Classifier Algorithm Analysis for Liver Disease Prediction

REVIEW 3

for

DATA MINING TECHNIQUES (ITE2006)

in

B.Tech – Information Technology and Engineering

By

Ishan Rajesh Kasat [20BIT0391]

Dhruv Umesh Sompura [20BIT0357]

Chavan Mukul Manish[20BIT0238]

Under the Guidance of

VALARMATHI B

Associate Professor Sr., SITE

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology and Engineering

Fall Semester 2022 - 2023

DECLARATION BY THE CANDIDATE

We here by declare that the project report entitled **"Classifier Algorithm Analysis for Liver Disease Prediction"** submitted by us to Vellore Institute of Technology University, Vellore in partial fulfillment of the requirement for the award of the course **Data Mining Techniques (ITE2006)** is a record of bonafide project work carried out by us under the guidance of **Prof. B.Valarmathi.** We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other course.

Place : Vellore

Signature

Date : 14th November 2022

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology & Engineering [SITE]

CERTIFICATE

This is to certify that the project report entitled **“Classifier Algorithm Analysis for Liver Disease Prediction”** submitted by **Ishan Rajesh Kasat [20BIT0391], Dhruv Umesh Sompura [20BIT0357], Chavan Mukul Manish[20BIT0238]** to Vellore Institute of Technology University, Vellore in partial fulfillment of the requirement for the award of the course **Data Mining Techniques (ITE2006)** is a record of bonafide work carried out by them under my guidance.

Prof. B.Valarmathi

GUIDE

Associate Professor (Senior), SITE

Classifier Algorithm Analysis for Liver Disease Prediction

Ishan Rajesh Kasat [1], *Dhruv Umesh Sompura* [2], *Chavan Mukul Manish* [3]

[1,2,3] **Department of Information Technology, VIT University, Vellore, Tamil Nadu, India**

Abstract

Machine learning can be used in automated diagnosis of various diseases like liver diseases. India has more than a million people getting diagnosed with liver diseases each year hence it's important to detect them at an early stage. These are caused due to consumption of alcohol, contaminated food, obesity, thus we need a system that can predict the symptoms of liver diseases. We can predict these diseases using patient data and machine learning algorithms. Performance of this system can be measured in terms of accuracy, recall f-measure, etc. In this project we have tried to use machine learning and data mining techniques to help this noble cause of detecting liver diseases at an early stage.

In the existing method[ref. 18] the researchers have used MLCNN-LDPS which provided an accuracy of 90.75%. We have used three hybrid algorithms: CNN+LSTM(99.54%), CNN+GRU(98.21%), CNN+RNN(99.45%) and have achieved an accuracy as high as 99.48% using filters like upscaling and PCA. We also got used various algorithms and got the following accuracies- naive bayes: 76%, random forest: 80.26%, logistic: 72%, SVM: 76.93%, KNN: 76.67%.

Keywords – Liver Disease, SVM , Machine learning, Naïve Bayes, CNN, LSTM, GRU, RNN

I. Introduction

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate in predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values. Machine Learning is simply recognizing patterns in your data to be able to make improvements and intelligent decisions on its own. Python is the most suitable programming language for this because it is easy to understand and you can read it for yourself.

We all know that the liver is the largest internal organ of the body, which performs very important bodily functions including the formation of blood clotting factors and proteins, the production of triglycerides and cholesterol, the synthesis of glycogen and the production of bile. Usually, more than 75% of the liver tissue needs to be affected by the decline in function. It is therefore important to detect the disease at an early stage so that the disease can be treated before it becomes serious. Machine learning techniques have become very important in healthcare nowadays for disease prediction from medical database. Many researchers and companies are using machine learning to improve medical diagnostics. Among various machine learning techniques, classification algorithms are widely used in disease prediction. Some of the popular algorithms include KNN, SVM, RF, NBC etc. We have studied the same and made some points on it.

II. BACKGROUND:

The main objective is to analyse the parameters of various classification algorithms and compare their predictive accuracies so as to find out the best classifier for determining the liver disease. Also, the other objectives could be listed as -

1) Obtaining a suitable dataset to implement the prediction model

The first and foremost objective of implementing a machine learning model is to find a suitable dataset which contains liver disease patients records which will include the values of several attributes.

2) Training the model to obtain a very high accuracy

Training the model plays an important role, the more data collected, the more the model is trained. The process of training an ML model involves providing an ML algorithm (that is, the learning algorithm) with the training dataset. Obtaining high accuracy is important.

3) Implementing performance metrics on the model

Performance metrics play an important role in machine learning models and only with these performance measures one can identify how good the model is and how effectively it is performing.

4) Develop a prototype web-based interface for prediction

The development of a prototype web-based interface is necessary and it is made available to all users. The user is required to have a necessary blood test report by which the user can enter the values and thus the system predicts the results based on the values.

5) Encouraging diagnostic centres to implement high accuracy models

The ultimate objective of this whole project is to encourage diagnostics to implement machine learning models for fast and accurate diagnostics of liver disease.

III.. Literature survey

Table 1: The following table suggest the diverse ways used to predict liver disease using machine learning and other various techniques.

S.No.	Title of the Paper And year	Algorithms used	Data set being used	Performance measures	Scope for future work
1.	A Comparative Study on Liver Disease Prediction Using Supervised Machine Learning Algorithms (2019)**[1]**	LR, RF, DT, SVM, KNN and NB	Data set from the UCI Machine Learning Repository	Accuracy, LR – 75%, RF – 74 %, DT – 69 % SVM – 64% KNN – 62 % NB – 53 %	More algorithms can be picked to assemble an increasingly precise model of liver disease prediction and performance can be progressively improved.
2.	Liver Patient Classification using Logistic Regression (2018) **[2]**	Logistic Regression	Dataset from UCI machine learning repository, published as "Indian Liver Patient Records".	Accuracy – 74%	To use simpler ML techniques to achieve better results than advanced and complex techniques.
3.	Liver Disease Prediction using SVM and Naïve Bayes Algorithms (2015) **[3]**	SVM and Naïve Bayes Algorithms	Indian Liver Patient Dataset (ILPD) from the UCI Repository	Accuracy – Naïve Bayes – 55.8% SVM – 76.6%	To evaluate the classification algorithms on the basis of highest classification accuracy and minimum execution time.
4.	Liver Disease Prediction By Using Different Decision Tree Techniques (2018) **[4]**	Decision Tree - J48, LMT, Random Forest, Random tree, REPTree, Decision Stump, and Hoeffding Tree	Data set from the UCI Machine Learning Repository	Accuracy – J48 – 65.69% LMT-69.47% Random Forest-69.30% Random tree-66.55% REPTree-66.13% Decision Stump-70.67% Hoeffding Tree-69.75%	To collect the very recent data from various regions across the world for liver disease diagnosis. The results of which will encourage us to continue developing other advanced decision trees such as CART.

5.	Software-based Prediction of Liver Disease with Feature Selection and Classification Techniques (2020) **[5]**	Logistic Regression, Naive Bayes, SMO, IBk, J48 and Random Forest	Indian Liver Patient Dataset (ILPD) from the UCI Repository	Accuracy – Logistic Regression-74.36% Naive Bayes-55.9% SMO-71.36% IBk-67.41% J48-70.67% Random Forest-71.87%	To use feature selection techniques for prediction of liver disease.
6.	Performance Analysis of Liver Disease Prediction Using Machine Learning Algorithms (2018) **[6]**	J48, MLP, SVM, Random Forest, Bayesian Network	Indian Liver Patient Dataset (ILPD) from the UCI Repository	Accuracy – J48-95.04% MLP-77.54% SVM-73.44% Random Forest-80.22% Bayesian Network-90.33%	The method requires further improvement mostly regarding feature selection of the liver into multiple components and also it could be employed for detecting the heart diseases in future with the heart dataset and classification of the diseases.
7.	Rule Optimization of Boosted C5.0 Classification Using Genetic Algorithm for Liver disease Prediction**[7]**	Genetic Algorithm, Boosted C5.0	Dataset from UCI database which is about the liver patient in one of the Indian hospitals in 2012	Accuracy – Genetic Algorithm-92.93% Boosted C5.0-81.87%	To optimize the rules of boosted C5.0 algorithm and reduce their number and find the effective attributes in the liver disease diagnosis.
8	Decision Factors on Effective Liver Patient Data Prediction (2014) **[8]**	Naïve Bayes, Decision Tree, Multi-Layer Perceptron, KNN, Random forest and Logistic	Data set from the UCI Machine Learning Repository	Accuracy – Logistic – 91.3%	More effective performance criteria should be accompanied with the choice of more appropriate algorithms.
9	Prediction and Analysis of Liver Disorder Diseases by using Data Mining Technique: Survey (2018) **[9]**	SVM, NBC and C4.5	Liver disorder disease dataset	Accuracy – 87.12%	Use of Hybrid approach to get better performance accuracy for liver disorder diseases prediction with

					their suitable data sets.
10	Prediction of Fatty Liver Disease using Machine Learning Algorithms (2018) **[10]**	RF, NB, ANN, and LR	Data set from the UCI Machine Learning Repository	Accuracy – RF-87.48% NB-82.65% ANN-81.85% LR-76.96%	Future studies are needed to validate the proposed model to predict FLD in various types of datasets.
11	EFFICIENT PREDICTION OF LIVER DISEASE USING SELECTED ATTRIBUTES, 2017**[11]**	Bayesian Logistic Regression, Multilayer Perceptron, SVM, Attribute Selected Classifier, Classification via Regression, NBTree, J48 and Random forest	whole dataset of Indian liver disease patients same as available on kaggle (UCI Machine Learning Repository)	Bayesian Logistic Regression-71.3551% Multilayer Perceptron 68.7822% SVM 71.3551% Attribute Selected Classifier 67.4099% Classification Via Regression 70.4974% NBTree 67.2384% J48 68.4391% Random Forest 69.9828%	For future study, the CT scans data of the abdomen can be used to achieve more accurate results in the prediction of liver disease at an early stage. For this purpose, we will use a pixel segmentation technique for real time images of liver disease patients. Firstly, we will find LFT dataset results and then compare and match with the results of CT scans images data. Through this approach there will be more chances to achieve high accuracy for prediction
12	LIVER DISEASE DIAGNOSIS USING MACHINE LEARNING, 2017**[12]**	Logistic Regressions, SMV, KNN, Random forest, Decision tree	The Indian Liver Patient Dataset (ILPD), which was chosen from the UCI Machine Learning repository	SMV-75.54% SMV and ANN-98.83% PNN- 95%	
13	A Comparison based Liver Disease Prediction Using	support vector classifier, k-nearest neighbor,	liver dataset based on data	SMV-71.72%	The experimental results show that decision trees and

	Machine Learning Techniques, 2022**[13]**	decision tree classifier, random forrest classifier	from the UCI repository	KNN- 64.14% Decision Tree- 100% Random Forest Classifier- 100%	random forest classifiers are better at predicting liver illness.
14	Performance Evolution of Different Machine Learning Algorithms for Prediction of Liver Disease, 2019**[14]**	Supervised Learning Algorithms, K-Nearest Neighbor Algorithm(KNN), SVM, Logistic Regression, Navi Bayes, Random Forest	Liver data set from kaggle	KNN- 55% SVM: 53% Logistic Regression- 48% Navi Bayes- 70% Random Forest- 50%	Re-enactment results demonstrated that Logistic Regression classifier demonstrated its exhibition in foreseeing with best outcomes regarding precision and least execution time
15	NeuroSVM: A Graphical User Interface for Identification of Liver Patients, 2015**[15]**	Naïve Bayes, Bagging,Random forest and SVM	The Indian Liver Patient Dataset (ILPD) was selected from UCI Machine learning repository for this study	Naïve Bayes 53.09 Bagging 66.73 Random Forest 67.67 SVM 76.22 NeuroSVM 98.83	A hybrid NeuroSVM model was developed to classify liver patients based on their biological parameters using an artificial neural network. The hybrid model is deployed as a graphical user interface (GUI) in R. The GUI can be used by clinicians as a screening tool to predict liver disease in patients in the future
16	Prediction of Liver Diseases by Using Few Machine Learning Based Approaches, 2018**[16]**	Random Forest, Perceptron, Decision Tree, K-Nearest Neighbors (KNN) and Support Vector Machine (SVM)	ILPD (Indian Liver Patient Dataset)	SVM: 71.35%, Random forest: 71.86%, KNN: 74.15%,	Classification rules and disease identifying techniques may also be generated by using different efficient algorithms. More than one database for comparative

					analysis may also be used. Their works has certain limitations as the model has underperformed having less accuracy than expectations. So, in future, inclusion of deep learning methods may improve their results further.
17	Efficiency measure of Machine Learning Algorithms on Liver Disease Diagnosis, 2020**[17]**	The selected Classification Algorithms that are Naive Bayes, Decision Tree, Random Forest, Support Vector Machines and Multi-Layer Perceptron	Three datasets are named Visakhapatnam dataset, Vijayawada dataset and Tirupathi dataset based on geographical region	Naïve Bayes-93.41% Decision Tree- 100% Random Forest- 100% SVM-83.12% MLP-96.707%	ANOVA and MANOVA analysis is also suggested for the various confidence levels like 99 % and 90 %. This statistical analysis may be applied for various regions of India i.e different states of India to investigate the geographical effect and to suggest the localized settings for the diagnosis of liver disease
18	Accurate liver disease prediction system using convolutional neural network, 2021**[18]**	Score based Artificial Fish Swarm Algorithm (SAFSA), MCNN	Indian Liver Patient Dataset from UCI	MLPNN-86.70% MLCNN-LDPS-90.75%	the performance of NB Tree algorithm will be the target of improvement of the accuracy by finding the most significant factor in identifying liver disease patients
19	Machine Learning Techniques in Analysis and Prediction of Liver Disease, 2021 **[19]**	Logistic Regression, KNN, Decision Tree, Random Forest, SVM Classifier	Indian Liver Disease Patients (ILDP) available in UCI repository	Logistic Regression-71%, KNN-81%, Decision Tree-86%, Random Forest-90%,	In future an improved C4.5 could be derived with various parameters. This

				SVM Classifier-93%	paper gives generalization of various data mining techniques to diagnose liver disease at an earlier stage.
20	Statistical Machine Learning Approaches to Liver Disease Prediction, 2021**[20]**	ANN, RF, SVM	UCI Machine Learning Repository (UCI-MLR)	ANN-88.89%, RF-98.14%, SVM-96.75%	In the future, the local interpretable model-agnostic explanation (LIME) method will be used to understand the model's interpretability. Instead of binary classification, one may use multinomial classification by separating the types of liver disease. In this way, each model's performance can be compared.
21	A Comparative Analysis of the Ensemble Method for Liver Disease Prediction,2019**[21]**	AdaBoost, LogitBoost, BeggRep, BeggJ48 and random Forest	ILPD (Indian Liver Patient Dataset)	LogitBoost - 71.53% AdaBoost - 69.98% BeggJ48 - 69.93% BeggRep - 70.15% RF - 69.12%	obtain the most current data from different regions worldwide for the treatment of liver disorders and the prevention of liver disease.
22	A Data Mining Approach to Prediction of Liver Diseases,2020**[22]**	Support Vector Machine, Naive Bayes and Decision Tree	University of California Irvine (UCI)	Bayes Point Machines - 70.52% Neural Networks 66.85%	to get better accuracy of results
23	Application of Data Mining Techniques to Explore Predictors of HCC in Egyptian Patients with HCV-related Chronic Liver Disease, 2015**[23]**	Decision tree algorithm	Endemic Medicine Department, Cairo University Hospital.	a sensitivity of 96% and specificity of 82%	Test on different range of dataset

