYSTEM ANALYSIS

2.SYSTEM ANALYSIS

SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to be done.

2.1 PROBLEM DEFINITION

Liver disease is a serious health condition that affects millions of people worldwide. Early diagnosis and treatment are critical to prevent severe complications. Traditional diagnostic methods rely heavily on manual evaluation of clinical data, which can be time-consuming, error-prone, and costly. There is a need for efficient, automated tools that can assist healthcare providers in predicting liver disease more accurately and promptly.

The project aims to explore and evaluate different machine learning algorithms that can classify whether a patient has liver disease based on various attributes such as age, gender, albumin levels, bilirubin levels, enzyme counts, etc. By comparing models, the study will determine which algorithms offer the best performance in terms of accuracy, precision, recall, and overall robustness for this predictive task.

2.2 EXISTING SYSTEM

The diagnostic systems for liver diseases rely on a combination of clinical evaluations, biochemical tests, imaging techniques, and, in some cases, invasive procedures. Clinical assessments and biochemical tests, such as blood tests measuring liver enzymes and function, provide initial indications of liver dysfunction but are often nonspecific. Imaging techniques, including ultrasound, computed tomography (CT), and magnetic resonance imaging (MRI), offer varying degrees of detail and accuracy in visualizing liver abnormalities, with ultrasound being the most accessible and cost-effective, while MRI provides the highest resolution. Liver biopsy remains the gold standard for assessing liver fibrosis and inflammation but is invasive and carries inherent risks. Additionally, non-invasive methods like Fibro Scan, which measures liver stiffness, offer an alternative to biopsy but may be affected by factors such as obesity. Despite their utility, these traditional diagnostic methods often suffer from limitations in sensitivity, specificity, and accessibility. Machine learning has the potential to address these shortcomings by integrating and analyzing diverse data sources to enhance diagnostic accuracy and efficiency.

2.2.1 LIMITATIONS OF EXISTING SYSTEM

- Lack of Specificity and Sensitivity
- Invasiveness and Risk
- Radiation Exposure
- High Costs
- Limited Accessibility
- Suboptimal Early Detection
- Time-Consuming
- Influence of Patient Factors

2.3 PROPOSED SYSTEM

To The proposed system for diagnosing liver diseases harnesses the power of machine learning (ML) to address the limitations of current diagnostic methods. By integrating diverse data sources such as imaging studies (ultrasound, CT, MRI), biochemical test results, and electronic health records (EHRs) the system aims to provide a comprehensive and accurate assessment of liver health. Advanced ML algorithms, including deep learning models like convolutional neural networks (CNNs) and supervised learning techniques such as support vector machines (SVMs), are utilized to analyze and interpret this multimodal data. The system also features predictive analytics for early detection and risk stratification, enabling timely intervention and personalized treatment plans. With a user-friendly interface designed for seamless integration into clinical workflows, the system offers real-time decision support to healthcare professionals and personalized feedback to patients. Additionally, it includes adaptive learning capabilities to continuously improve its diagnostic accuracy by incorporating new data and insights. This approach aims to enhance the diagnostic process, making it more efficient, non-invasive, and precise.

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

- Early Detection and Risk Stratification
- Non-Invasive Approach
- Personalized Treatment Planning
- Real-Time Decision Support
- Cost-Effectiveness
- Continuous Learning and Improvement
- Improved Patient Engagement
- Efficiency in Diagnosis

2.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential

Three key considerations involved in the feasibility analysis are

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

2.4.1 ECONOMIC FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2.4.2 TECHNICAL FEASIBILITY

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

2.4.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system; instead, they must accept it as a necessity. The level of acceptance by the users solely depends on the methods employed to educate them about the system and to make them familiar with it. Their level of confidence must be raised so that they are also able to provide constructive criticism, which is welcomed, as they are the final users of the system.

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS:

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

- Processor : Intel Dual Core@ CPU 2.90GHz.
- Hard disk : 16GB and Above.
- Memory : 4GB RAM and Above.

2.5.2 SOFTWARE REQUIREMENTS:

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

- Operating system : Windows 8 and Above.
- Language : Python (Version 3.7.0)

3.ARCHITECTURE

3.ARCHITECTURE

3.1 PROJECT ARCITECTURE

This This project architecture represents the flow of the machine learning process for liver disease prediction machine learning, starting from input to final classification.

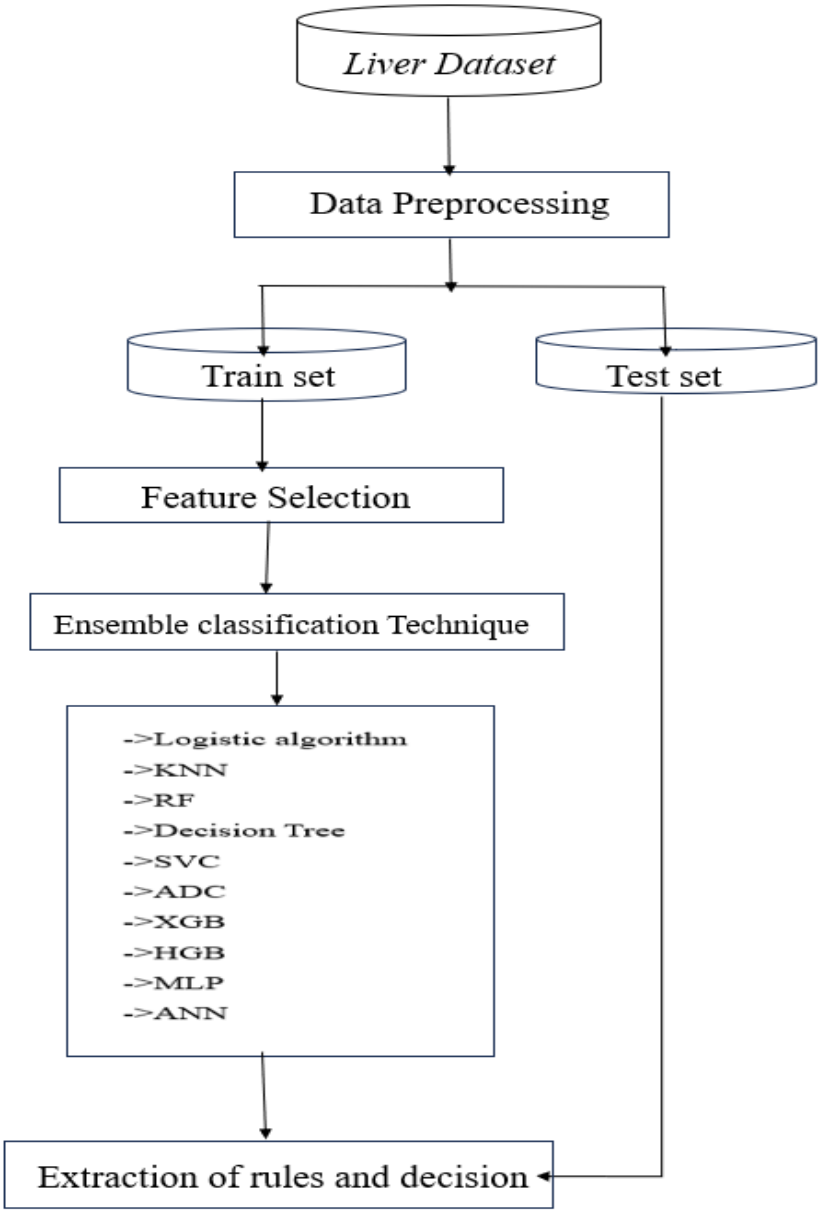


Figure 3.1: Project Architecture of Classification of Software Defined Network Traffic to Provide Quality of Service

3.2 DESCRIPTION

Liver Dataset: The input to the system is a dataset related to liver disease. This could include various features like patient demographics, blood test results, etc.

Data Preprocessing: Before using the data for training, the dataset undergoes preprocessing. This step handles missing values, outliers, data normalization, and encoding of categorical variables to ensure the dataset is clean and ready for machine learning algorithms.

Train Set and Test Set: The preprocessed data is split into two subsets:

Train Set: Used for training the machine learning models.

Test Set: Held back for evaluating the performance of the trained models.

Feature Selection: In this step, important features that contribute most to the target prediction (liver disease) are selected. This improves the model's efficiency and reduces complexity by eliminating irrelevant or redundant data.

Ensemble Classification Technique:T he system applies various classification algorithms, forming an ensemble of models. The classification algorithms include:

- Logistic Regression (Logistic algorithm)
- K-Nearest Neighbors (KNN)
- Random Forest (RF)
- Decision Tree (DT)
- Support Vector Classifier (SVC)
- AdaBoost Classifier (ADC)
- XG Boost (XGB)
- Histogram Gradient Boosting (HGB)
- Multilayer Perceptron (MLP)
- Artificial Neural Network (ANN)

Extraction of Rules and Decisions: Based on the classification techniques and their results, rules and decisions are extracted. These rules can help in understanding the critical factors contributing to liver disease and provide a decision-making framework for diagnosis.

3.3 USE CASE DIAGRAM

The use case diagram describes the interaction between the **user** and the **system** in running various machine learning models. The user can upload the dataset, preprocess it, generate training and test sets, and then run multiple classification algorithms, allowing for model comparison and selection.

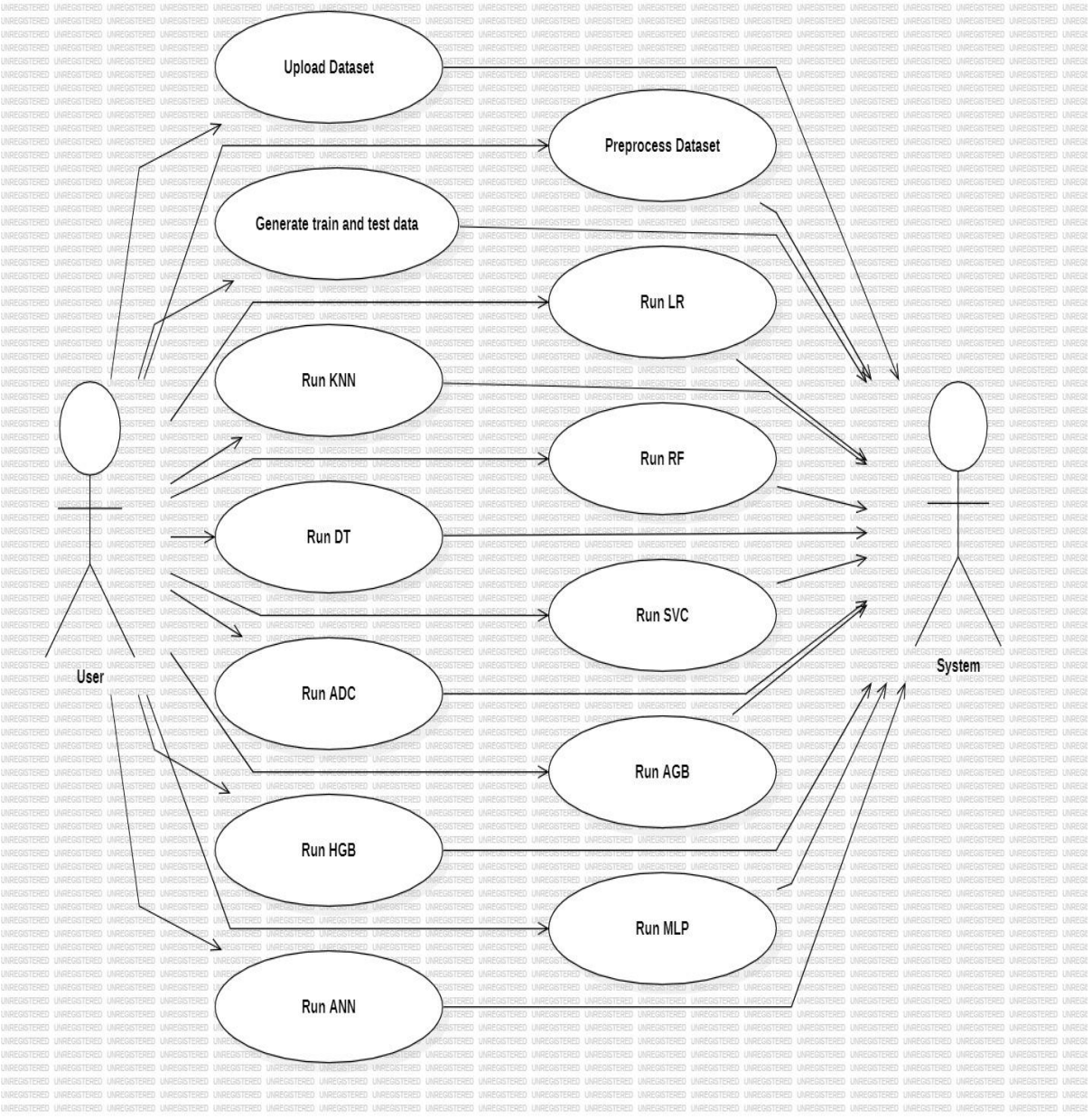


Figure 3.3: Use Case Diagram for User and System for Comparative Analysis of Liver Disease Prediction using Machine Learning

3.1 CLASS DIAGRAM

Class Diagram is a collection of classes and objects. In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.

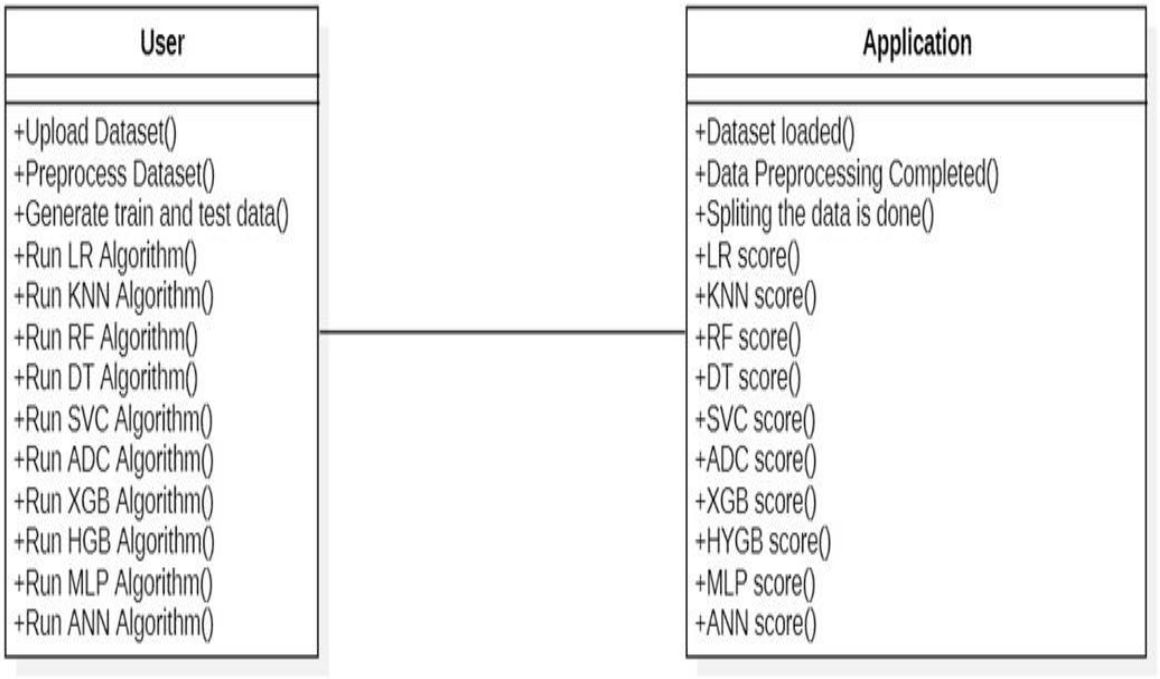


Figure 3.4: Class Diagram for User and System for Comparative Analysis of Liver Disease Prediction using Machine Learning

3.2 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

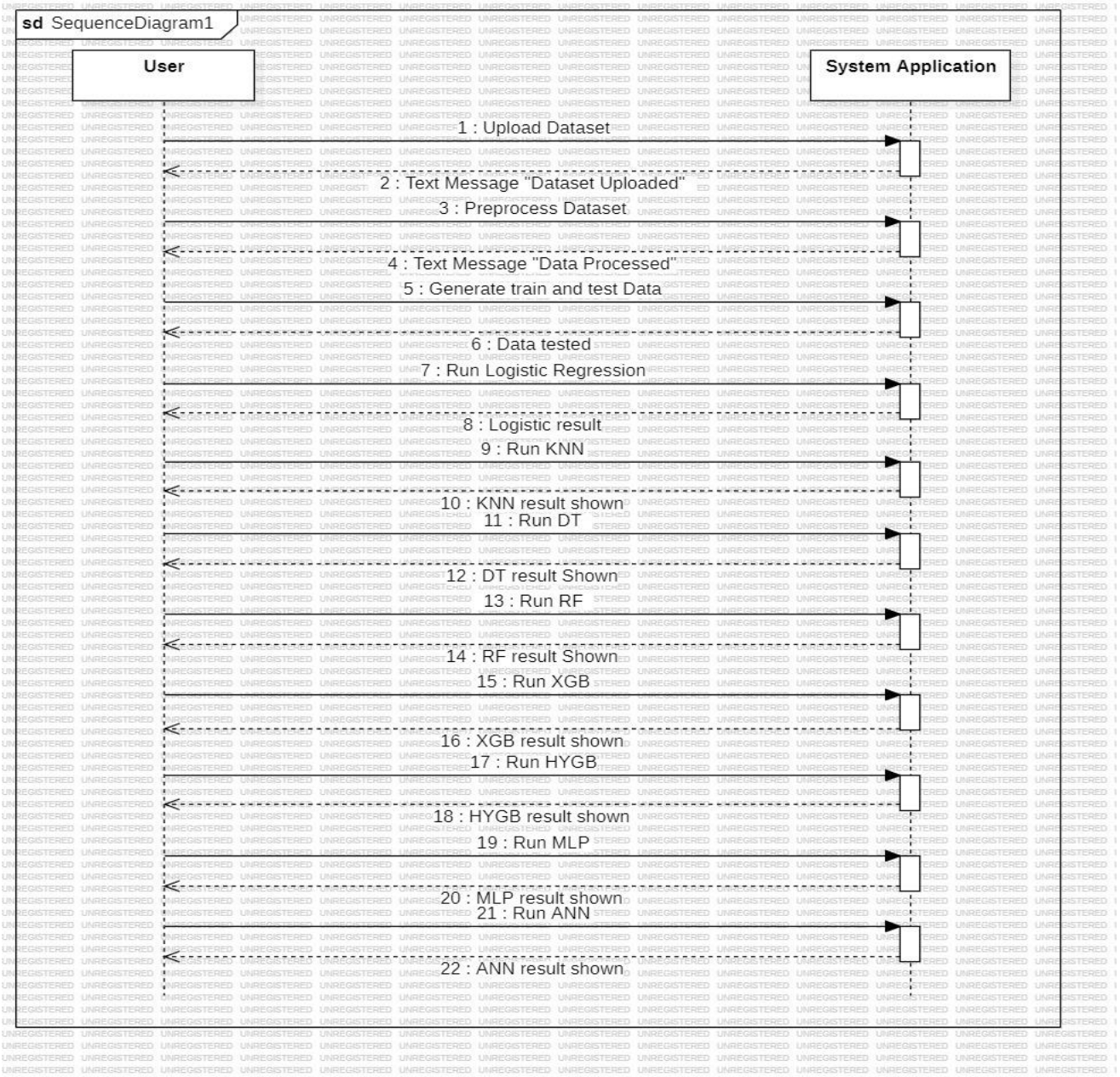


Figure 3.5: Sequence Diagram for User and System for Comparative Analysis of Liver Disease Prediction using Machine Learning

3.3 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and the actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