24	Prognosis of Liver Disease: Using Machine Learning Algorithms,2018**[24]**	SVM, Logistic Regression and Decision tree	Laboratory report of 584 patients	Logistic Regression - 95.8% SVM - 82.7% decision tree - 94.9%	considering the tumor characteristics of the patient once he is diagnosed with liver disorder
25	Review on Effective Disease Prediction through Data Mining Techniques, 2021**[25]**	SVM,KNN, SMO and Random Forest	UCI repository	94%	big data using spark and deep learning approaches inspired by recent work
26	Analysis of Data Mining Techniques For Healthcare Decision Support System Using Liver Disorder Dataset, 2016**[26]**	Decision trees J48, Naive Bayes, Multilayer perceptron, VFI algorithm	Liver Function Test (LFT) dataset	J48 – 68.97% ZeroR – 57.97% Multilayer Perceptron – 71.59% Naive Bayes – 55.36 VFI – 60.28	more experiments with different datasets are required to support the findings
27	An Approach of Data Mining for Predicting the Chances of Liver Disease in Ectopic Pregnant Groups, 2012**[27]**	REGRESSION ANALISIS	collection of data in ectopic	standard error is 0.577 and the proportion is 8.7487. Significance test gives the value of 1.7419	to identify the factors affecting the increased cases in recent time with the help of theory of evidence and fuzzy logic.
28	Analysis of classification algorithms for liver disease diagnosis, 2017**[28]**	K star, NBC, Bagging, Logistic and Rep Tree	AP and UCLA	Bagging 84% K-Star 98.5% Navie Bayes 35% Logistic 75.6% REPTree 80.4%	Larger datasets
29	A Critical Study of Selected Classification Algorithms for Liver Disease Diagnosis, 2011**[29]**	Naive Bayes Algorithm, C4.5 Algorithm, Back propagation Algorithm, K-Nearest Neighbor Algorithm, Support Vector Machines (SVM) Algorithm	University of California at Irvine (UCI) Machine Learning Repository	NBC 56.52% C 4.5 68.69% Back Propagation 71.59% K-NN 62.89% SVM 58.26%	KNN, Back propagation and SVM are giving better results with all the feature set combinations
30	Prediction of Liver Diseases Based on Machine Learning	Naive Bayes classifier C5.0 classifier	Egyptian Liver Research Institute and the Mansoura	Proposed model - 97.20%	the use of fast datasets technique like Apache Hadoop or Spark

	Technique for Big Data, 2018**[30]**	Support Vector Machine	Central Hospital, Dakahlia Governorate, Egypt.	SVM – 96.5% C5.0 – 93.7% Naïve Bayes – 95.2%	

Table 1

1] This paper mainly focused on the use of clinical data for liver disease prediction and explore different ways of representing such data through analysis of different ML algorithms. In this study, the main aspect is to predict the results more efficiently and reduce the cost of diagnosis in the medical sector. Therefore, we used different classification techniques for the classification of patients who have liver disease or not. Six machine learning techniques have been applied including LR, KNN, DT, SVM, NB, RF and the performance of these techniques were estimated on various perspectives such as accuracy, precision, recall, f-1 score. Moreover, the performance was compared using the receiver operating characteristic (ROC). The goal of this work was to evaluate the performance of different Machine Learning algorithms in order to reduce the high cost of chronic liver disease diagnosis by prediction. It focuses to build an application which will have the option to predict liver infection prior and advise the wellbeing condition.

2] This paper mainly focussed on applying machine learning approach to classify liver patient that is Liver Patient or Not Liver Patient using patient gender and laboratory medical test data. The motivation behind this work was to apply simple and less computational classification technique like Logistic Regression and compare its results with earlier results obtained on the same dataset by other researchers. The classification results of Logistic regression have proved its significance on this dataset by achieving better classification accuracy than NBC (Naïve Bayes Classifier), C4.5 (Decision Tree), SVM (Support Vector Machine), ANN (Artificial Neural Network), and KNN (K Nearest Neighbors). The scope is to use simpler ML techniques to achieve b3etter results than advanced and complex techniques.

3] This paper proposes the research work of Naïve Bayes and Support Vector Machine (SVM) classifier algorithms used for liver disease prediction and analysis. Some of the findings from this paper are that the mechanisms that are currently used in the prediction of liver disease are prone to have different levels of accuracy and effectiveness. The sense of importance, though, is determined by the need of the hour. Different diseases demand accuracy of a different set of parameters and might not demand the same set of inferences, throughout more than a single case. This research paper concludes the use of Naive Bayes and Support Vector Machine algorithms for the prediction of liver diseases. There are two major parameters that are involved in understanding the suitability of the respective methodologies and they are - the time taken to execute the prediction process and the accuracy of the predictive result. It is clear through various studies and experimentations that the SVM classifier is the best of all the algorithms owing to the extremely high accuracy rates. But when it comes to the time taken to execute the predictive process, the Naive Bayes classifier reflects higher suitability since it takes the least possible time to execute the process.

4] This research paper explores the early prediction of liver disease using various decision tree techniques. Decision Tree algorithms have been successfully applied in various fields especially in medical science. The liver disease dataset which is selected for this study consist of attributes like total bilirubin, direct bilirubin, age, gender, total proteins, albumin and globulin ratio. The purpose of this study is to compare the decision tree algorithms such as J48, LMT, Random Tree, Random Forest, REPTree, Decision Stump and Hoeffding Tree in diagnosis liver disease. The liver dataset is analyzed using above decision tree algorithms and compare their performance with respect to seven performance metrics (ACC%, MAE, PRE, REC, FME, Kappa Statistics and runtime). The analysis proves that Decision Stump provides the highest accuracy than other techniques.

5] This research paper mainly focuses on prediction of liver disease based on a software engineering approach using classification and feature selection technique. The different attributes like age, direct bilirubin, gender, total bilirubin, Alkphos, sgpt, albumin, globulin ratio and sgot etc, of the liver patient dataset, are used to predict the liver diseases risk level. In this paper, six classification algorithms J-48, Random Forest, Logistic Regression, SMO (Support Vector Machine), IBk (k nearest Neighbor), Logistic Regression, Naive Bayes have been considered for implementation and comparing their results based on the ILPD (Indian Liver Patient Dataset). The comparison between different classifier results is done of feature selection and without using feature selection technique. The development of intelligent liver disease prediction software (ILDPS) is done by using feature selection and classification prediction techniques based on software engineering model. The proposed work focuses on the development of the software that will help in the prediction of the level diseases based upon the various symptoms.

6] This research paper focuses on investigating liver patient datasets for building classification models in order to predict liver disease. This thesis implemented a feature model construction and comparative analysis for improving prediction accuracy of Indian liver patients in three phases. In first phase, min max normalization algorithm is applied on the original liver patient datasets collected from UCI repository. In liver dataset prediction second phase, by the use of PSO feature selection, subset (data) of liver patient dataset from whole normalized liver patient datasets are obtained which comprises only significant attributes. Third phase, classification algorithms are applied on the data set. In the fourth phase, the accuracy will be calculated using root mean square value, root mean error value. J48 algorithm is considered as the better performance algorithm after applying PSO feature selection. Finally, the evaluation is done based on accuracy values. Thus outputs show from proposed classification implementations indicate that J48 algorithm performances all other
classification algorithm.

7] This research paper presents an optimization approach to reduce and optimize the rules of disease diagnosis. The proposed approach uses the genetic algorithm to optimize boosted C5.0 algorithm which is a data mining algorithm and it has used to find rules for liver disease by considering a dataset and its results are compared with other proposed approaches. The main goal of this research is to use the genetic algorithm for optimizing a data mining method to reduce the number of rules which are extracted from boosted C5.0 algorithm and
find the effective attributes with the highest accuracy on patient diagnosis. Then, finally, we will have optimal and accurate rules for the liver disease diagnosis. After the genetic algorithm was implemented, totally 24 rules were generated instead of 92 rules and the comparable statistical parameters like accuracy, FDR, FPR, and other metrics show that the proposed method has better performance than boosted C5.0. Also, having 24 rules instead of 92 rules for diagnosing liver disease will help to reduce the time of diagnosis. So instead of using an evolutionary algorithm for producing rules, the genetic algorithm is used for improving and reducing rules of another algorithm.

8] This research paper treats an evaluation of the analyzed results of classification algorithms selected for better prediction based on the characteristics of data from the data set with liver disease. Here various classification algorithms were investigated and analyzed such as Naïve Bayes, Decision Tree, Multi-Layer Perceptron and k-NN which were used in a previous study, and helped to develop the dataset, and additionally Random Forest, Logistic are added in the study. Those algorithms were compared in several kinds of evaluating criteria like precision, recall, sensitivity, specificity, and so on. Through the experiments, it was seen that in view of precision, Naïve Bayes is preferable than others, but in other criteria such as Recall and Sensitivity, Logistic and Random Forest took precedence over other algorithms in the performance of prediction test as considering the algorithmic characteristics to liver patient data set.

9] This research paper surveyed some data mining techniques to predict liver disease at an earlier stage. The study analysed algorithms such as C4.5, Naive Bayes, Decision Tree, Support Vector Machine, Back Propagation Neural Network and Classification and Regression Tree Algorithms. These algorithms give various results based on speed, accuracy, performance and cost. It is seen that C4.5 gives better results compared to other algorithms. In future an improved C4.5 could be derived with various parameters. This paper gives generalization of various data mining techniques to diagnose liver disease at an earlier stage.

10] This research paper focuses on developing a machine learning model to predict FLD that could assist physicians in classifying high-risk patients and make a novel diagnosis, prevent and manage FLD. Here, classification models such as random forest (RF), Naïve Bayes (NB), artificial neural networks (ANN), and logistic regression (LR) were developed to predict FLD. The area under the receiver operating characteristic curve (ROC) was used to evaluate performances among the four models. In this study, the random forest model showed higher performance than other classification models. Implementation of a random forest model in the clinical setting could help physicians to stratify fatty liver patients for primary prevention, surveillance, early treatment, and management

11] The proposed methodology provided assistance in predicting liver disease by acquiring less data. In this article, there were various data mining classification techniques were used, in which bayesian logistic regression, multilayer perceptron, svm, attribute selected classifier, classification via regression, nbtree, j48 and random forest were involved for the evaluation of liver disease in patients. They created two types of datasets, i. e. , whole attributes dataset (wad) and selected attributes dataset (sad), and experimented with different data mining classifiers to predict liver disease.

12] The main goal of this study was to use classification algorithms to distinguish between liver patients and healthy people. Chemical components (bilirubin, albumin, proteins, alkaline phosphatase) present in the human body, as well as tests such as sgot and sgpt, determine whether a person was a patient or needed to have been diagnosed. Excessive consumption of alcohol, inhalation of toxic gases, and consumption of contaminated food, pickles, and medicines all contributed to the increase in patients with liver disease. The aim of this research was to analyze prediction algorithms with the aim of relieving doctors of their workload.

13] This paper presents a prediction based on a comparison with machine learning-based classification algorithms using performance measures such as accuracy, precision, and recall. The main purpose of this research was to find an effective way to predict the existence of liver disorders in patients by comparing different classification techniques.

14] Liver sickness might be distinguished with incalculable ordered systems, and these had been classified as the utilization forecast of a number of highlights and classifier blends. In this investigation, we applied five sorts of classifiers that was naïve bayes, logistic regression, support vector machines, random forest, k nearest neighbor for the examination of liver malady. The classification exhibitions were assessed with 5 distinctive by and large execution measurements, i. e. , precision, kappa, mean absolute error (mae), root mean square error (rmse), and f measures. The objective of this query worked was to foresee liver infection with different machines learned and picked the most efficient algorithm.

15] This study used data mining approaches to classify liver patients from healthy individuals. Four algorithms (naïve bayes, bagging, random forest and svm) were implemented for classification using the r platform. Furthermore, a hybrid neuro svm model using svm and feedforward artificial neural network (ann) was developed to improve the classification accuracy. The hybrid model was tested for its performance using statistical parameters such as root mean square error (rmse) and mean absolute percentage error (mape). The model resulted in a prediction accuracy of 98. 83%. The results indicate that the development of the hybrid model improved the prediction accuracy. To serve the medical community to predict liver disease among patients, a graphical user interface (gui) used r had been developed. The GUI was deployed as a package in the local r platform repository for users to perform prediction.

16] In this work, they constructed techniques for creating a computational model for the accurate prediction of liver disease. They used some practical classification algorithms: random forest, perceptron, decision tree, k-nearest neighbors (knn), and support vector machine (svm) to predict liver disease. Their work provides an implementation of hybrid model construction and comparative analysis to improve prediction performance. First, the classification algorithms were applied to the original datasets of liver patients collected from the uci repository. They then analyzed the features and improved and performed a comparative analysis between the classifiers. They investigated that the knn algorithm outperformed all other feature selection techniques.

17] On performance was very high in the decision tree classification algorithm for the Visakhapatnam and Tirupathi datasets, while the classification performance was very high in the random forest classification algorithm for the Vijayawada dataset. Construction time was more for mlp in Vijayawada dataset. This study motivated the development of the liver diagnosis app using a decision tree algorithm.

18] In this worked, a liver disease prediction system based on a modified convolutional neural network (mcnn-ldps) was presented for accurate liver disease prediction results. In the proposed research paper, dimensionality reduction was done using modified principal component analysis. Optimal feature selection was performed using the safest algorithm (artificial fish swarm algorithm). In the safe algorithm, the values of information gained and entropy was taken as input values, which proved the exact result. This research method was analyzed on a dataset of Indian liver patients. The analysis of the researched work proves that the proposed mcnn-ldps method achieves better results in terms of increased accuracy and precision. Here, the comparative analysis showed that mcnn-ldps achieves 4. 05% higher precision, 21. 23% f-measure, 4. 22% accuracy and 34. 26% recall. This research method was compared with the existing multi-layer perceptron neural network (mlpnn) for performance analysis. The main limitation of cnn was its inability to encode orientation and relative spatial relations, and viewpoint. Cnns did not encode the position and orientation of the data. Lack of ability has been spatially invariant to the input data sample. This was solved in this researched worked by combining the genetic algorithm with cnn method.

19] The technique predicts liver disease using patient data and machine-learned algorithms. The experiments and comparative analysis, increase the classification accuracy and also lead to shortening the classification time, thus helping to predict liver disease more effectively. The performance was measured in terms of accuracy, AUC score, precision, recall, and f-measure. Several classification algorithms were used, and based on the classification report and performance, the best model was selected and used to classify liver patients.