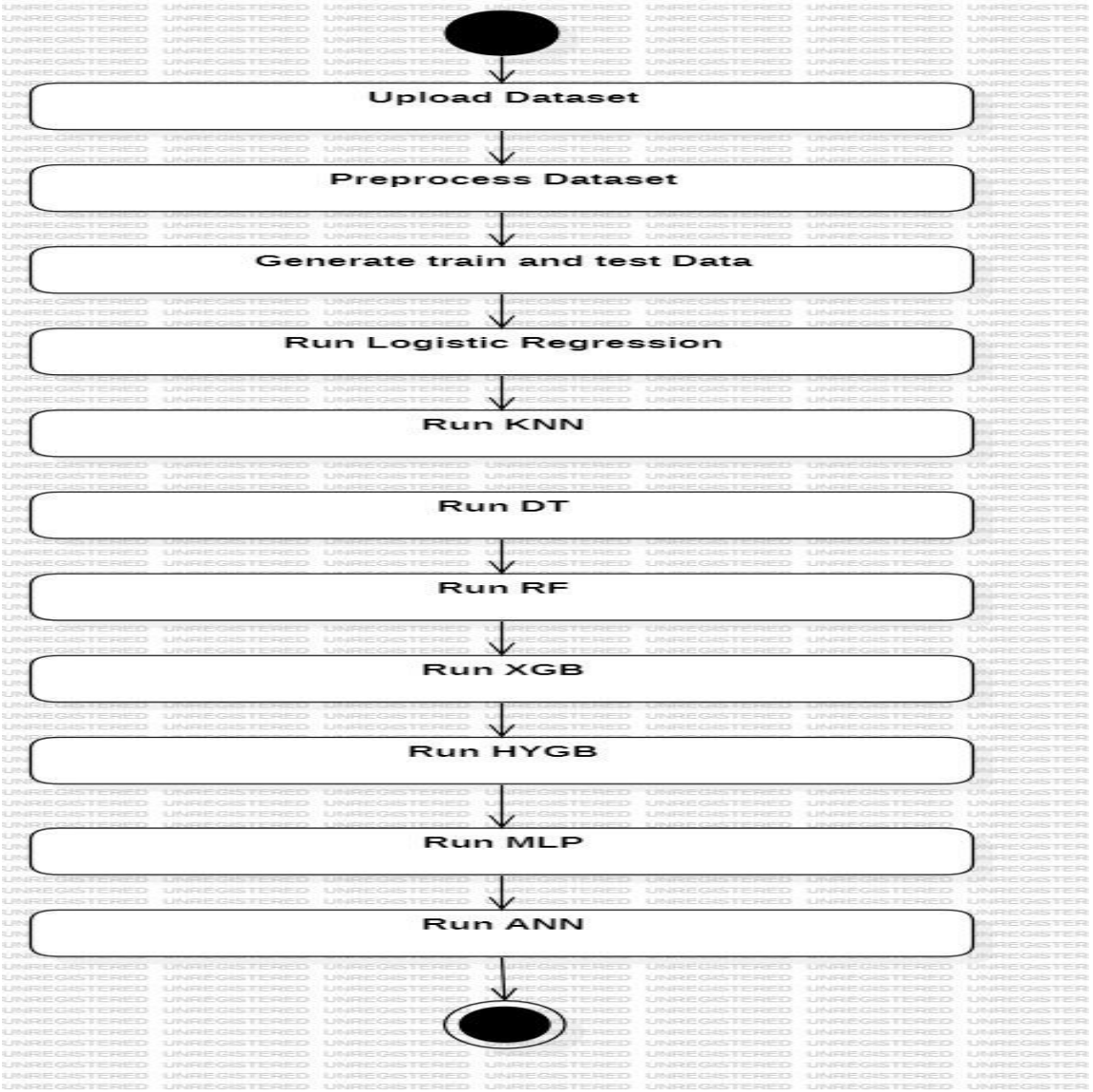


Figure 3.6: Activity Diagram for User and System for Comparative Analysis of Liver Disease Prediction using Machine Learning

4.IMPLEMENTATION

4. IMPLEMENTATION

4.1 SAMPLE CODE

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
from imutils import paths
from tkinter.filedialog import askopenfilename
from tkinter import scrolledtext
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.svm import SVC
from sklearn.preprocessing import MinMaxScaler
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
#get_ipython().run_line_magic('matplotlib', 'inline')
import scikitplot as skplt
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from matplotlib import pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.svm import LinearSVC
from sklearn.preprocessing import StandardScaler
from sklearn.experimental import enable_hist_gradient_boosting  # noqa
from sklearn.ensemble import HistGradientBoostingClassifier
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.models import load_model
```

```
import os
from os import path

main = tkinter.Tk()
main.title("Severity of Liver Fibrosis for Chronic HBV based on Physical Layer with Serum Markers")
main.geometry("1300x1200")
global filename
global raw_data
global X, y, X_train, X_test, y_train, y_test
global ltsm_acc, ann_acc, mlp_acc,cnn_acc
global MODEL_PATH
def upload():
    global filename
    text.delete('1.0', END)
    filename = askopenfilename(initialdir="dataset")
    pathlabel.config(text=filename)
    text.insert(END, "Dataset loaded\n\n")
def get_redundant_pairs(df):
    '''Get diagonal and lower triangular pairs of correlation matrix'''
    pairs_to_drop = set()
    cols = df.columns
    for i in range(0, df.shape[1]):
        for j in range(0, i+1):
            pairs_to_drop.add((cols[i], cols[j]))
    return pairs_to_drop
def get_top_abs_correlations(df, n=5):
    labels_to_drop = get_redundant_pairs(df)
    au_corr = au_corr.drop(labels=labels_to_drop).sort_values(ascending=False)
    return au_corr[0:n]
def preprocess():
    global filename
    global raw_data
    global X,y
text.delete('1.0',END)
    text.insert(END,"Importing dataset\n")
    raw_data = pd.read_excel(filename)
    text.insert(END,"Data column information: "+str(raw_data.columns)+"\n\n")
    text.insert(END,"data shape"+str(raw_data.shape)+"\n\n")
    raw_data['Gender'] = raw_data['Gender'].map({'F': 0, 'M': 1})
    text.insert(END,"Top Absolute Correlations")
    text.insert(END,"Top Correlation values: "+str(get_top_abs_correlations(raw_data, 10))+"\n\n")
```

```
raw_data.isnull().sum()
    raw_data = raw_data.drop(['Weight','Obesity', 'Waist','Bad Cholesterol'],axis=1)
    raw_data.dtypes
    print("Top Absolute Correlations")
    print(get_top_abs_correlations(raw_data, 10))
    raw_data.isnull().sum()
    cols_mode = ['Hepatitis', 'Diabetes', 'HyperTension', 'Education', 'Unmarried','PoorVision','Income']
    for column in cols_mode:
        raw_data[column].fillna(raw_data[column].mode()[0], inplace=True)
    cols_mode = ['Height', 'Body Mass Index', 'Maximum Blood Pressure', 'Minimum Blood Pressure', 'Good
    for column in cols_mode:
        raw_data[column].fillna(raw_data[column].mean(), inplace=True)
    raw_data.isnull().sum()
    raw_data.dtypes
    y = raw_data['ALF']
    raw_data.drop(columns=['ALF'],inplace=True)
    X = raw_data
    y = y[:6000]
def dataSplit():
    global X,y
    X = pd.DataFrame(scaler.transform(X),columns=X.columns)
    X.head()
    X_pred = X[:6000]
    X = X[:6000]
    X_train, X_test, y_train, y_test = train_test_split(X,y,stratify = y,shuffle=True ,test_size=0.2)
    text.insert(END,"Spliting the data is done")
def logit():
    global X_train, X_test, y_train, y_test
    text.delete('1.0',END)
    lr = LogisticRegression()
    lr.fit(X_train, y_train)
    text.insert(END,"Score of logistic Algo: "+str(lr.score(X_test, y_test))+"\n\n")
    y_pred = lr.predict(X_test)
    text.insert(END,"Classification Report: "+str(classification_report(y_test, y_pred))+"\n\n")
    text.insert(END,"Confusion Matrix : "+str(confusion_matrix(y_test,y_pred))+"\n\n")
def xgb():
    global X_train, X_test, y_train, y_test
    text.delete('1.0',END)
    xgb = XGBClassifier(random_state=10)
    text.insert(END,"Classification Report: "+str(classification_report(y_test, y_pred))+"\n\n")
    text.insert(END,"Confusion Matrix : "+str(confusion_matrix(y_test,y_pred))+"\n\n")
```

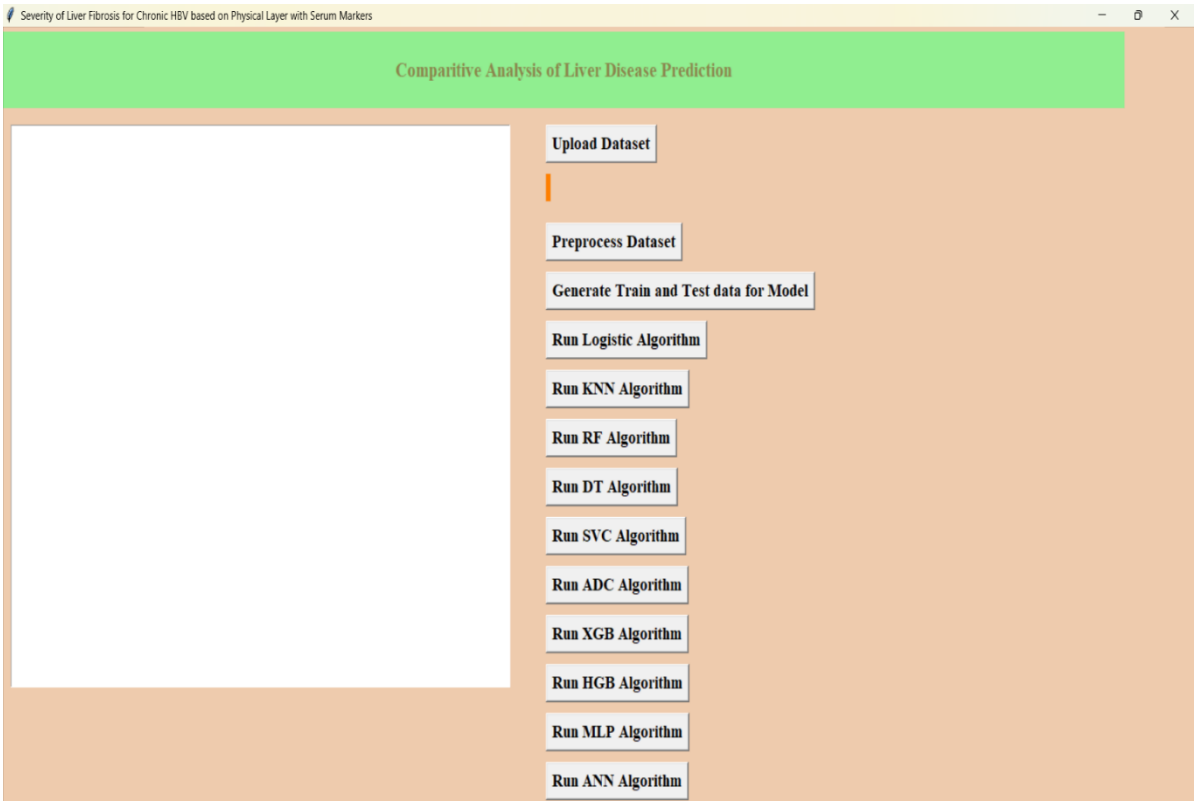
```
def knn():
    global X_train, X_test, y_train, y_test
    text.delete('1.0',END)
    k_range = range(1,15)
    scores = {}
    scores_list = []
    for k in k_range:
        knn =  KNeighborsClassifier(n_neighbors = k)
        knn.fit(X_train, y_train)
        y_predict = knn.predict(X_test)
        scores[k] = metrics.accuracy_score(y_test, y_predict)
        scores_list.append(metrics.accuracy_score(y_test, y_predict))
        text.insert(END,"Score of KNN : "+str(knn.score(X_test, y_test))+"  K value : "+str(k)+"\n\n")
    knn =  KNeighborsClassifier(n_neighbors = 5)
    knn.fit(X_train, y_train)
    text.insert(END,"Score of KNN Algo: "+str(knn.score(X_test, y_test))+"\n\n")
    y_pred = knn.predict(X_test)
    text.insert(END,"Classification Report: "+str(classification_report(y_test, y_pred))+"\n\n")
    text.insert(END,"Confusion Matrix : "+str(confusion_matrix(y_test,y_pred))+"\n\n")
def dt():
    global X_train, X_test, y_train, y_test
    text.delete('1.0',END)
    dt = DecisionTreeClassifier(random_state=0)
    dt.fit(X_train,y_train)
    text.insert(END,"Score of DT Algo: "+str(dt.score(X_test, y_test))+"\n\n")
    y_pred = dt.predict(X_test)
    text.insert(END,"Classification Report: "+str(classification_report(y_test, y_pred))+"\n\n")
    text.insert(END,"Confusion Matrix : "+str(confusion_matrix(y_test,y_pred))+"\n\n")
def rf():
    global X_train, X_test, y_train, y_test
    text.delete('1.0',END)
    rf = RandomForestClassifier(n_estimators = 10,max_depth=2, random_state=0)
    rf.fit(X_train, y_train)
    text.insert(END,"Classification Report: "+str(classification_report(y_test, y_pred))+"\n\n")
    text.insert(END,"Confusion Matrix : "+str(confusion_matrix(y_test,y_pred))+"\n\n")
def adc():
    global X_train, X_test, y_train, y_test
    adcx = AdaBoostClassifier(n_estimators=100, random_state=0,base_estimator=XGBClassifier(random_state=10))
    text.insert(END,"Score of ADC+Logit Algo: "+str(adcl.score(X_test, y_test))+"\n\n")
    text.insert(END,"Classification Report: "+str(classification_report(y_test, y_pred))+"\n\n")
    text.insert(END,"Confusion Matrix : "+str(confusion_matrix(y_test,y_pred))+"\n\n")
```