20] The purpose of this study was to extract significant predictors of liver disease from the medical analysis of 615 people using ml algorithms. Data visualizations were implemented to reveal significant findings such as missing values. Multiple imputations used chained equations (mice) was used to generate missing data points and principal component analysis (PCA) was used to reduce dimensionality. To validate the significant predictors obtained from PCA, variable importance evaluation using the Gini index was implemented. Training data (ntrain = 399) for training and testing data (ntest = 216) in ml methods were used to predict the classifications. The study compared binary classifier machine-learned algorithms (i. e. , artificial neural network, random forest (rf), and support vector machine) that were used on a published liver disease dataset to classify individuals with liver disease, enabling healthcare professionals to make a better diagnosis

21]Early diagnosis of liver disease was very important in order to save human lives and take appropriate measure to control the disease. In several fields, especially in the field of medical science, the ensemble method was successfully applied. This research work used different ensemble methods to investigate the early detection of liver disease. The selected dataset for this analysis had made up of attributes such as total bilirubin, direct bilirubin, age, sex, total protein, albumin, and globulin ratio. This research mainly aimed at measuring and comparing the efficiency of different ensemble methods. AdaBoost, LogitBoost, BeggRep, BeggJ48 and

Random Forest was the ensemble method used in this research. The study showed that LogitBoost was the most accurate model than other ensemble approaches.

22] One of the major diseases in the world was liver disease. The liver was an important part of the large organs in the human body and was also considered a gland. This was because it created and bites bile. Liver disease was a liver problem that caused the disease. The objective of this study was to propose a rule-based classification model with basic decision making techniques to predict various types of heart disease. To get better results the experiment was done using a different data mining algorithm compared with previous liver disease predictions. All experiments had been implemented at Azure Machine Learning tool. This paper was about to study the prediction of liver disease to produce better performance accuracy by comparing various mining data classification algorithms.

23] Hepatocellular carcinoma (HCC) was the second most common malignancy in egypt. Data mining was a method of predictive analysis which could explore tremendous volumes of information to discover hidden patterns and relationships. Our aimed here was to develop a non-invasive algorithm for prediction of hcc. Such an algorithm should been economical, reliable, easy to apply and acceptable by domain experts. Methods: this cross-sectional studied enrolled 315 patients with hepatitis c virus (hcv) related chronic liver disease (cld); 135 hcc, 116 cirrhotic patients without hcc and 64 patients with chronic hepatitis c. Used data mining analysis, we constructed a decision tree learned algorithm to predict hcc. Results: the decision tree algorithm was able to predict hcc with recall (sensitivity) of 83. 5% and precession (specificity) of 83. 3% used only routine data. The correctly classified instances was 259 (82. 2%), and the incorrectly classified instances was 56 (17. 8%). Out of 29 attributes, serum alpha fetoprotein (afp), with an optimal cutoff valued of ≥50. 3 ng/ml was selected as the best predictor of hcc. To a lesser extent, male sex, presence of cirrhosis, ast>64u/l, and ascites was variables associated with hcc. Conclusion: data mining analysis allows discovery of hidden patterns and enables the development of models to predict hcc, utilizing routine data as an alternative to ct and liver biopsy. This studied had highlighted a new cutoff for afp (≥50. 3

ng/ml). Presence of a score of >2 risk variables (out of 5) could successfully predict hcc with a sensitivity of 96% and specificity of 82%

24]The process of identifying patterns in huge datasets comprising methods such as machine learning, statistics, and database system could beconsidered data mining. It was a multidisciplinary field in computer science and it excerpted knowledge from the massive data set and converts into comprehensible format. The Medical environment was rich in information but weak in knowledge. Medical systems contained wealth of data which required a dominant analysis tool for determining concealed association and drift in data. The health care condition that comprehended to liver disorder was termed as Liver disease. Liver disorder led to abrupt health status that precisely governed the working of liver and intern affecting other organs in the body. Data mining classification techniques like Decision Tree, Linear Discriminant, SVM Fine Gaussian and Logistic Regression algorithms was applied. Laboratory parameters of the patients was used as the dataset. Data containeds features that could establish a rigorous model using Classification technique. MATLAB2016 was used in this paperfor implementing classification algorithm on the dataset. Linear Discriminant algorithm showed the highest prediction accuracy 95.8% and ROC was 0.93.

25] Hidden and unknown pattern was extracted from large data sets by performing several combinations of techniques from database and machine learning. Data mining played a significant role for handling a huge amount of data. Data mining deals with heterogeneity, privacy and correctness of data. Moreover, medical data mining was tremendously important research area and significant attempts had made in this area in recent years because inaccuracy in medical data systems may cause seriously disingenuous medical treatments. Medical data sets should be analyzed using suitable mining algorithms. To perform related operations, techniques of data mining had been used in developing medical systems for prediction of diseases through a set of medical data set. This paper reviews state of the art data mining algorithms for predicting different diseases and to analyze the performance of classification techniques i.e. Naive Bayes (NB), J48, REF Tree, Sequential Minimal Optimization (SMO), Multi-Layer Perceptron and Vote on different data sets of different diseases i.e. chronic kidney disease (CKD), heart disease, liver and diabetes. The experimental setup for performance evaluation of various algorithms using disease data sets retrieved from UCI respiratory had been made in WEKA tool. Values of different parameters i.e. correctly classified instances, precision, recall and F-Measure, time taken was analyzed by applying different classification algorithms.

26]Accuracy in data classification depended on the dataset used for learneding. Now-a-days the most important cause of death for both men and women was due to the Liver Problem. The healthcare industry collected a huge amount of data which was not properly mined and not put to the optimum use. Discovery of these hidden patterns and relationships often went unexploited. Our research focused on this aspect of Medical diagnosis by learneding pattern through the collected data of Liver disorder to develop intelligent medical decision support systems to help the physicians. In this paper, we proposed the use decision trees J48, Naive Bayes, ANN, ZeroR, 1BK and VFI algorithm to classify these diseases and compare the effectiveness, correction rate among them. Detection of Liver disease in its early stage was the key of its cure. It led to better performance of the classification models in terms of their predictive or descriptive accuracy, diminishing of computing time needed to build models as they learned faster, and better understanding of the models. In this paper, a comparative analysis of data classification accuracy using Liver disorder data in different scenarios was presented. The predictive performances of popular classifiers was compared quantitatively.

27] Diseases was the most serious social and expensive problem faced by the society. In the past decade, world had experienced a rapid increase in various Liver diseases and Ectopic Pregnancy. In this work we proposed a novel approach to evaluate the increased tendency of ectopic pregnancy and liver disease among such groups, using data mining techniques. It’s due to the modern adaptive life style and cultural changes of our society.

28] Now a days liver disease was extending markedly due to excessive alcohol consumption, smoking, drinking arsenic contaminated water, obesity, low immunity and by inheritance. Liver cancer symptoms may include jaundice, abdominal pain, fatigue, nausea, vomiting, back pain, abdominal swelling, weight loss, general itching. Selective algorithms may be used on medical instruments (e.g. CT scanner, MRI, Ultra sono, ECG etc.) to lessen time and cost on hepatic disease diagnosis. Here some of the algorithms such as, Naive Bayes classification (NBC), Bagging, KStar, Logistic and REP tree were used to evaluate the accuracy, precision, sensitivity and specificity. For these two data sets of UCLA and AP were considered to find out the best algorithm. The whole analysis was done using the software Weka 3.6.10. It was revealed that, KStar algorithm had the maximum accuracy, precision, sensitivity and specificity. On the other, minimum accuracy was obtained from NBC.Therefore K* algorithm could be used on diagnosis tools or instruments for rapid identification of specific liver disorder.

29] Patients with Liver disease had been continuously increasing because of excessive consumption of alcohol, inhale of harmful gases, intake of contaminated food, pickles and drugs. Automatic classification tools may reduce burden on doctors. This paper evaluated the selected classification algorithms for the classification of some liver patient datasets. The classification algorithms considered here were Naïve Bayes classifier, C4.5, Back propagation Neural Network algorithm, and Support Vector Machines. These algorithms was evaluated based on four criteria: Accuracy, Precision, Sensitivity and Specificity.

30] Liver diseases had produced a big data such as metabolomics analyses, electronic health records, and report including patient medical information, and disorders. However, these data must be analyzed and integrated if they was to produce models about physiological mechanisms of pathogenesis. We used machine learning based on classifier for big datasets in the fields of liver to Predict and therapeutic discovery. A dataset was developed with twenty three attributes that included the records of 7000 patients in which 5295 patients were male and rests were female. Support Vector Machine (SVM), Boosted C5.0, and Naïve Bayes (NB), data mining techniques was usedd with the proposed model for the prediction of liver diseases. The performance of these classifier techniques was evaluated with accuracy, sensitivity, specificity

IV. Datasets Description & Sample data

a) Data Set Information:

The data was received from UCI Machine Learning Repository. The information about the dataset is below. (UCI Machine Learning Repository, 2013)

The data set contains 416 liver patient records and 167 non liver patient records collected from North East of Andhra Pradesh, India. The "Dataset" column is a class label used to divide groups into liver patient (liver disease) or not (no disease). This data set contains 441 male patient records and 142 female patient records.

Dataset Link –

https://archive.ics.uci.edu/ml/datasets/ILPD+(Indian+Liver+Patient+Dataset)

b) Attribute Information:

- Age of the patient
- Gender of the patient
- Total Bilirubin
- Direct Bilirubin
- Alkaline Phosphotase
- Alamine Aminotransferase
- Aspartate Aminotransferase
- Total Protiens
- Albumin
- Albumin and Globulin Ratio
- Class: field used to split the data into two sets (patient with liver disease, or no disease)

c) Sample Dataset:

Table 2: This table contains the sample dataset containing 10 entries (of 5 liver disease and 5 non liver disease affect patients) from the dataset referred (liver disease prediction from UCI machine learning repository).

Before Preprocessing:

Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	Dataset
65	Female	0.7	0.1	187	16	18	6.8	3.3	0.9	1
62	Male	10.9	5.5	699	64	100	7.5	3.2		1
62	Male	7.3	4.1	490	60	68	7	3.3	0.89	1
58	Male	1	0.4	182	14	20	6.8	3.4	1	1
72	Male	3.9	2	195	27	59	7.3	2.4	0.4	1
70	Male	0.6	0.1	862	76	180	6.3	2.7	0.75	2
11	Male	0.7	0.1	592	26	29	7.1	4.2		2
50	Male	4.2	2.3	450	69	50	7	3	0.7	2
55	Female	8.2	3.9	1350	52	65	6.7	2.9	0.7	2
55	Female	10.9	5.1	1350	48	57	6.4	2.3	0.5	2

After Preprocessing:

Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	Dataset
65	Female	0.7	0.1	187	16	18	6.8	3.3	0.9	1
62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
62	Male	7.3	4.1	490	60	68	7	3.3	0.89	1
58	Male	1	0.4	182	14	20	6.8	3.4	1	1
72	Male	3.9	2	195	27	59	7.3	2.4	0.4	1
70	Male	0.6	0.1	862	76	180	6.3	2.7	0.75	2
11	Male	0.7	0.1	592	26	29	7.1	4.2	1.4	2
50	Male	4.2	2.3	450	69	50	7	3	0.7	2
55	Female	8.2	3.9	1350	52	65	6.7	2.9	0.7	2
55	Female	10.9	5.1	1350	48	57	6.4	2.3	0.5	2

V. PROPOSED ALGORITHM WITH FLOWCHART

Existing methods-

- Classified liver disease using artificial neural network (ANN) classification algorithm resulted in low accuracy.
- Testing accuracy of MLP was found to be 77.54 %, logistic regression method gave 74.36% and for SMO it gave 71.36 %.

We are going to use the following deep learning algorithms for our project and compare their accuracy –

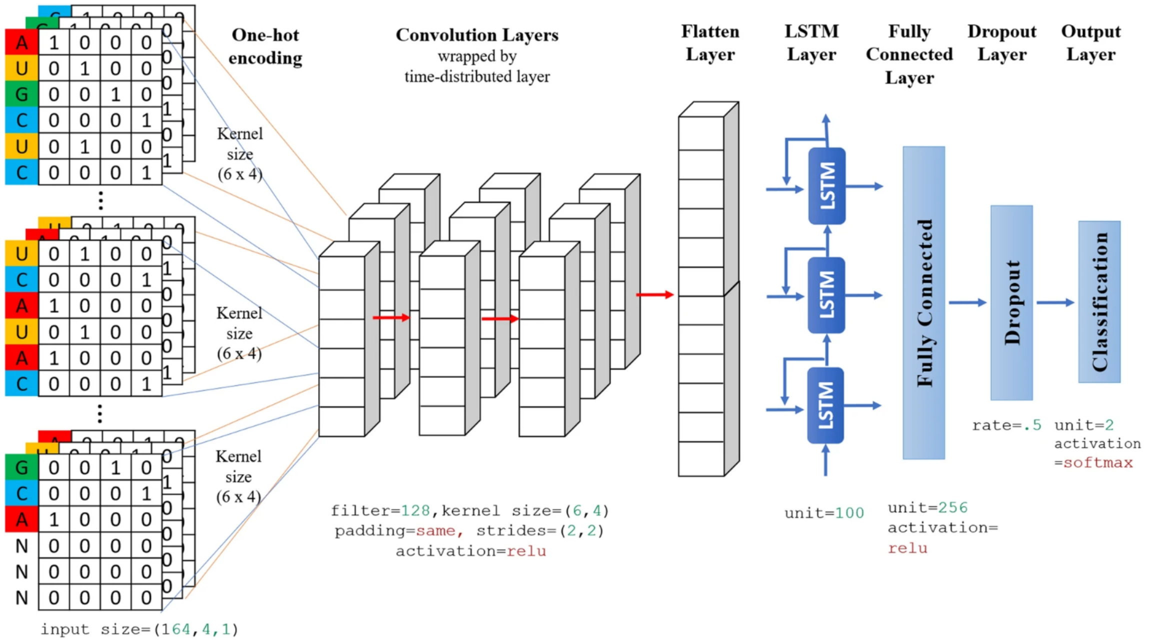
1. CNN + RNN
2. CNN + LSTM
3. CNN + GRU

LSTM-CNN:

There are several ways to enhance model performance, such as changing batch size and number of epochs, dataset curating, adjusting the ratio of the training, validation, and test datasets, changing loss functions and model architectures, and so on. In this project, we will improve model performance by changing the model architecture. More specifically, we will see if the CNN-LSTM model can predict liver disease cases better than the LSTM model.

The CNN layers that extract the feature from input data and LSTMs layers to provide sequence prediction

This collection demonstrates how to construct and train a deep, LSTM using CNN features.



Detailed architecture with visualization of the proposed methodology.