```
def svc():
    global X_train, X_test, y_train, y_test
    text.delete('1.0',END)
    svmg = SVC(gamma= 0.0000001, C=0.2,max_iter=100,probability=True)
    svmg.fit(X_train, y_train)
    text.insert(END,"Score of SVC Algo: "+str(svmg.score(X_test, y_test))+"\n\n")
    y_pred = svmg.predict(X_test)
    text.insert(END,"Classification Report: "+str(classification_report(y_test, y_pred))+"\n\n")
    text.insert(END,"Confusion Matrix : "+str(confusion_matrix(y_test,y_pred))+"\n\n")
def hgb():
    global X_train, X_test, y_train, y_test
    text.delete('1.0',END)
    hgb = HistGradientBoostingClassifier().fit(X_train, y_train)
    text.insert(END,"Score of HGB Algo: "+str(hgb.score(X_test, y_test))+"\n\n")
    y_pred = hgb.predict(X_test)
    text.insert(END,"Classification Report: "+str(classification_report(y_test, y_pred))+"\n\n")
    text.insert(END,"Confusion Matrix : "+str(confusion_matrix(y_test,y_pred))+"\n\n")
def stackclassify():
    global X_train, X_test, y_train, y_test
    text.delete('1.0',END)
    estimators = [('rf', RandomForestClassifier(n_estimators=10, random_state=42)),
            ('svr', make_pipeline(LinearSVC(random_state=42)))]
    sc = StackingClassifier(estimators=estimators, final_estimator=LogisticRegression())
    sc.fit(X_train, y_train).score(X_test, y_test)
    text.insert(END,"Score of stackclassify Algo: "+str(sc.score(X_test, y_test))+"\n\n")
    y_pred = sc.predict(X_test)
    text.insert(END,"Classification Report: "+str(classification_report(y_test, y_pred))+"\n\n")
    text.insert(END,"Confusion Matrix : "+str(confusion_matrix(y_test,y_pred))+"\n\n")
def mlp():
    global X_train, X_test, y_train, y_test
    text.delete('1.0',END)
    mlp = MLPClassifier(activation='tanh',solver='sgd',learning_rate='adaptive')
    mlp.fit(X_train,y_train)
    mlp.score(X_test,y_test)
    mlp = MLPClassifier(activation='logistic',solver='sgd',learning_rate='adaptive')
    mlp.fit(X_train,y_train)
    mlp.score(X_test,y_test)
    text.insert(END,"Score of MLP Algo: "+str(mlp.score(X_test, y_test))+"\n\n")
    y_pred = mlp.predict(X_test)
    text.insert(END,"Classification Report: "+str(classification_report(y_test, y_pred))+"\n\n")
    text.insert(END,"Confusion Matrix : "+str(confusion_matrix(y_test,y_pred))+"\n\n")
```