FIG 1: Shows the working of the hybrid CNN-LSTM algorithm

Flowchart:

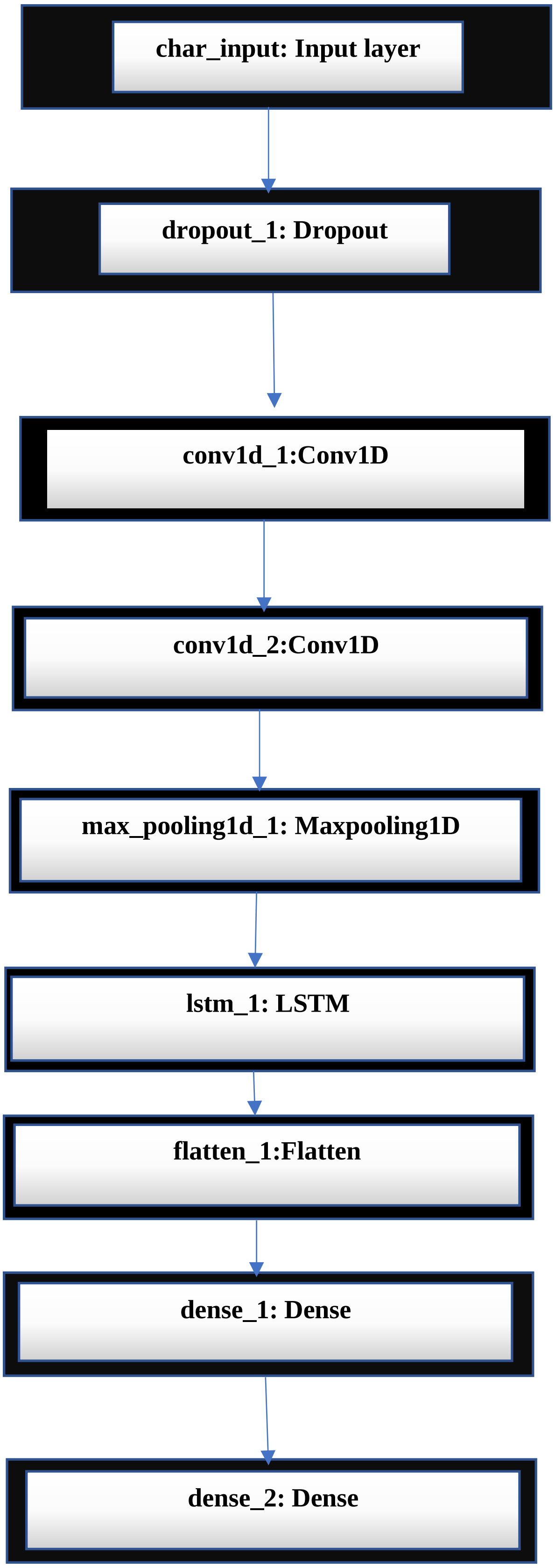


FIG 2: Shows the flowchart for CNN-LSTM hybrid algorithm

CNN + GRU

To combine the advantages of the GRU module which can well process time sequence data and the advantages of the CNN module which is ideal for handling high-dimensional data, the GRU-CNN hybrid neural networks was proposed

The proposed GRU-CNN hybrid neural network framework consists of GRU and CNN modules. The inputs are time series data collected from the energy system and information from the spatiotemporal matrix. The output is a prediction of future load values. As for the CNN module, it is good at processing two-dimensional data such as: B. Spatio-temporal matrices and images. The CNN engine uses local connections and shared weights to directly extract local features from spatio-temporal matrix data and obtain efficient representations through convolution and pooling layers. The structure of the CNN module contains two layers of convolutions and one flattening operation, and each layer of convolutions contains one convolution operation and one pooling operation. After the second pooling operation, the high-dimensional data is flattened to 1-dimensional data and the output of the CNN module is combined into a fully connected layer. On the other hand, the purpose of the GRU module is to grasp long-term dependencies, and the GRU module can learn useful information from historical data through memory cells over a long period of time, and unneeded information can be learned over a long period of time. be forgotten. Gate of Oblivion. The input to the GRU module is time series data. The GRU module contains many gate recursion units, and the outputs of all these gate recursion units are connected to fully connected layers. Finally, the load prediction result can be obtained by averaging over all neurons in the fully connected layer.

Flowchart

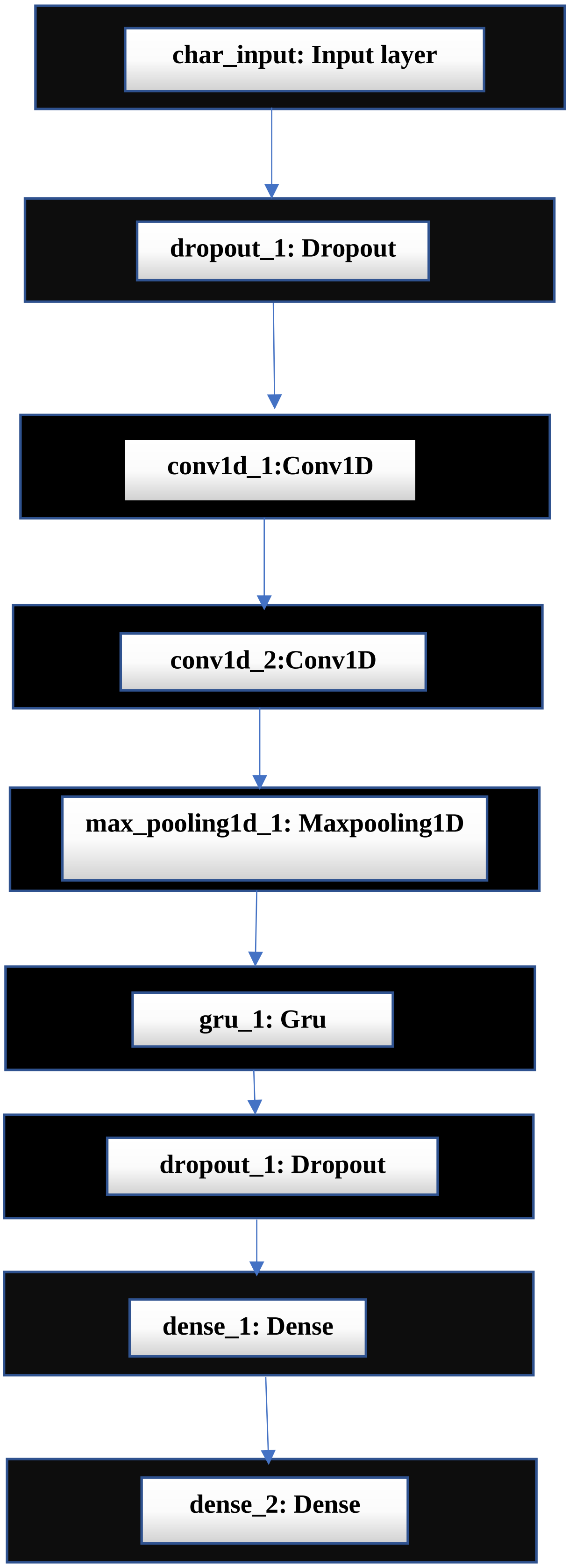


FIG 3: Shows the flowchart for CNN-GRU hybrid algorithm

CNN + RNN

The proposed model makes use of the ability of the CNN to extract local features and of the LSTM to learn long-term dependencies. First, a CNN layer of Conv1D is used for processing the input vectors and extracting the local features that reside at the text-level. The output of the CNN layer (i.e. the feature maps) are the input for the RNN layer of LSTM units/cells that follows. The RNN layer uses the local features extracted by the CNN and learns the long-term dependencies of the local features The proposed model makes use of the ability of the CNN to extract local features and of the LSTM to learn long-term dependencies. First, a CNN layer of Conv1D is used for processing the input vectors and extracting the local features that reside at the text-level. The output of the CNN layer (i.e. the feature maps) are the input for the RNN layer of LSTM units/cells that follows. The RNN layer uses the local features extracted by the CNN and learns the long-term dependencies of the local features

Flowchart:

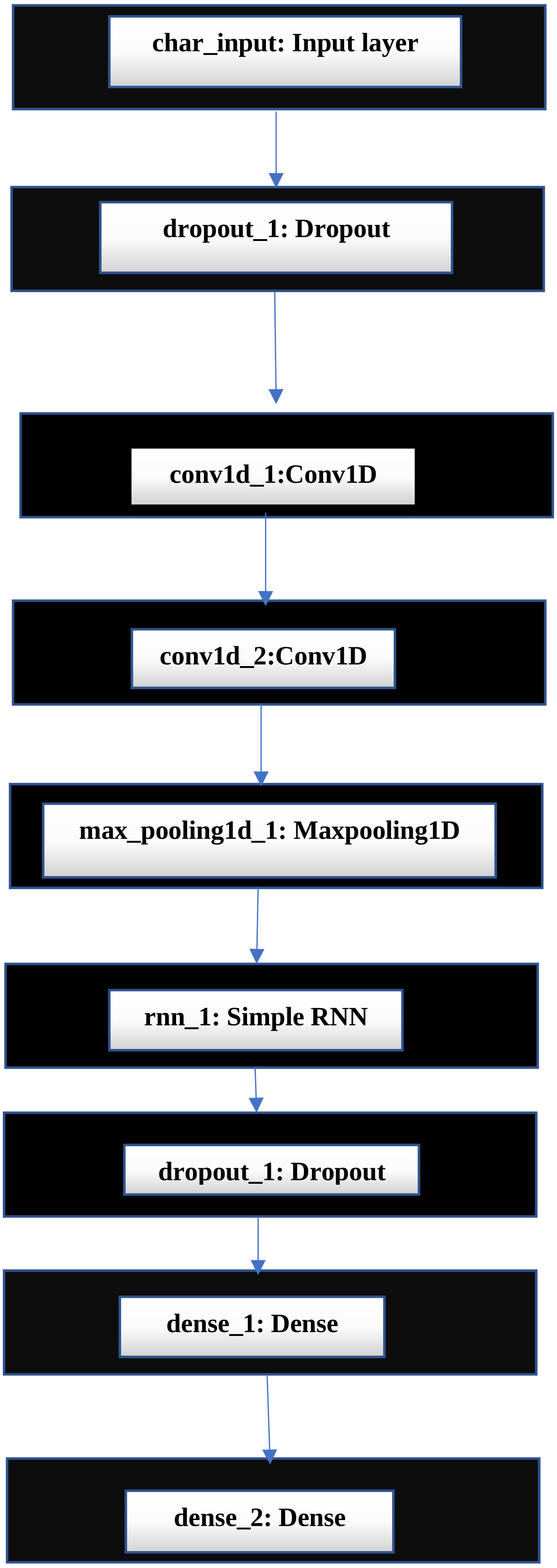
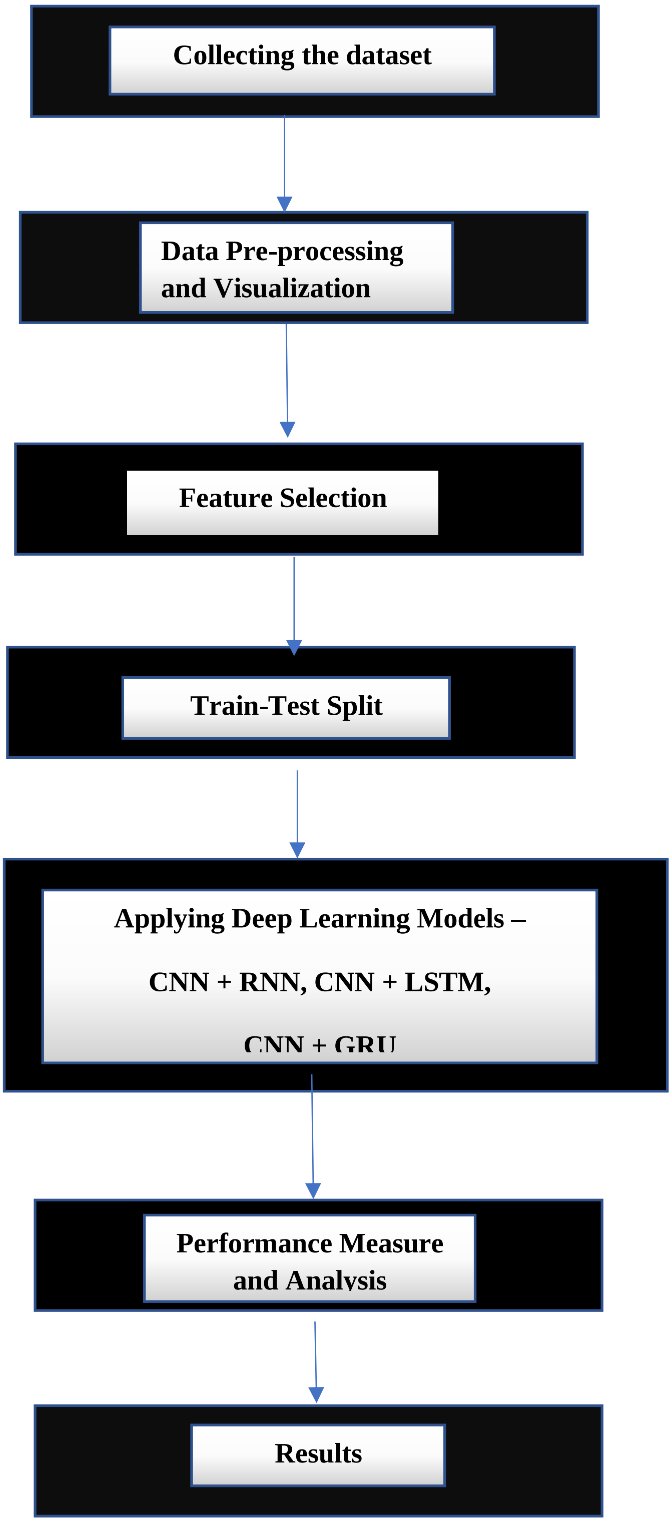


FIG 4: Shows the flowchart for CNN-RNN hybrid algorithm

Architecture



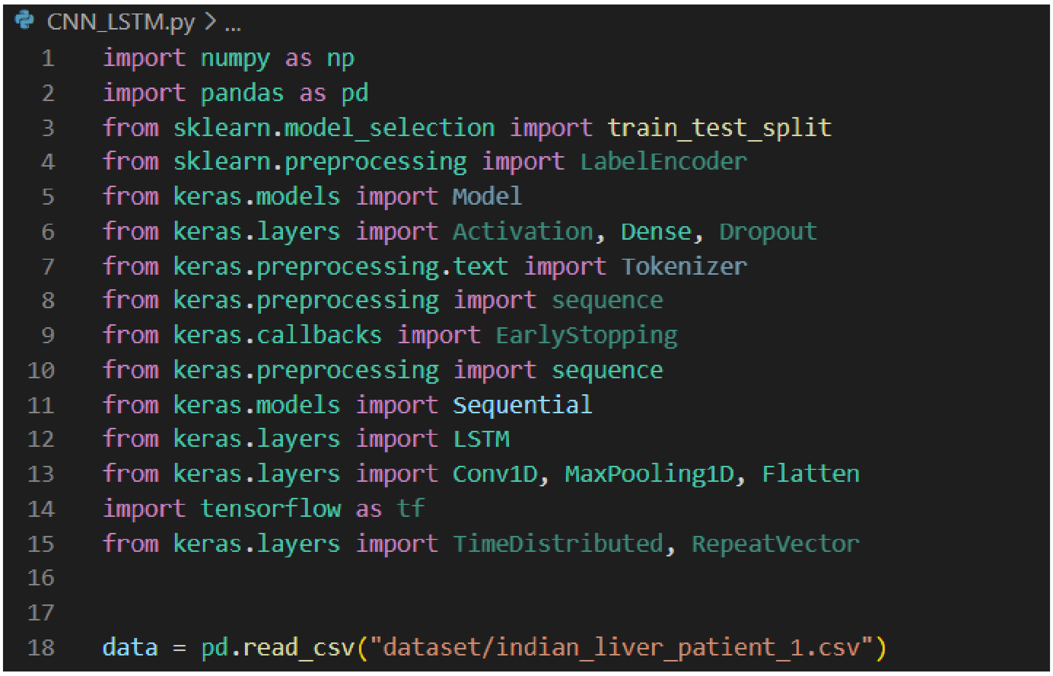
Collecting the dataset
Data Pre-processing and Visualization
Feature Selection
Train-Test Split
Applying Deep Learning Models –
CNN + RNN, CNN + LSTM,
CNN + GRU
Performance Measure and Analysis
Results