```
ann():
    global X_train, X_test, y_train, y_test
    text.delete('1.0',END)
runcnn = Button(main, text="Run RF Algorithm", command=rf)
runcnn.place(x=700,
runann = Button(main, text="Run SVC Algorithm", command=svc)
runann.place(x=700, y=500)
runann.config(font=font1)
runltsm = Button(main, text="Run ADC Algorithm", command=adc)
runltsm.place(x=700, y=550)
runltsm.config(font=font1)
runcnn = Button(main, text="Run XGB Algorithm", command=xgb)
runcnn.place(x=700, y=600)
runcnn.config(font=font1)
runmlp = Button(main, text="Run HGB Algorithm", command=hgb)
runmlp.place(x=700, y=650)
runmlp.config(font=font1)
runltsm = Button(main, text="Run MLP Algorithm", command=mlp)
runltsm.place(x=700, y=700)
runltsm.config(font=font1)
runcnn = Button(main, text="Run ANN Algorithm", command=ann)
runcnn.place(x=700, y=750)
runcnn.config(font=font1)
runmlp = Button(main, text="Run stackclassify Algorithm", command=stackclassify)
runmlp.place(x=700, y=800)
runmlp.config(font=font1)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=100)
text.config(font=font1)
main.config(bg='PeachPuff2')
main.mainloop()
```

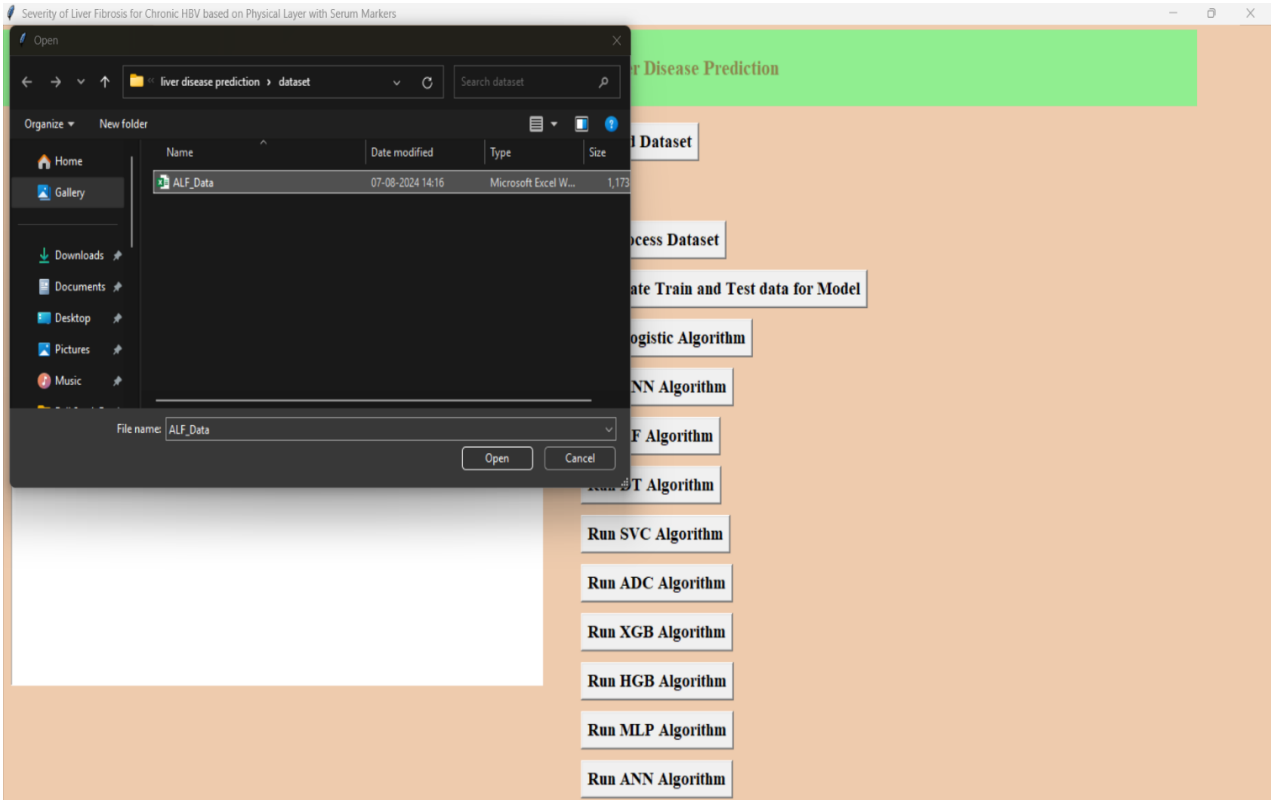
5. SCREENSHOTS

5.1 Comparative Analysis of Liver Disease Prediction using Machine Learning



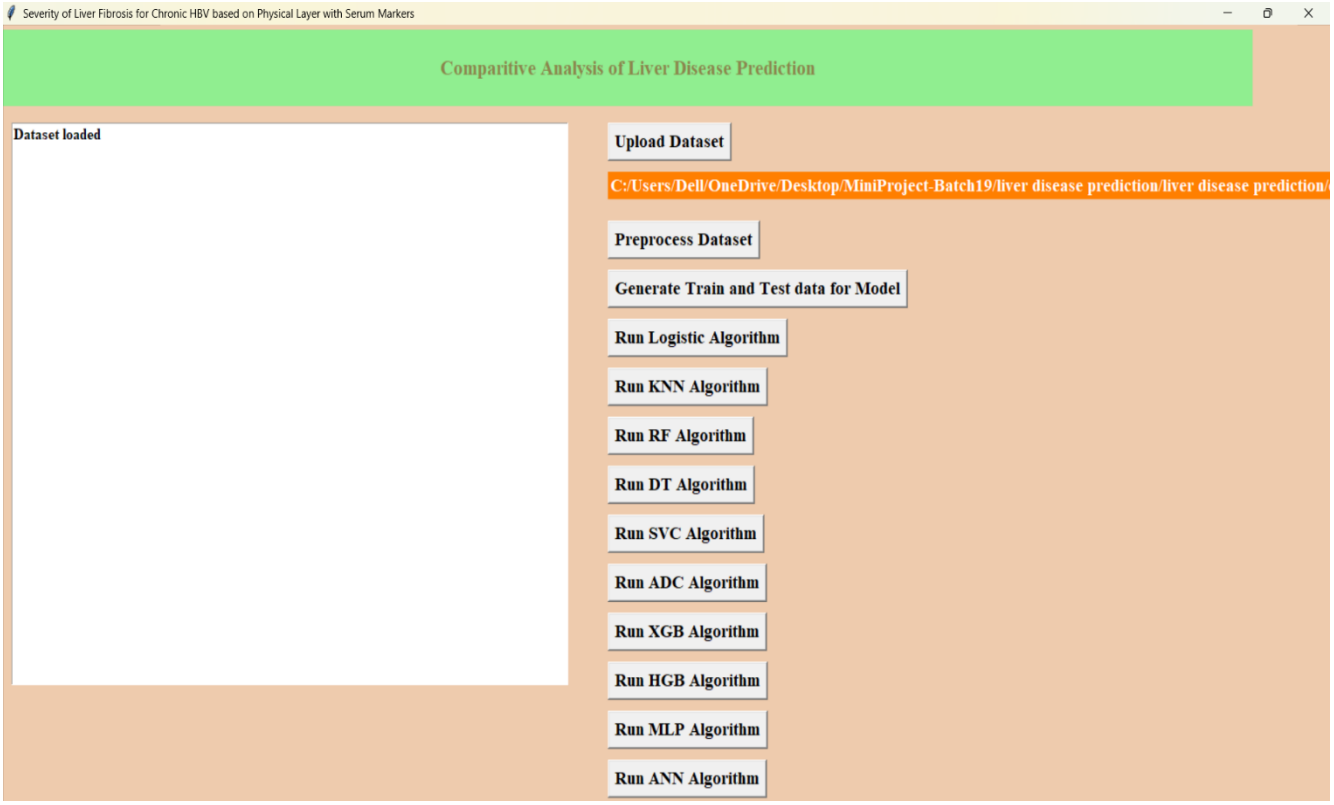
Screenshot 5.1: Comparative Analysis of Liver Disease Prediction

5.2 Uploading Dataset



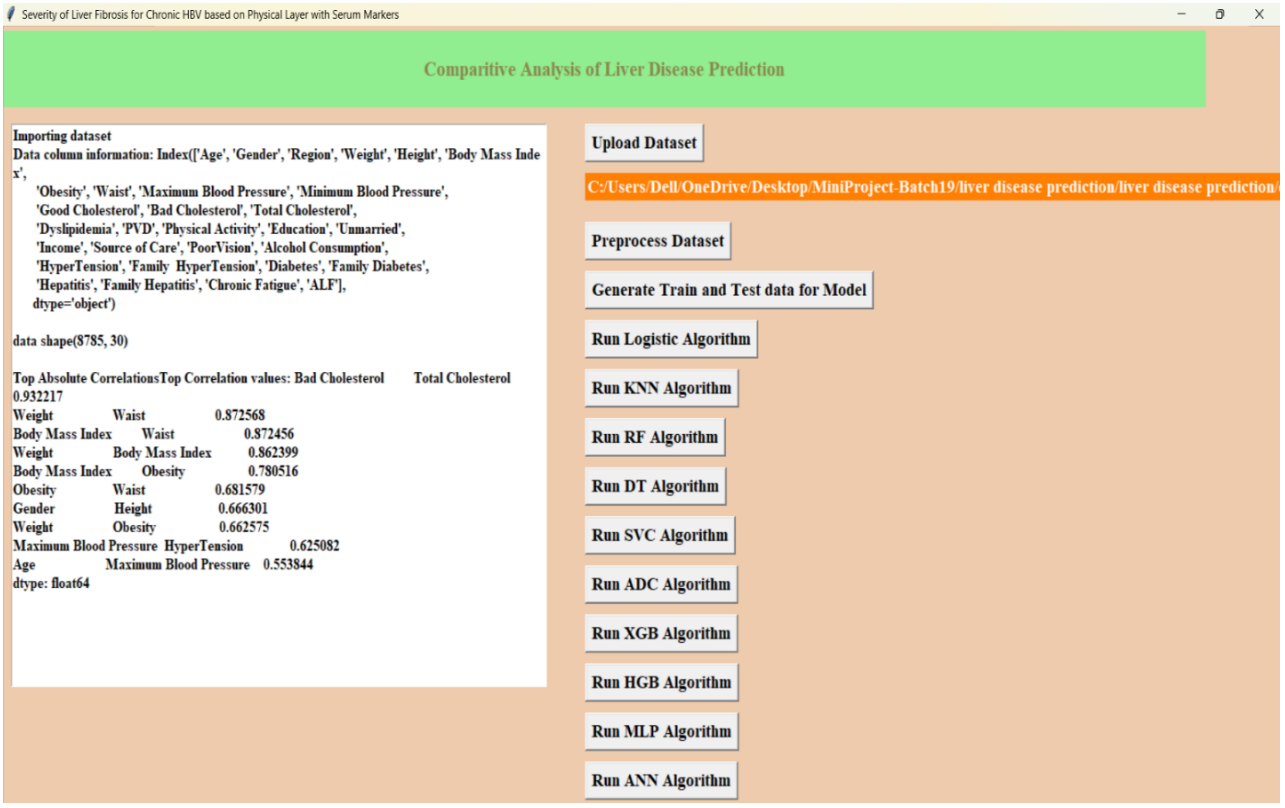
Screenshot 5.2: Uploading data from the local drive

5.3 Dataset Loaded



Screenshot 5.3: Data is uploaded from the local drive and file is displayed on Screen

5.4 Dataset Preprocessing

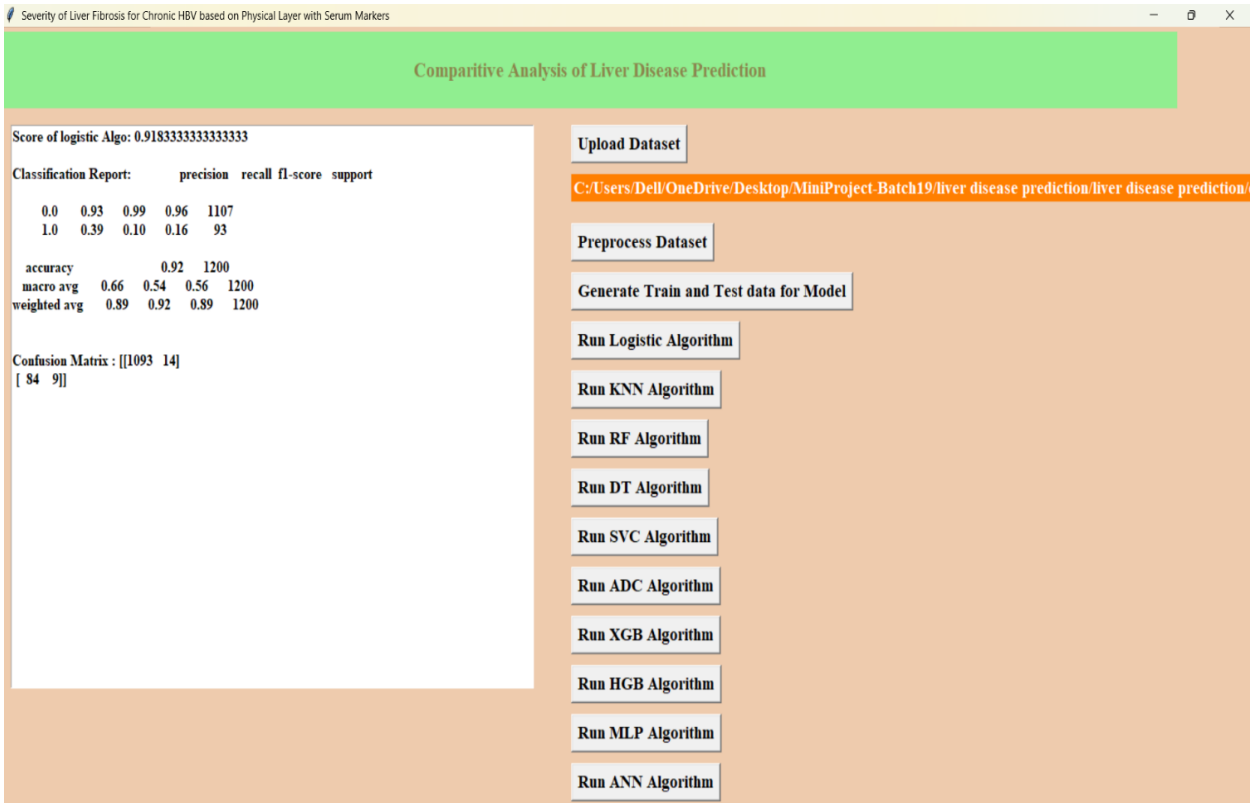


Screenshot 5.4: Dataset Preprocessing for Comparative Analysis of Liver Disease Prediction using Machine Learning

5.5 Split Train-Test Data

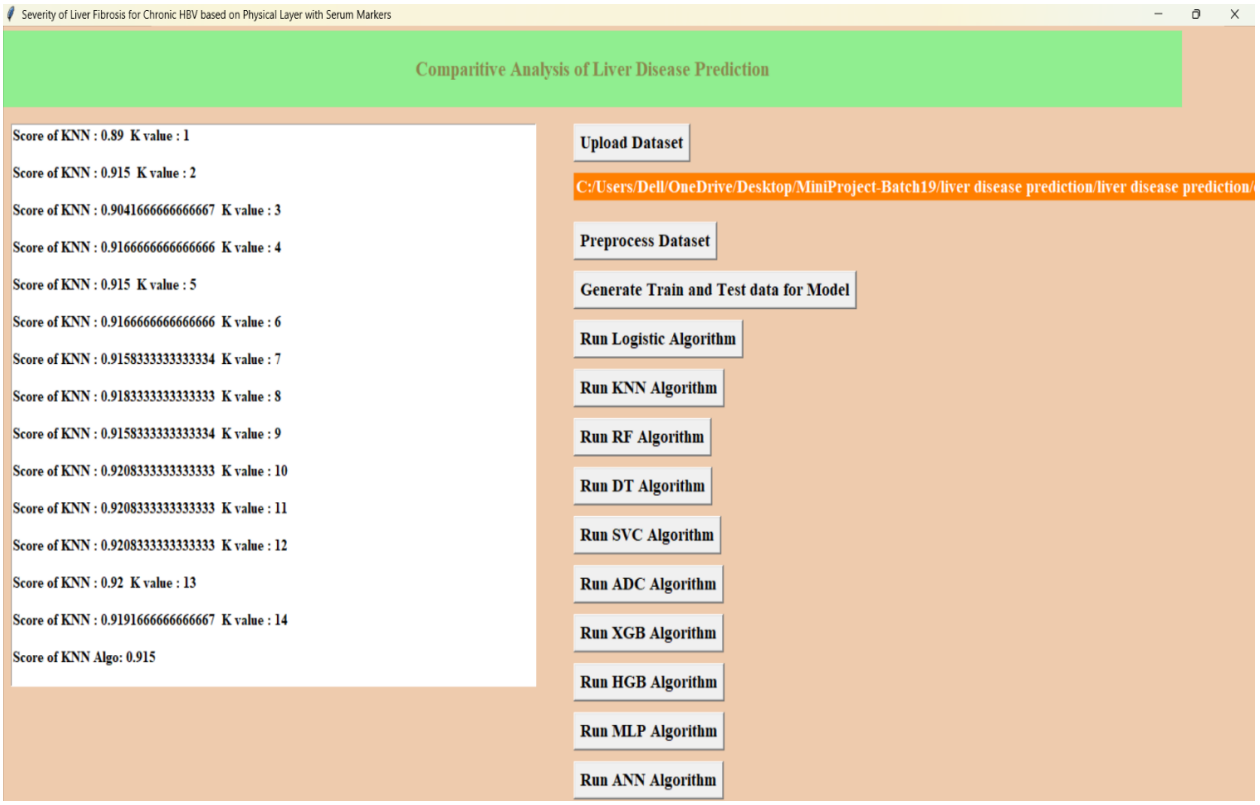
Screenshot 5.5: Split the Dataset for Trainning and Testing

5.6 Run Logistic Algorithm



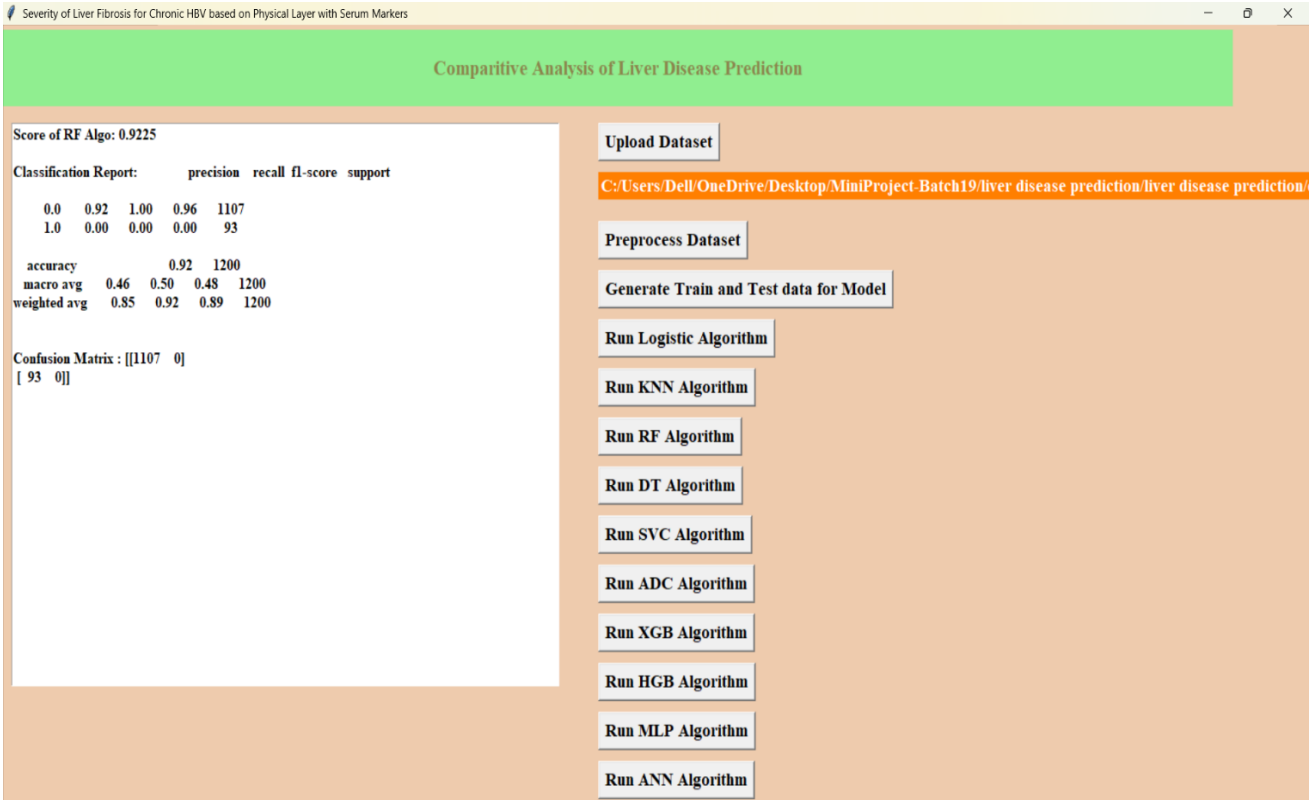
Screenshot 5.6: After Running Logistic Algorithm and Score of Logistic Algorithm is Displayed on the Screen

5.7 Run KNN Algorithm



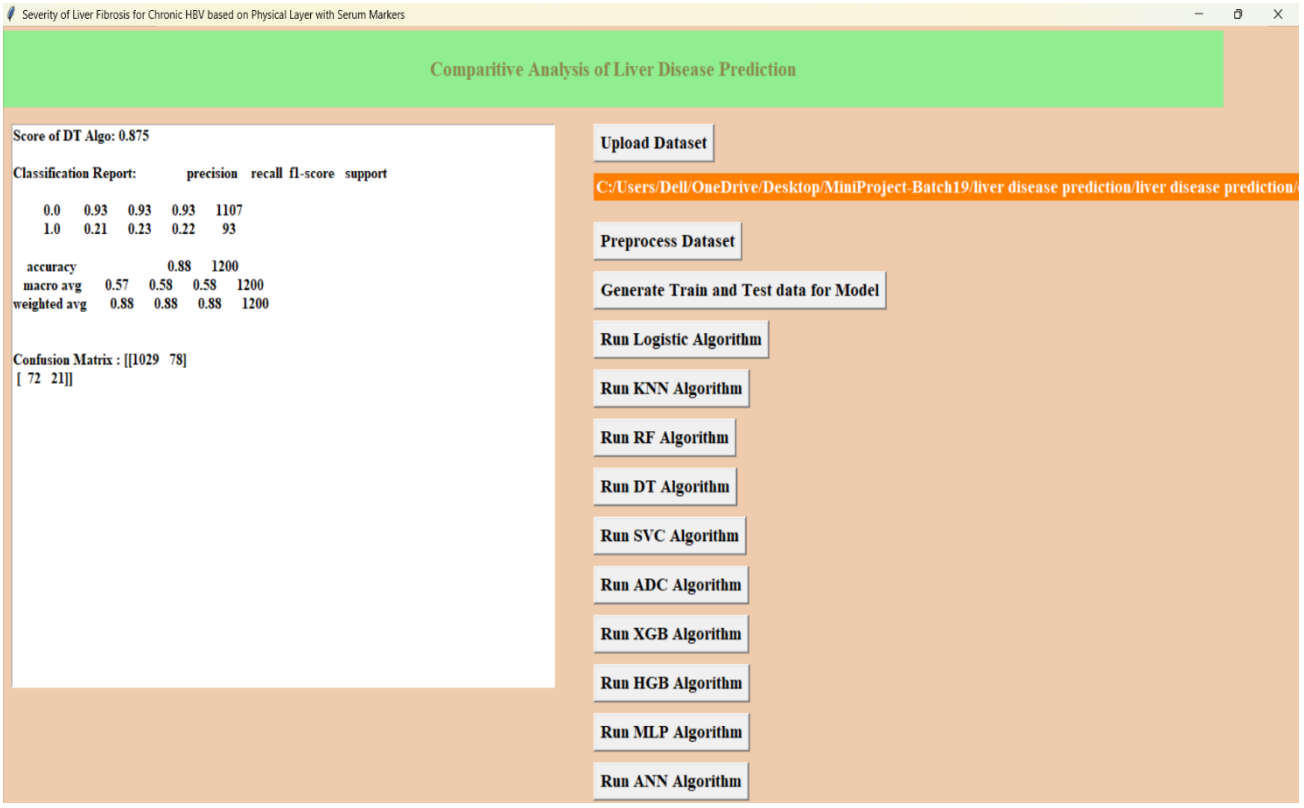
Screenshot 5.7: Score of KNN algorithm is displayed

5.8 Run RF Algorithm



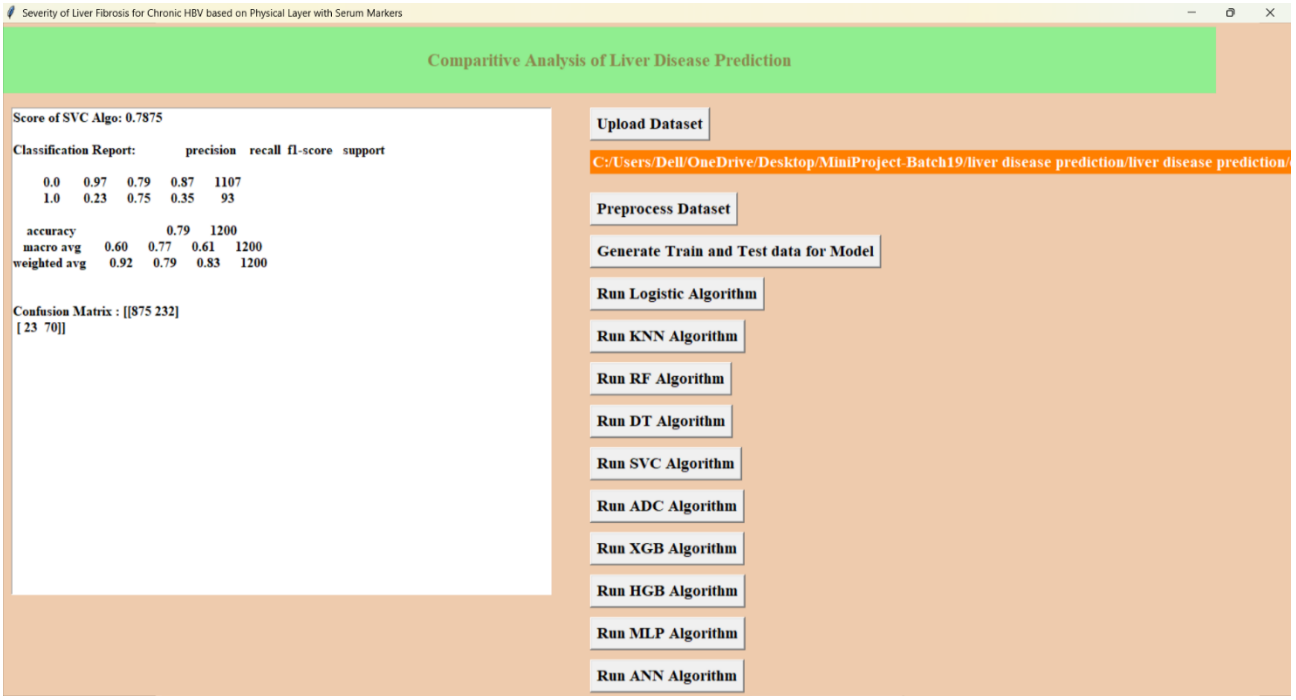
Screenshot 5.8: Score of RF Algorithm is displayed