FIG 5: Shows the flowchart for the architecture of the code used for the calculation of accuracies of hybrid algorithms

IMPLEMENTATION

Loading the Dataset

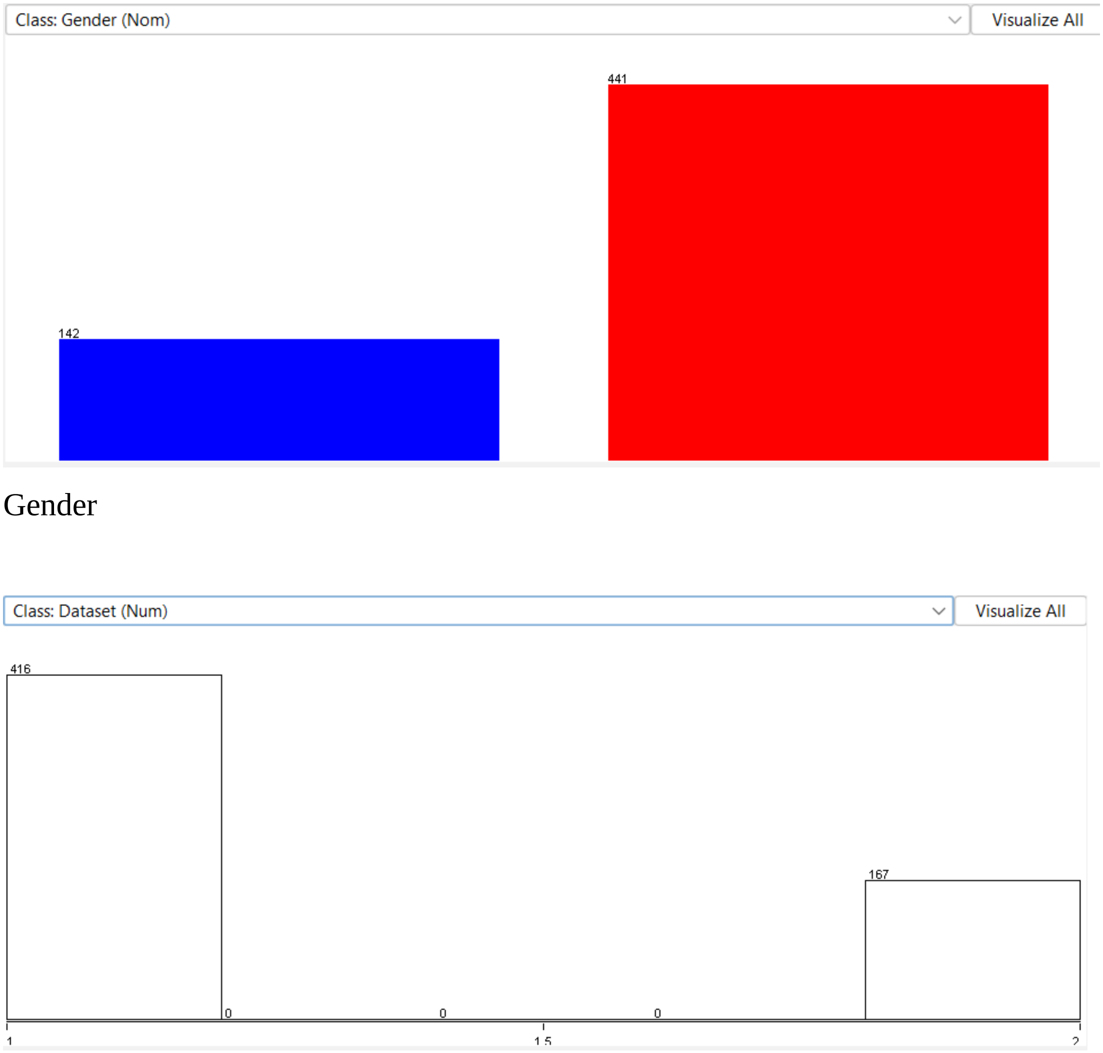
Before loading the dataset, we should import all the required libraries such as pandas, tokenizer, numpy, seaborn, label encoder to perform operations of implementing deep-learning models as well to perform steps of data pre- processing. Here, we have downloaded the dataset from the UCI repository and saved it as indian_liver_patient.csv which is now loaded and can be read as a data frame which is now named as data.



```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.models import Model
from keras.layers import Activation, Dense, Dropout
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras.callbacks import EarlyStopping
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Conv1D, MaxPooling1D, Flatten
import tensorflow as tf
from keras.layers import TimeDistributed, RepeatVector


data = pd.read_csv("dataset/indian_liver_patient_1.csv")
```

Data Pre-processing and Visualization

While creating our project, the dataset which we imported from the repository was not clean and formatted and before employing the deep learning models on the data, it is very necessary to clean and put formatted data, hence data pre-processing is required and it is basically the process of preparing the raw data and making it ready for the deep learning model. The following graphs show number of liver and non-liver disease along with male and females in the dataset.


Class: Gender (Nom)

Class: Dataset (Num)

Gender

FIG 7: Shows visual representation of the dataset and it’s columns

Liver Disease Class Histogram

1) Observations

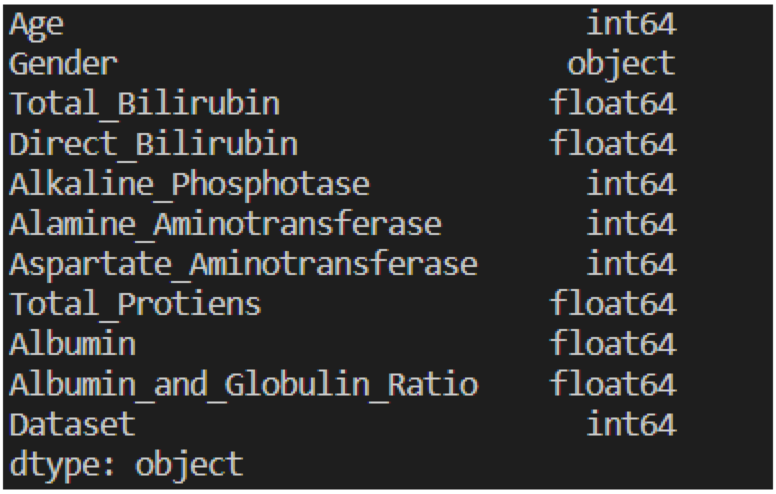
By using the command data.describe, we can figure out some of the observations of the dataset such as:

• Gender is a non- numerical variable and other all are numeric values.

• There are 10 features and 1 output which is the dataset.

• In Albumin and Globulin ratio we can see that there are four missing values.

• Values of Alkaline_Phosphatase, Alamine_Aminotransferase, Aspartate_Aminotransferase which are int should be converted for float values for better accuracy.

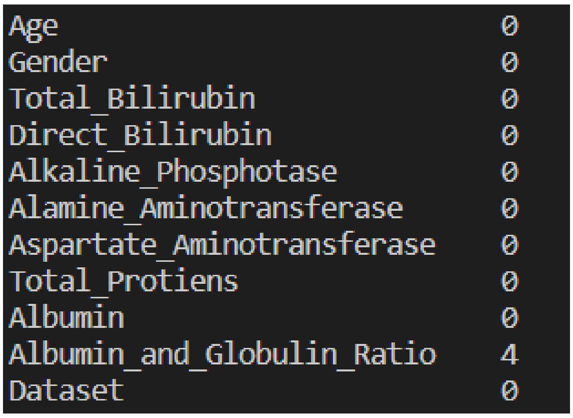


```
Age                              int64
Gender                          object
Total_Bilirubin                float64
Direct_Bilirubin               float64
Alkaline_Phosphotase             int64
Alamine_Aminotransferase         int64
Aspartate_Aminotransferase       int64
Total_Protiens                 float64
Albumin                        float64
Albumin_and_Globulin_Ratio     float64
Dataset                          int64
dtype: object
```

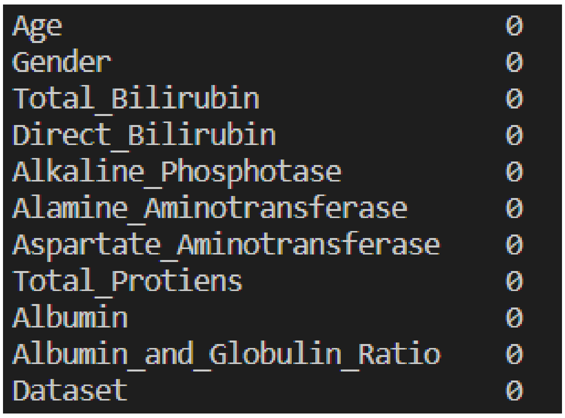
2) Filling of Missing Values

It is the process of identifying the missing variables and adding the mean values. For our dataset, the Albumin and Globulin ratio had four missing values which are replaced by considering the mean of that column which is 94.7

These values are filled in the second fig which shows that the column A/G ratio has no more null values.



```
Age                           0
Gender                        0
Total_Bilirubin               0
Direct_Bilirubin              0
Alkaline_Phosphotase          0
Alamine_Aminotransferase      0
Aspartate_Aminotransferase    0
Total_Protiens                0
Albumin                       0
Albumin_and_Globulin_Ratio    4
Dataset                       0
```



```
Age                           0
Gender                        0
Total_Bilirubin               0
Direct_Bilirubin              0
Alkaline_Phosphotase          0
Alamine_Aminotransferase      0
Aspartate_Aminotransferase    0
Total_Protiens                0
Albumin                       0
Albumin_and_Globulin_Ratio    0
Dataset                       0
```

3) Identifying Duplicate Values

Duplicate values were identified and by the observations we can see around 13 duplicate values but for a medical dataset duplicate value can exist and thus we are not dropping any of the duplicate values.

4) Resampling

Because of the imbalance in the dataset where we can observe a majority in liver disease patients and a minority in non-liver disease patients, smote which is synthesize minority oversampling technique which generates new values for the minority data and then synthesizes new samples for minorities. This will help in obtaining a better accuracy for the model during the implementation of machine learning models to the dataset in Weka Tool.

Also, we have applied PCA to achieve better results and then lastly made combinations using smote and PCA to compare the accuracy among various ML algorithms.

Feature Selection

Feature Selection is a process of figuring out which inputs are the best for the model and checking if there is a possibility of eliminating certain inputs. Considering the Dataset, we can see a very high linear relationship between Total and Direct Bilirubin and by considering this linear relationship, Direct Bilirubin can be opted to be dropped, but as per medical analysis Direct Bilirubin constitutes to almost 10% of the Total Bilirubin and this 10% may prove crucial in obtaining higher accuracy for the model, thus none of the features are removed.

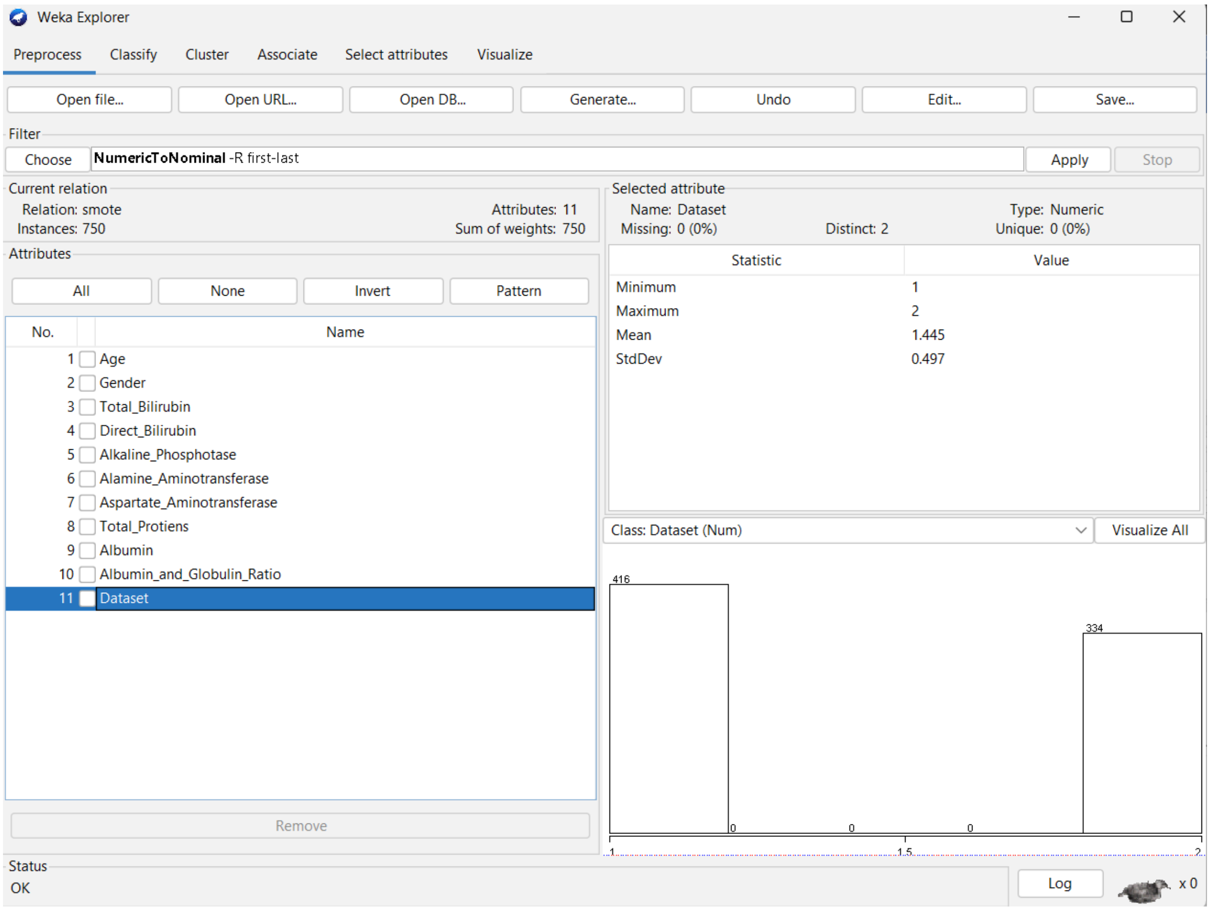
Train Test Split

The train-test split is a technique for evaluating the performance of a deep learning algorithm. The procedure involves taking a dataset and dividing it into two subsets. It is a fast and easy procedure to perform, the results of which allows us to compare the performance of deep learning algorithms for our predictive modelling problem. For the liver disease prediction model, we have considered 80 % of training data and 20 % of data for testing.