5.9 Run DT Algorithm



Screenshot 5.9 : Score of DT Algorithm is displayed

5.10 Run SVC Algorithm

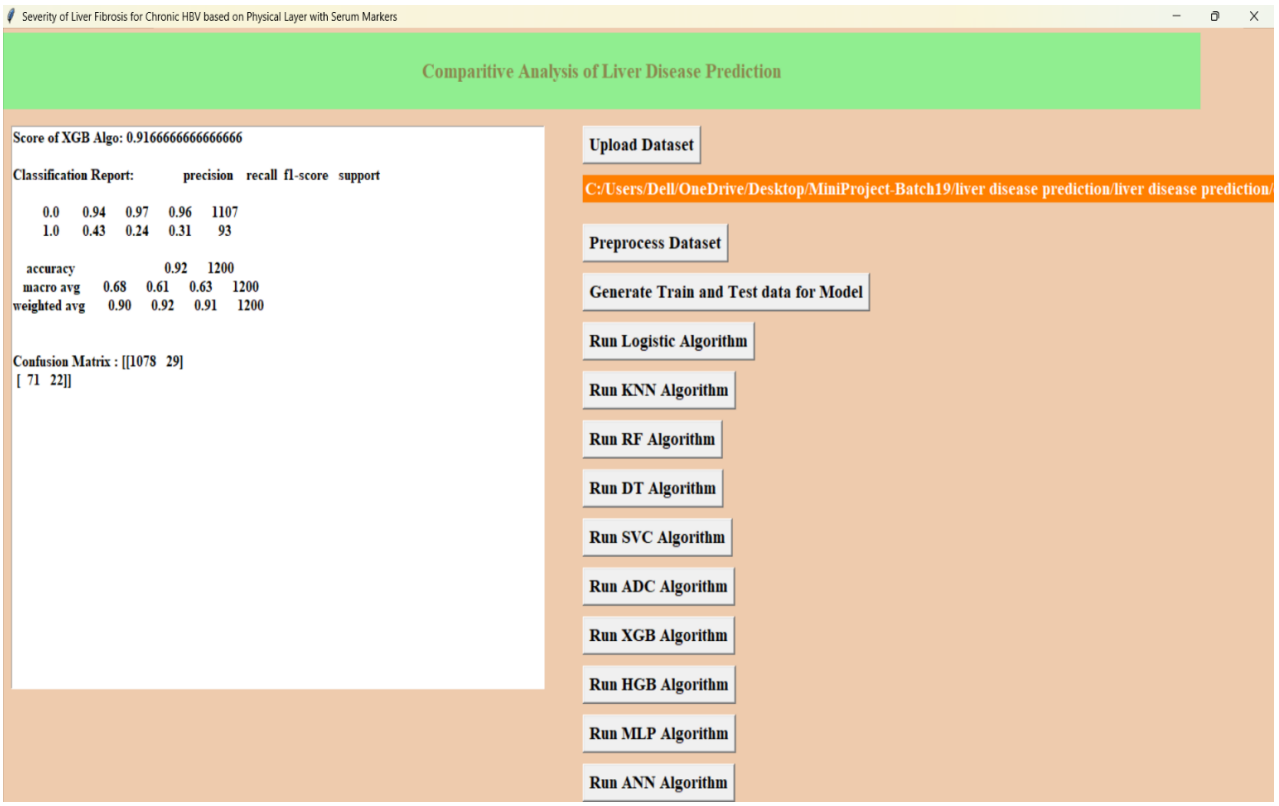


Screenshot 5.10: Score of SVC Algorithm is displayed

5.11 Run ADC Algorithm

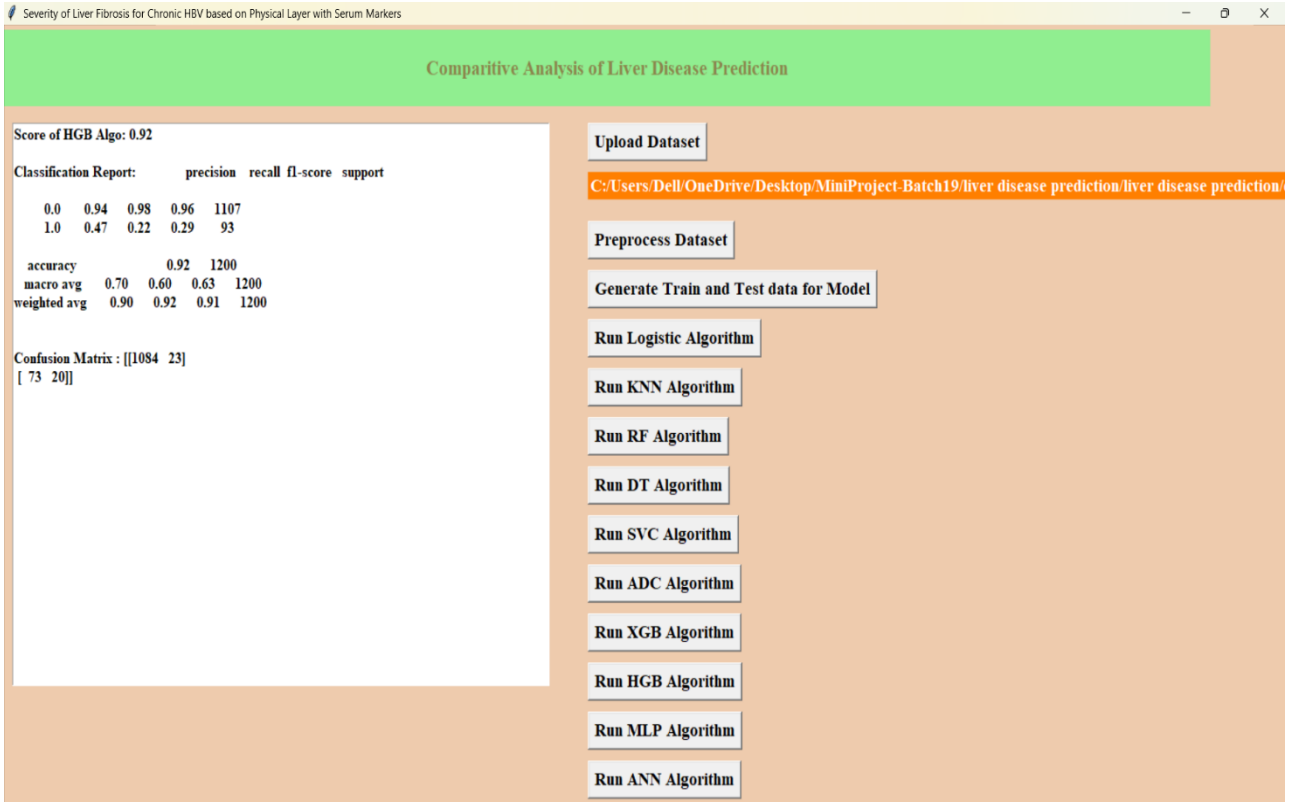
Screenshot 5.11 : Score of ADC Algorithm is displayed

5.12 Run XGB Algorithm



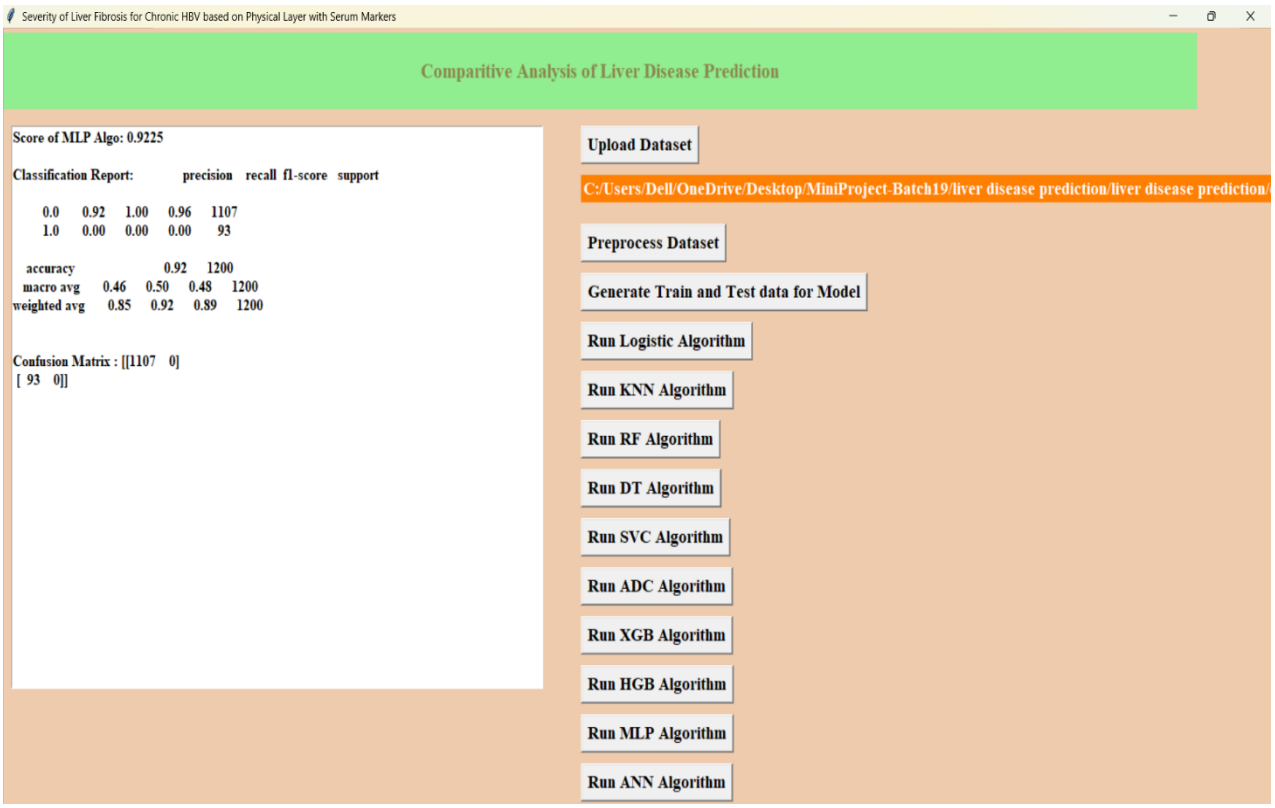
Screenshot 5.12: Score of XGB Algorithm is displayed

5.13 Run HGB Algorithm



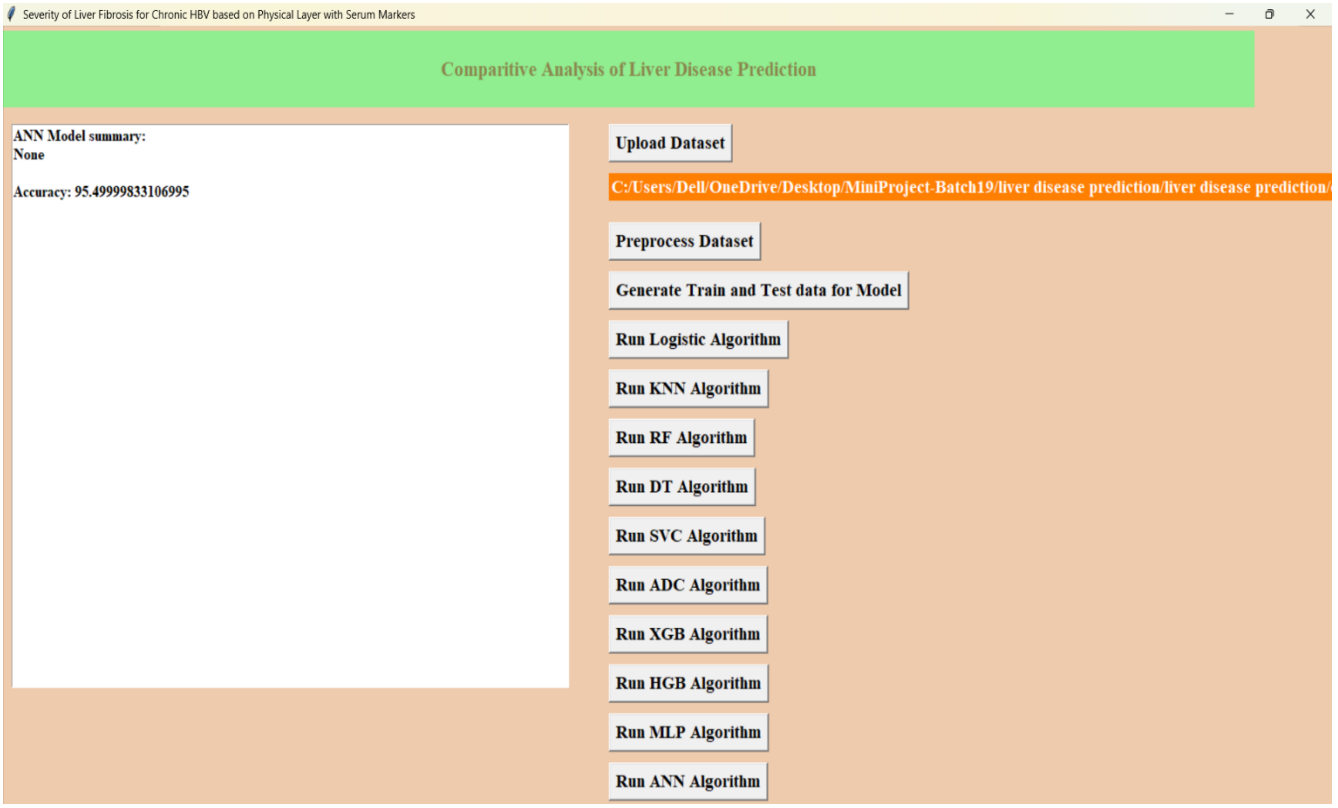
Screenshot 5.13: Score of HGB Algorithm is displayed

5.14 Run MLP Algorithm



Screenshot 5.14: Score of MLP Algorithm is displayed

5.15 Run ANN Algorithm



Screenshot 5.15: Score of ANN Algorithm is displayed

6. TESTING

6.TESTING

6.1 INTRODUCTION TO TESTING

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

6.2 TYPES OF TESTING

6.2.1 UNIT TESTING

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

6.2.2 INTEGRATION TESTING

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

6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as the specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

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

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

Organization and preparation of functional tests is focused on requirements, key functions, special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes.

6.2.4 SYSTEM TESTING

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

6.2.5 WHITE BOX TESTING

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

6.2.6 BLACK BOX TESTING

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

6.3 TEST CASES

6.3.1 UPLOADING DATA

Test case ID	Test case name	Purpose	Test Case	Output
1	Upload Dataset	Ensure dataset is uploaded correctly	Click the "Upload Dataset" button, select the dataset file.	File path displayed, "Dataset loaded" message in the text area.
2	Preprocess Dataset	Verify the preprocessing step works as expected	Click the "Preprocess Dataset" button after uploading the file.	Columns dropped, null values handled, correlation output shown.
3	Split Data	Validate dataset is split into train and test sets.	Click the"Generate Train and Test data" button.	Dataset split confirmation message displayed

6.3.2 OUTPUT PREDICTION

Test case ID	Test case name	Purpose	Input	Output
1	Logistic Regression Algorithm	Test if Logistic Regression runs without errors	Click "Run Logistic Algorithm" after dataset split.	Accuracy score, classification report, and confusion matrix displayed.
2	KNN Algorithm	Test KNN classification on dataset	Click "Run KNN Algorithm" after dataset split.	Accuracy score, classification report, and confusion matrix displayed for different K values.
3	Random Forest Algorithm	Ensure Random Forest classifier runs successfully	Click "Run RF Algorithm" after dataset split.	Gives the Ensemble algorithm graph
4	Decision Tree Algorithm	Validate Decision Tree algorithm accuracy	Click "Run DT Algorithm" after dataset split.	Accuracy score, classification report, and confusion matrix displayed.
5	Support Vector Classifier (SVC)	Ensure SVC model runs properly	Click "Run SVC Algorithm" after dataset split.	Accuracy score, classification report, and confusion matrix displayed.
6	AdaBoost Classifier (ADC)	Test AdaBoost performance	Click "Run ADC Algorithm" after dataset split.	Accuracy score, classification report, and confusion matrix displayed for different base estimators.
7	XGBoost Classifier	Test XGBoost algorithm performance	Click "Run XGB Algorithm" after dataset split.	Accuracy score, classification report, and confusion matrix displayed.
8	HistGradientBoosting (HGB)	Validate HistGradientBoosting algorithm	Click "Run HGB Algorithm" after dataset split.	Accuracy score, classification report, and confusion matrix displayed.
9	Multi-layer Perceptron (MLP)	Ensure MLP neural network runs as expected	Click "Run MLP Algorithm" after dataset split.	Accuracy score, classification report, and confusion matrix displayed.
10	ANN Algorithm	Validate ANN model is trained and loaded correctly	Click "Run ANN Algorithm" after dataset split.	Model loaded/trained, accuracy displayed, model summary shown.

7. CONCLUSION

7. CONCLUSION & FUTURE SCOPE

7.1 PROJECT CONCLUSION

In conclusion, the proposed machine learning-based system represents a significant advancement in the diagnosis of liver diseases, offering a transformative approach to addressing the limitations of current diagnostic methods. By integrating diverse data sources and employing advanced algorithms, the system enhances diagnostic accuracy, facilitates early detection, and enables personalized treatment