VI. EXPERIMENTS RESULTS

Applying Machine Learning Models Using WEKA Tool

SMOTE:



Filter

NumericToNominal -R first-last

Current relation

Relation: smote　Attributes: 11

Instances: 750　Sum of weights: 750

Selected attribute

Name: Dataset　Type: Numeric

Missing: 0 (0%)　Distinct: 2　Unique: 0 (0%)

Statistic	Value
Minimum	1
Maximum	2
Mean	1.445
StdDev	0.497

Class: Dataset (Num)

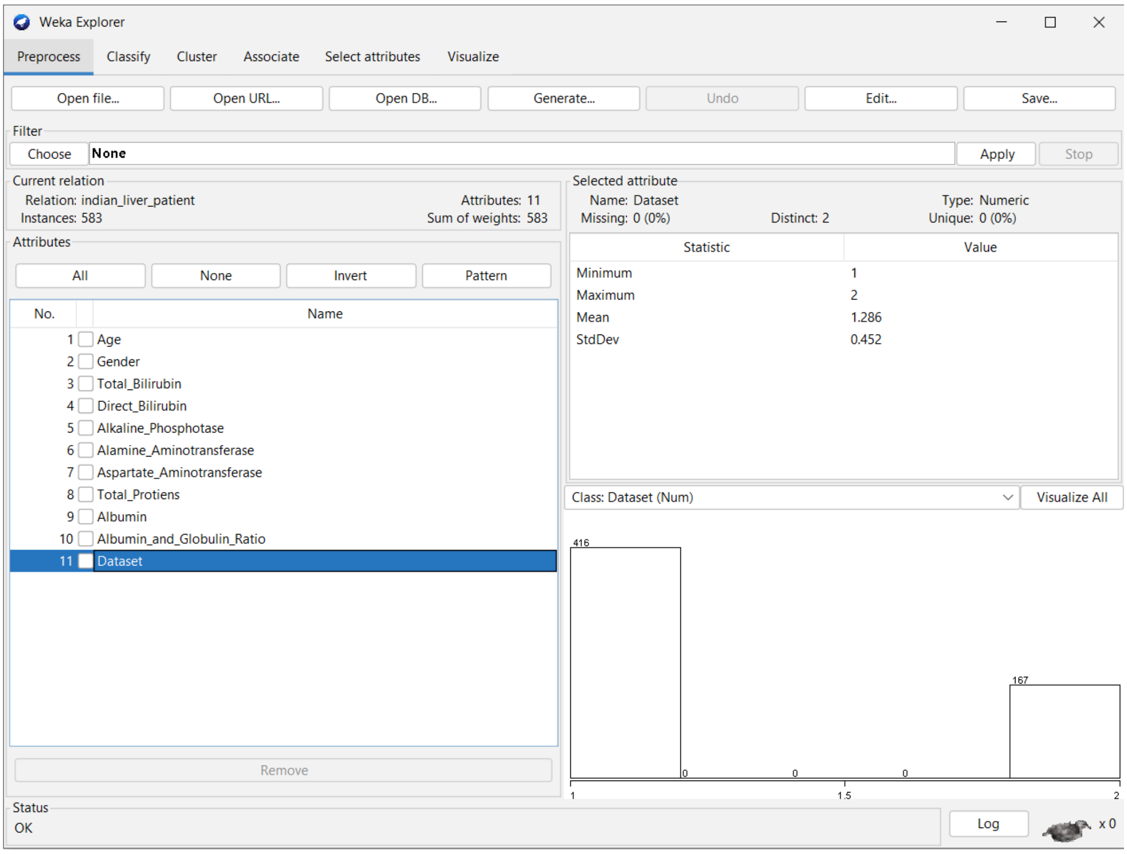
Attributes

No.	Name
1	Age
2	Gender
3	Total_Bilirubin
4	Direct_Bilirubin
5	Alkaline_Phosphotase
6	Alamine_Aminotransferase
7	Aspartate_Aminotransferase
8	Total_Protiens
9	Albumin
10	Albumin_and_Globulin_Ratio
11	Dataset

Status

OK

Using an imbalanced dataset:



Filter: None

Current relation
Relation: indian_liver_patient
Instances: 583
Attributes: 11
Sum of weights: 583

Attributes

No.	Name
1	Age
2	Gender
3	Total_Bilirubin
4	Direct_Bilirubin
5	Alkaline_Phosphotase
6	Alamine_Aminotransferase
7	Aspartate_Aminotransferase
8	Total_Protiens
9	Albumin
10	Albumin_and_Globulin_Ratio
11	Dataset

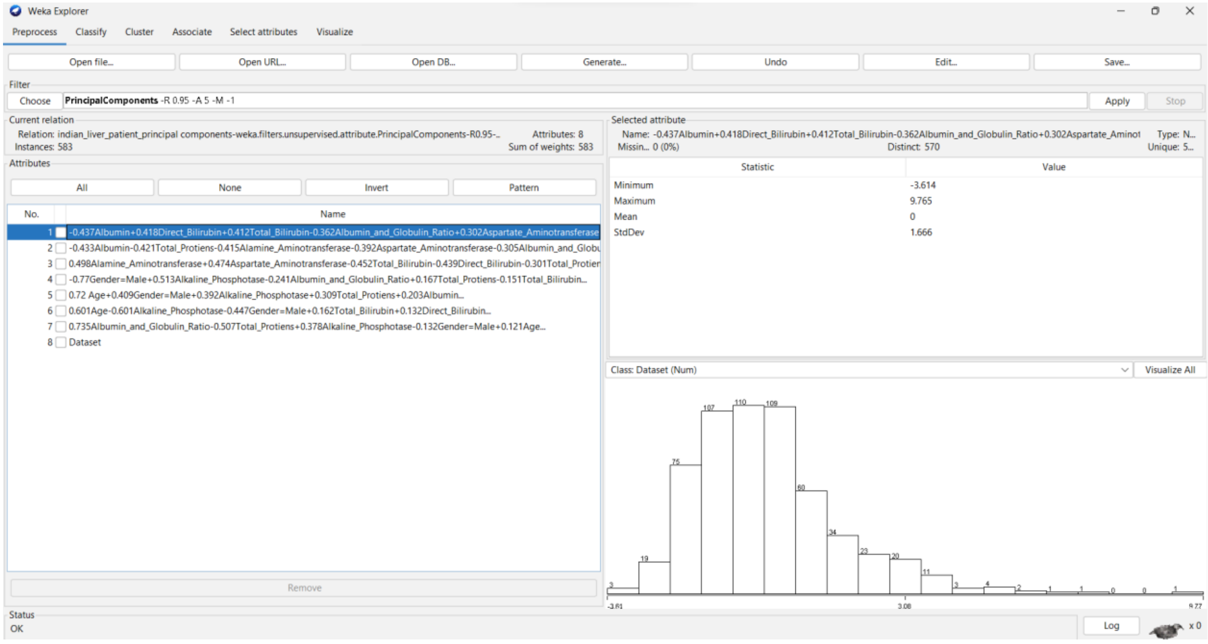
Selected attribute
Name: Dataset
Missing: 0 (0%)
Distinct: 2
Type: Numeric
Unique: 0 (0%)

Statistic	Value
Minimum	1
Maximum	2
Mean	1.286
StdDev	0.452

Class: Dataset (Num)

Status
OK

PCA



Filter: **PrincipalComponents** -R 0.95 -A 5 -M -1

Current relation
Relation: indian_liver_patient_principal components-weka.filters.unsupervised.attribute.PrincipalComponents-R0.95-...
Instances: 583
Attributes: 8
Sum of weights: 583

Attributes

No.	Name
1	-0.437Albumin+0.418Direct_Bilirubin+0.412Total_Bilirubin-0.362Albumin_and_Globulin_Ratio+0.302Aspartate_Aminotransferase
2	-0.433Albumin-0.421Total_Protiens-0.415Alamine_Aminotransferase-0.392Aspartate_Aminotransferase-0.305Albumin_and_Globu
3	0.498Alamine_Aminotransferase+0.474Aspartate_Aminotransferase-0.452Total_Bilirubin-0.439Direct_Bilirubin-0.301Total_Protien
4	-0.77Gender=Male+0.513Alkaline_Phosphotase-0.241Albumin_and_Globulin_Ratio+0.167Total_Protiens-0.151Total_Bilirubin...
5	0.72 Age+0.409Gender=Male+0.392Alkaline_Phosphotase+0.309Total_Protiens+0.203Albumin...
6	0.601Age-0.601Alkaline_Phosphotase-0.447Gender=Male+0.162Total_Bilirubin+0.132Direct_Bilirubin...
7	0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phosphotase-0.132Gender=Male+0.121Age...
8	Dataset

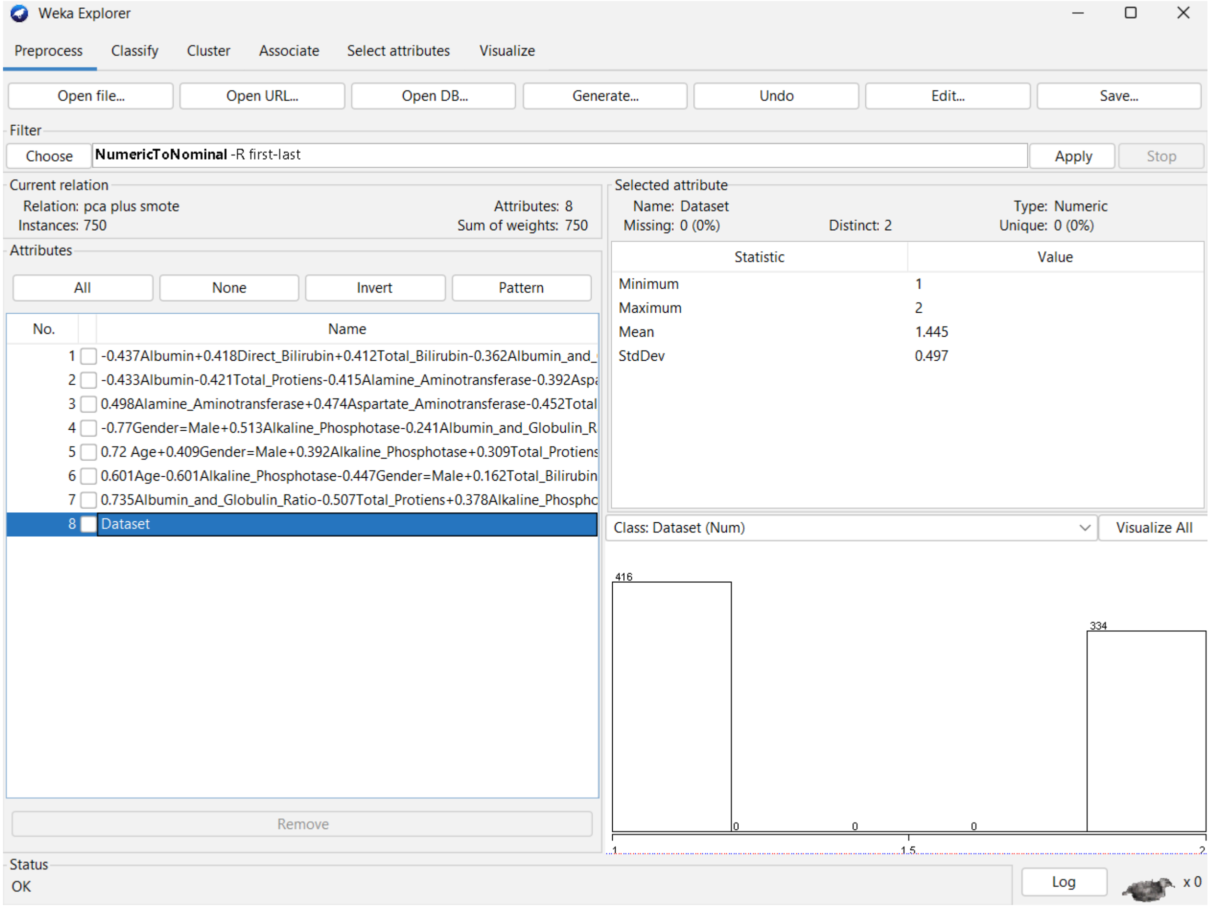
Selected attribute
Name: -0.437Albumin+0.418Direct_Bilirubin+0.412Total_Bilirubin-0.362Albumin_and_Globulin_Ratio+0.302Aspartate_Aminot
Missin... 0 (0%)
Distinct: 570
Type: N...
Unique: 5...