strategies. Its non-invasive nature and real-time decision support capabilities streamline the diagnostic process, reducing reliance on traditional, often invasive procedures and potentially lowering overall costs. Additionally, the system's ability to continuously learn and adapt from new data ensures that it remains at the forefront of diagnostic technology, improving over time. Overall, this innovative approach not only promises to improve clinical outcomes and patient engagement but also sets the stage for future advancements in liver disease diagnostics, contributing to more effective and timely management of these critical health conditions.

7.2 FUTURE SCOPE

The future scope of the machine learning-based liver disease diagnosis system holds great potential. It could evolve by integrating genomic and proteomic data for more personalized diagnostics, developing more advanced algorithms for greater accuracy in predicting complex liver conditions, and validating the system in real-world clinical settings. Collaboration with healthcare providers will be essential to refine the system, address ethical concerns, and ensure seamless integration into medical workflows. As computational power and data acquisition methods improve, the system will become more efficient, contributing to better liver disease management and patient outcomes.

8. BIBLIOGRAPHY

8. BIBLIOGRAPHY

8.1 REFERENCES

[1] Wang, X., Yang, L., & Zhang, L. (2020). Deep learning for liver disease diagnosis using medical imaging: A review. Journal of Medical Imaging and Health Informatics, 10(1), 119-129.

[2] Bai, H., Wang, W., & Yang, H. (2021). Machine learning models for early detection of liver cancer: A review. Computational Biology and Chemistry, 90, 107425.

[3] Kim, S., & Lee, J. H. (2020). A survey of machine learning techniques for liver disease diagnosis. International Journal of Computer Applications, 975, 35-44.

[4] Li, X., Xu, Y., & Yang, Z. (2019). Predictive models for liver fibrosis using machine learning algorithms. Artificial Intelligence in Medicine, 98, 76-85.

[5] Zhu, J., & Zhang, L. (2021). Leveraging deep learning for liver cancer classification: A comparative study. Bioinformatics, 37(10), 1464-1471

8.2 WEBSITES

https://github.com/Chinthakindi-Rushmitha/Comparative-analysis-of-liver-disease-predictionr