Statistic	Value
Minimum	-3.614
Maximum	9.765
Mean	0
StdDev	1.666

Class: Dataset (Num)

Status
OK

PCA + SMOTE



Filter

NumericToNominal -R first-last

Current relation

Relation: pca plus smote
Instances: 750
Attributes: 8
Sum of weights: 750

Attributes

No.		Name
1	□	-0.437Albumin+0.418Direct_Bilirubin+0.412Total_Bilirubin-0.362Albumin_and_
2	□	-0.433Albumin-0.421Total_Protiens-0.415Alamine_Aminotransferase-0.392Aspa
3	□	0.498Alamine_Aminotransferase+0.474Aspartate_Aminotransferase-0.452Total
4	□	-0.77Gender=Male+0.513Alkaline_Phosphotase-0.241Albumin_and_Globulin_R
5	□	0.72 Age+0.409Gender=Male+0.392Alkaline_Phosphotase+0.309Total_Protiens
6	□	0.601Age-0.601Alkaline_Phosphotase-0.447Gender=Male+0.162Total_Bilirubin
7	□	0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phospho
8	□	Dataset

Selected attribute

Name: Dataset
Missing: 0 (0%)
Distinct: 2
Type: Numeric
Unique: 0 (0%)

Statistic	Value
Minimum	1
Maximum	2
Mean	1.445
StdDev	0.497

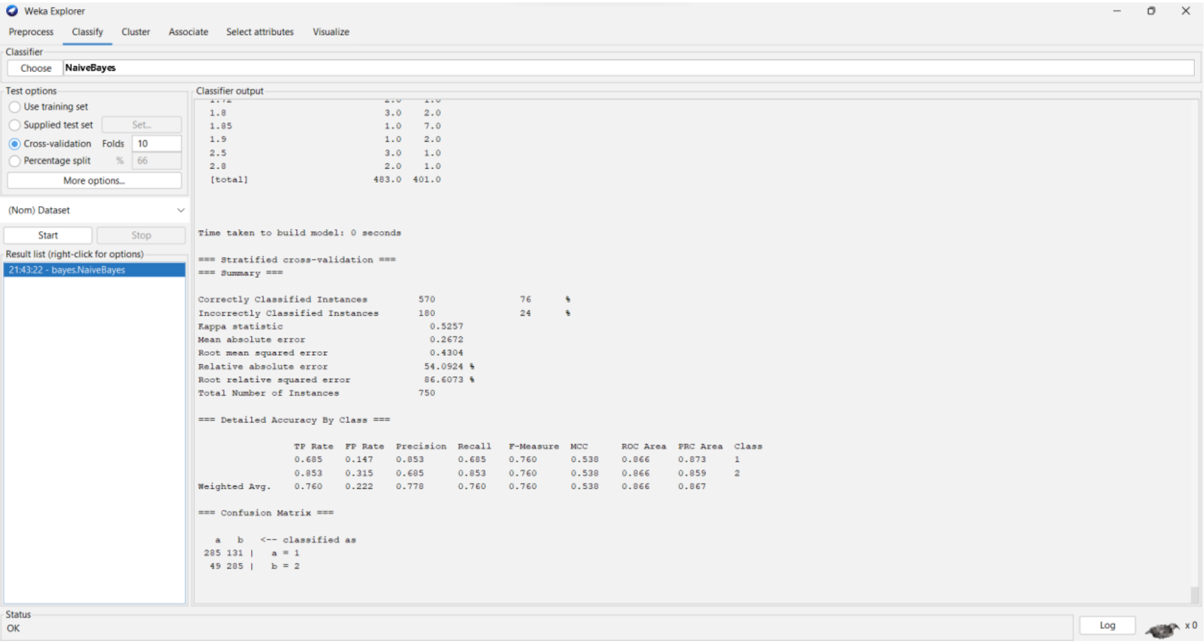
Class: Dataset (Num)

Status

OK

1. Naïve Bayes:

SMOTE: 76 %



NaiveBayes

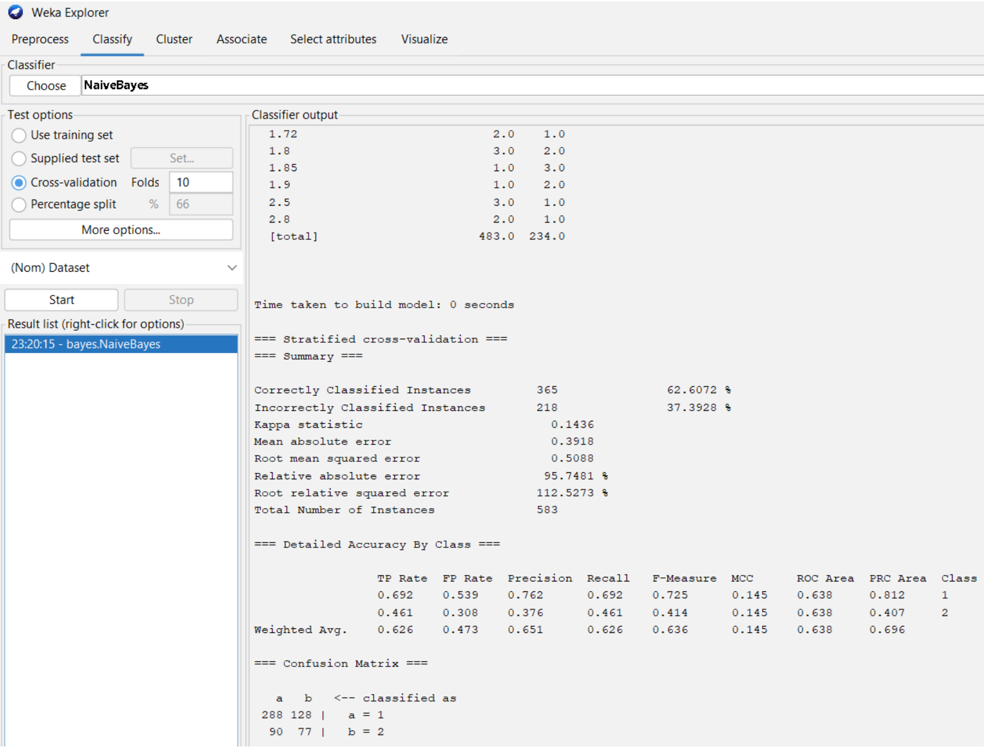
Folds 10

21:43:22 - bayes.NaiveBayes

OK

```
  1.72                          2.0    1.0
  1.8                           3.0    2.0
  1.85                          1.0    7.0
  1.9                           1.0    2.0
  2.5                           3.0    1.0
  2.8                           2.0    1.0
  [total]                     483.0  401.0



Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         570               76      %
Incorrectly Classified Instances       180               24      %
Kappa statistic                          0.5257
Mean absolute error                      0.2672
Root mean squared error                  0.4304
Relative absolute error                 54.0924 %
Root relative squared error             86.6073 %
Total Number of Instances              750

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.685    0.147    0.853      0.685    0.760      0.538    0.866     0.873     1
                 0.853    0.315    0.685      0.853    0.760      0.538    0.866     0.859     2
Weighted Avg.    0.760    0.222    0.778      0.760    0.760      0.538    0.866     0.867

=== Confusion Matrix ===

   a   b   <-- classified as
 285 131 |   a = 1
  49 285 |   b = 2
```

Without SMOTE: 62.6072%



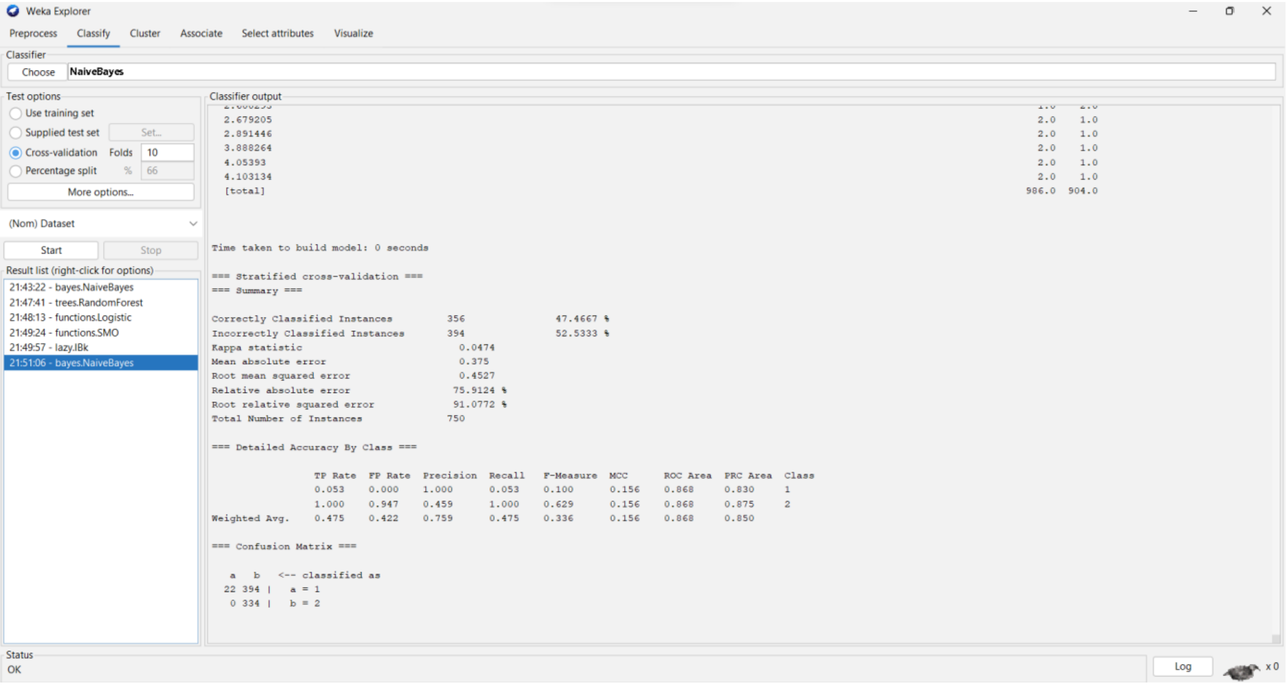
NaiveBayes

Folds 10

23:20:15 - bayes.NaiveBayes

```
  1.72                          2.0    1.0
  1.8                           3.0    2.0
  1.85                          1.0    3.0
  1.9                           1.0    2.0
  2.5                           3.0    1.0
  2.8                           2.0    1.0
  [total]                     483.0  234.0



Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         365               62.6072 %
Incorrectly Classified Instances       218               37.3928 %
Kappa statistic                          0.1436
Mean absolute error                      0.3918
Root mean squared error                  0.5088
Relative absolute error                 95.7481 %
Root relative squared error            112.5273 %
Total Number of Instances              583

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.692    0.539    0.762      0.692    0.725      0.145    0.638     0.812     1
                 0.461    0.308    0.376      0.461    0.414      0.145    0.638     0.407     2
Weighted Avg.    0.626    0.473    0.651      0.626    0.636      0.145    0.638     0.696

=== Confusion Matrix ===

   a   b   <-- classified as
 288 128 |   a = 1
  90  77 |   b = 2
```

PCA With SMOTE: 47.4667%



Classifier: NaiveBayes

Test options: ● Cross-validation Folds 10

(Nom) Dataset

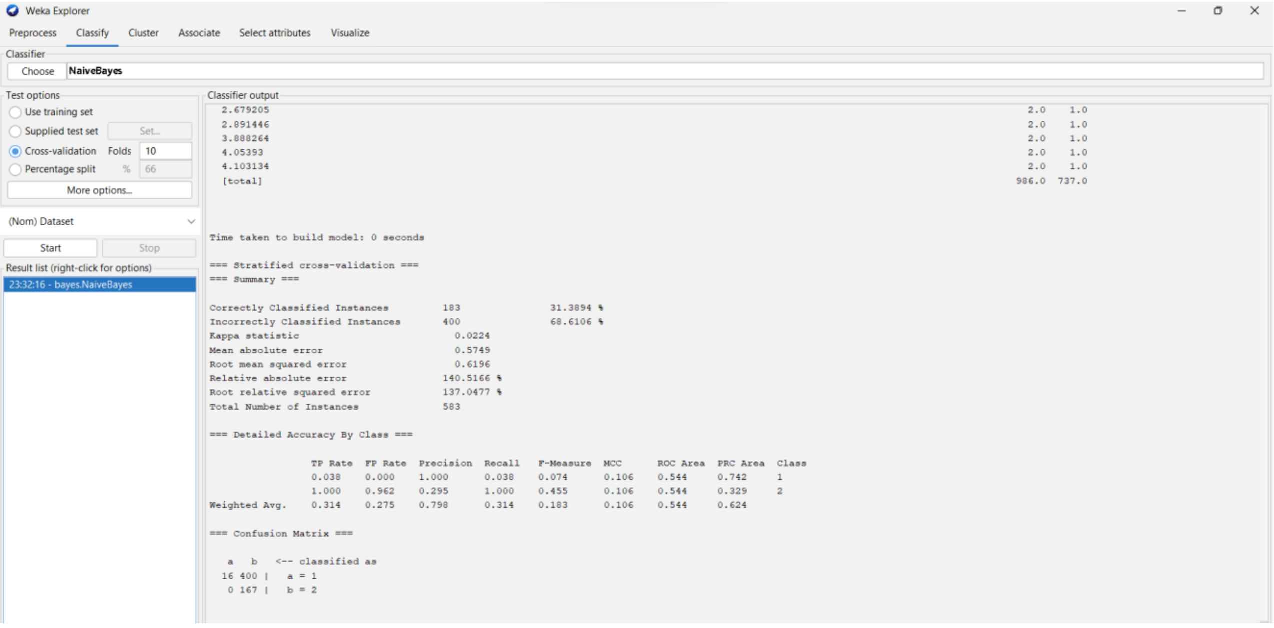
Result list (right-click for options)

21:43:22 - bayes.NaiveBayes
21:47:41 - trees.RandomForest
21:48:13 - functions.Logistic
21:49:24 - functions.SMO
21:49:57 - lazy.IBk
21:51:06 - bayes.NaiveBayes

```
  2.679205                                                              2.0    1.0
  2.891446                                                              2.0    1.0
  3.888264                                                              2.0    1.0
  4.05393                                                               2.0    1.0
  4.103134                                                              2.0    1.0
  [total]                                                             986.0  904.0


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         356               47.4667 %
Incorrectly Classified Instances       394               52.5333 %
Kappa statistic                          0.0474
Mean absolute error                      0.375
Root mean squared error                  0.4527
Relative absolute error                 75.9124 %
Root relative squared error             91.0772 %
Total Number of Instances              750

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.053    0.000    1.000      0.053    0.100      0.156    0.868     0.830     1
                 1.000    0.947    0.459      1.000    0.629      0.156    0.868     0.875     2
Weighted Avg.    0.475    0.422    0.759      0.475    0.336      0.156    0.868     0.850

=== Confusion Matrix ===

   a   b   <-- classified as
  22 394 |   a = 1
   0 334 |   b = 2
```

Status
OK

PCA Without SMOTE: 31.38%



Classifier: NaiveBayes

Test options: ● Cross-validation Folds 10

(Nom) Dataset

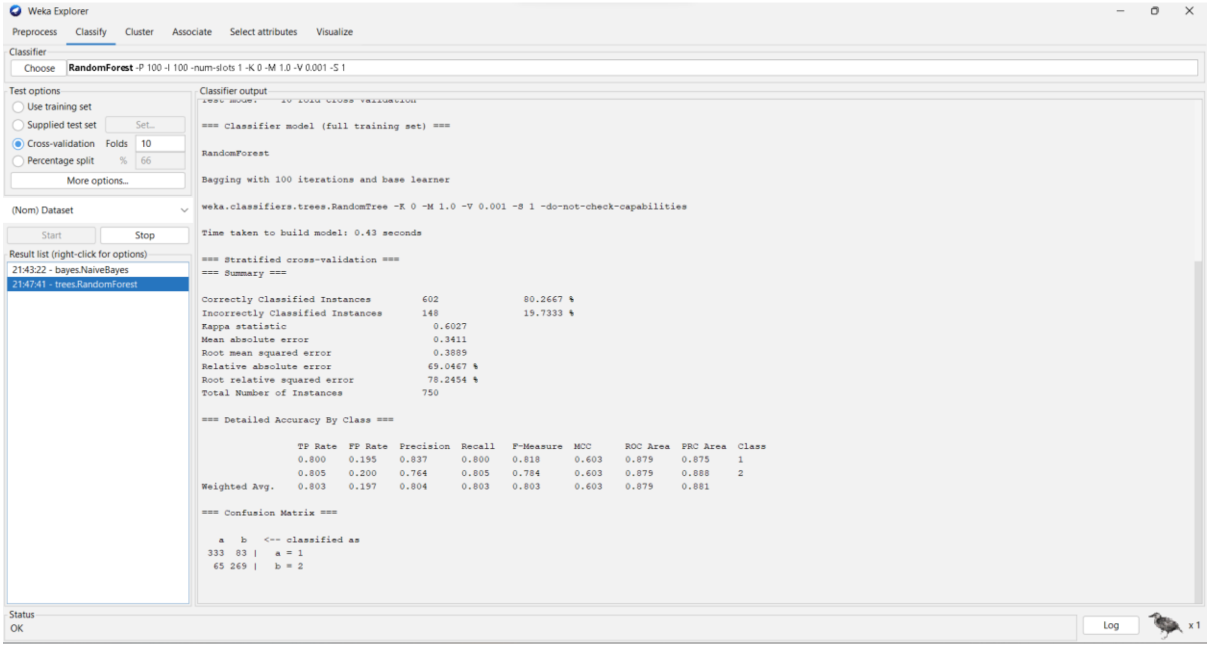
Result list (right-click for options)

23:32:16 - bayes.NaiveBayes

```
  2.679205                                                              2.0    1.0
  2.891446                                                              2.0    1.0
  3.888264                                                              2.0    1.0
  4.05393                                                               2.0    1.0
  4.103134                                                              2.0    1.0
  [total]                                                             986.0  737.0


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         183               31.3894 %
Incorrectly Classified Instances       400               68.6106 %
Kappa statistic                          0.0224
Mean absolute error                      0.5749
Root mean squared error                  0.6196
Relative absolute error                140.5166 %
Root relative squared error            137.0477 %
Total Number of Instances              583

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.038    0.000    1.000      0.038    0.074      0.106    0.544     0.742     1
                 1.000    0.962    0.295      1.000    0.455      0.106    0.544     0.329     2
Weighted Avg.    0.314    0.275    0.798      0.314    0.183      0.106    0.544     0.624

=== Confusion Matrix ===

   a   b   <-- classified as
  16 400 |   a = 1
   0 167 |   b = 2
```

2. Random Forest:

With SMOTE: 80.2667 %



Classifier: RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1

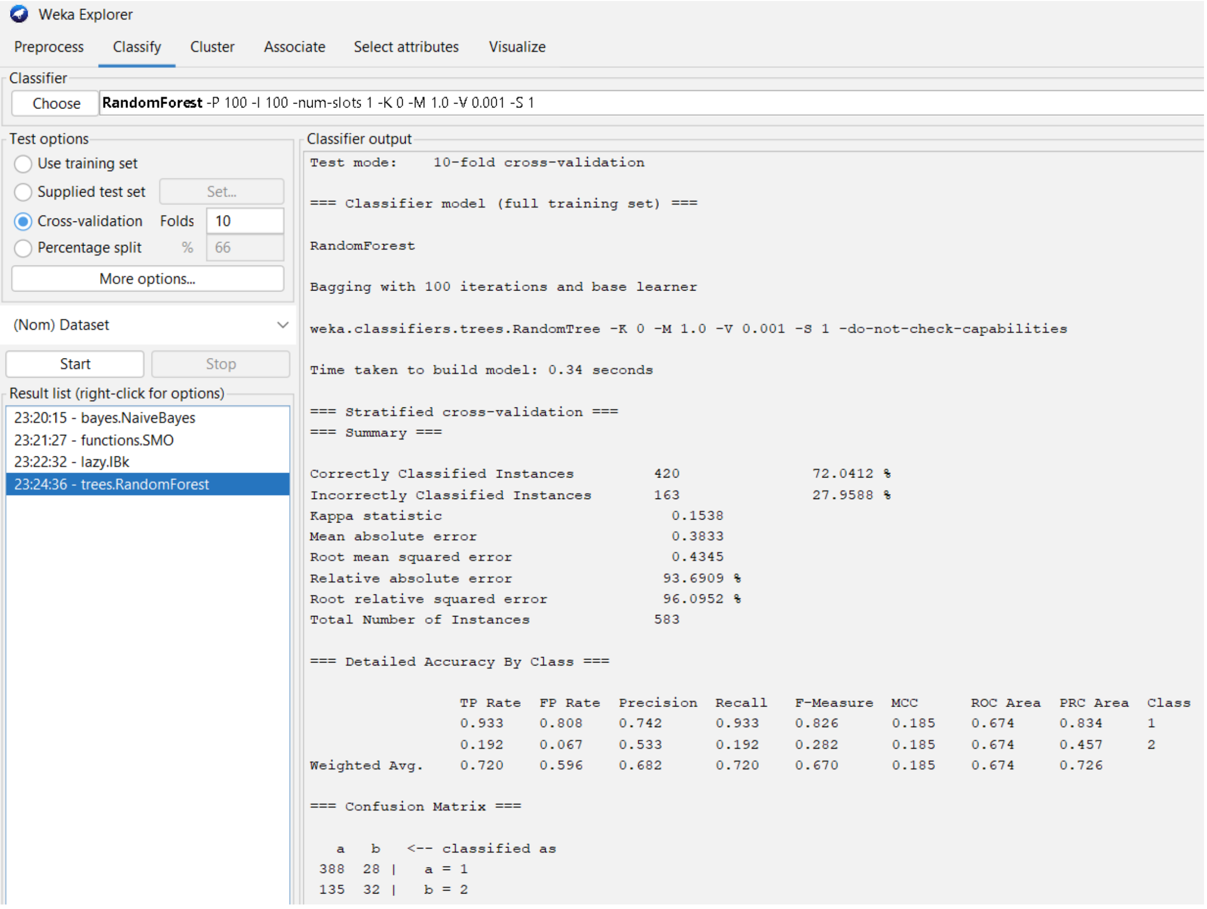
Test options: ● Cross-validation Folds 10

Result list (right-click for options)
21:43:22 - bayes.NaiveBayes
21:47:41 - trees.RandomForest

```
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 0.43 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         602               80.2667 %
Incorrectly Classified Instances       148               19.7333 %
Kappa statistic                          0.6027
Mean absolute error                      0.3411
Root mean squared error                  0.3889
Relative absolute error                 69.0467 %
Root relative squared error             78.2454 %
Total Number of Instances              750

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.800    0.195    0.837      0.800    0.818      0.603    0.879     0.875     1
                 0.805    0.200    0.764      0.805    0.784      0.603    0.879     0.888     2
Weighted Avg.    0.803    0.197    0.804      0.803    0.803      0.603    0.879     0.881

=== Confusion Matrix ===

   a   b   <-- classified as
 333  83 |   a = 1
  65 269 |   b = 2
```

Status
OK

Without SMOTE: 72.0412%



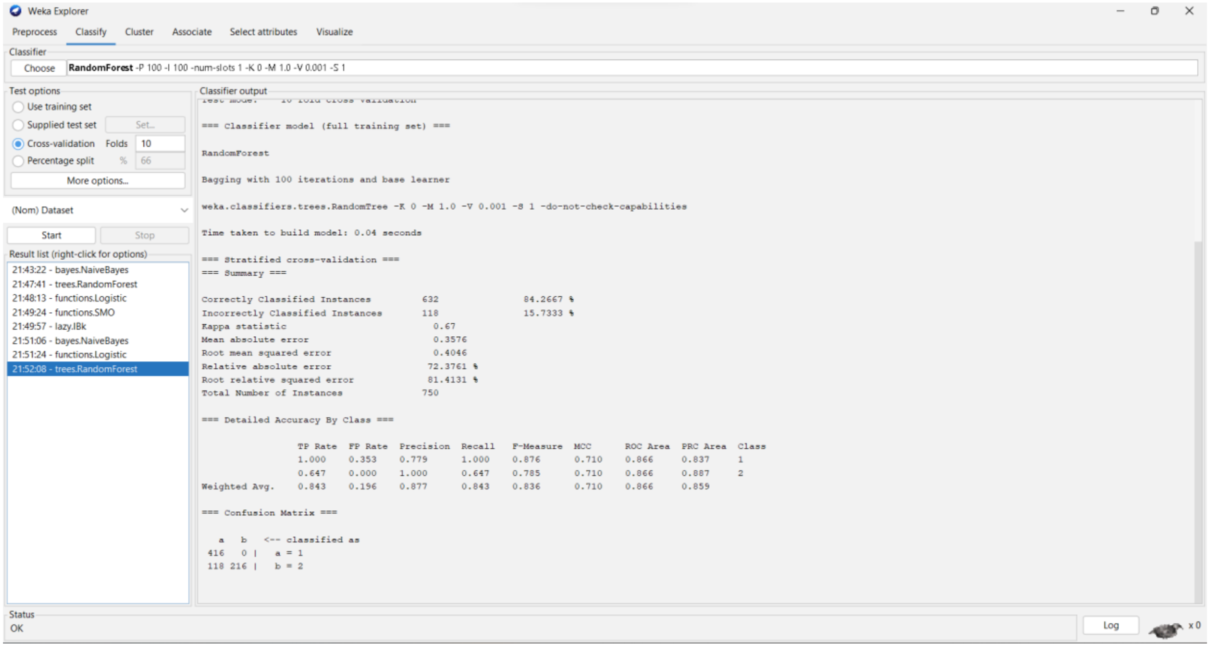
Classifier: RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1

Test options: ● Cross-validation Folds 10

Result list (right-click for options)
23:20:15 - bayes.NaiveBayes
23:21:27 - functions.SMO
23:22:32 - lazy.IBk
23:24:36 - trees.RandomForest

```
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 0.34 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         420               72.0412 %
Incorrectly Classified Instances       163               27.9588 %
Kappa statistic                          0.1538
Mean absolute error                      0.3833
Root mean squared error                  0.4345
Relative absolute error                 93.6909 %
Root relative squared error             96.0952 %
Total Number of Instances              583

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.933    0.808    0.742      0.933    0.826      0.185    0.674     0.834     1
                 0.192    0.067    0.533      0.192    0.282      0.185    0.674     0.457     2
Weighted Avg.    0.720    0.596    0.682      0.720    0.670      0.185    0.674     0.726

=== Confusion Matrix ===

   a   b   <-- classified as
 388  28 |   a = 1
 135  32 |   b = 2
```

PCA With SMOTE: 84.2667 %



Classifier: RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1

Test options
○ Use training set
○ Supplied test set
● Cross-validation Folds 10
○ Percentage split % 66

(Nom) Dataset

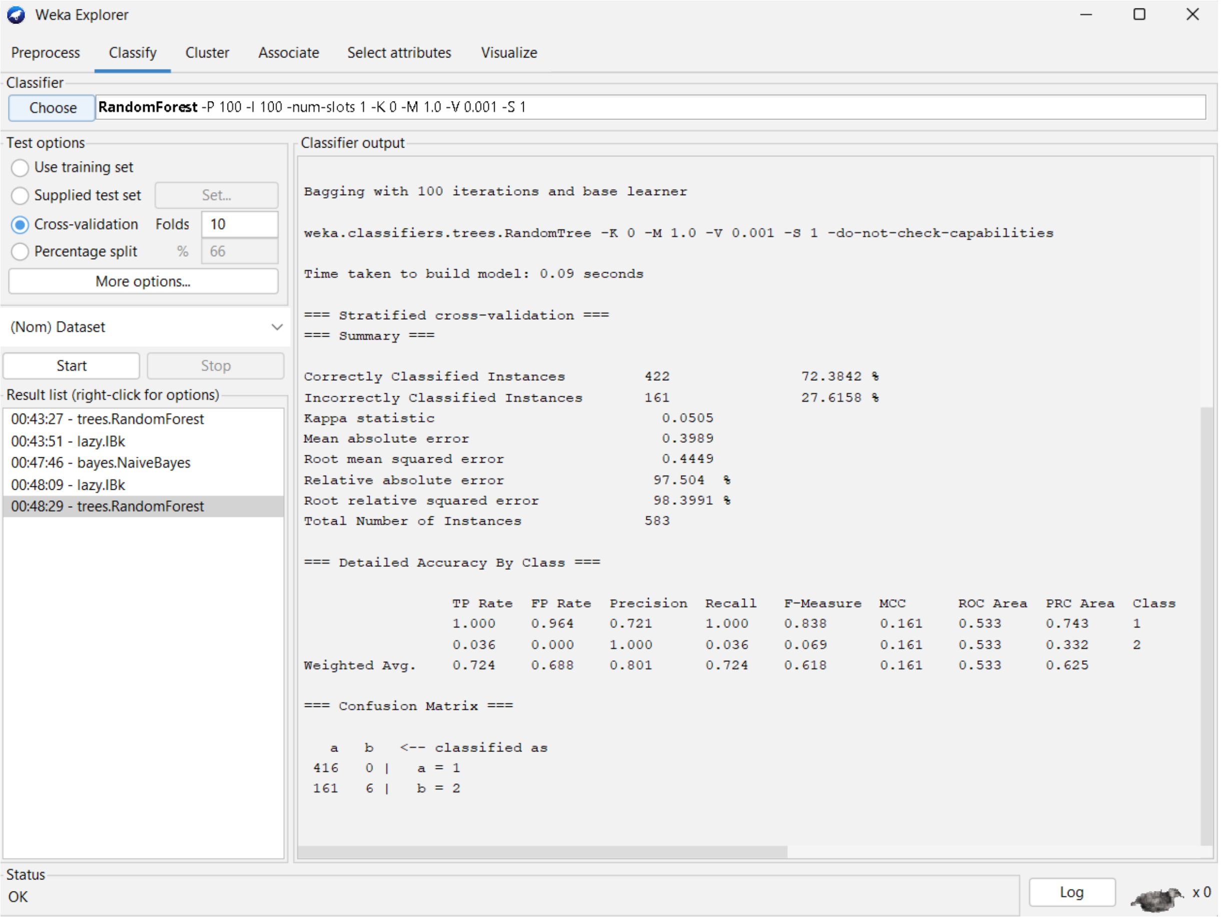
Result list (right-click for options)
21:43:22 - bayes.NaiveBayes
21:47:41 - trees.RandomForest
21:48:13 - functions.Logistic
21:49:24 - functions.SMO
21:49:57 - lazy.IBk
21:51:06 - bayes.NaiveBayes
21:51:24 - functions.Logistic
21:52:08 - trees.RandomForest

Classifier output

```
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 0.04 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         632               84.2667 %
Incorrectly Classified Instances       118               15.7333 %
Kappa statistic                          0.67
Mean absolute error                      0.3576
Root mean squared error                  0.4046
Relative absolute error                 72.3761 %
Root relative squared error             81.4131 %
Total Number of Instances              750

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.353    0.779      1.000    0.876      0.710    0.866     0.837     1
                 0.647    0.000    1.000      0.647    0.785      0.710    0.866     0.887     2
Weighted Avg.    0.843    0.196    0.877      0.843    0.836      0.710    0.866     0.859

=== Confusion Matrix ===

   a   b   <-- classified as
 416   0 |   a = 1
 118 216 |   b = 2
```

Status
OK

PCA Without SMOTE: 72.3842 %



Classifier: RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1

Test options
○ Use training set
○ Supplied test set
● Cross-validation Folds 10
○ Percentage split % 66

(Nom) Dataset

Result list (right-click for options)
00:43:27 - trees.RandomForest
00:43:51 - lazy.IBk
00:47:46 - bayes.NaiveBayes
00:48:09 - lazy.IBk
00:48:29 - trees.RandomForest

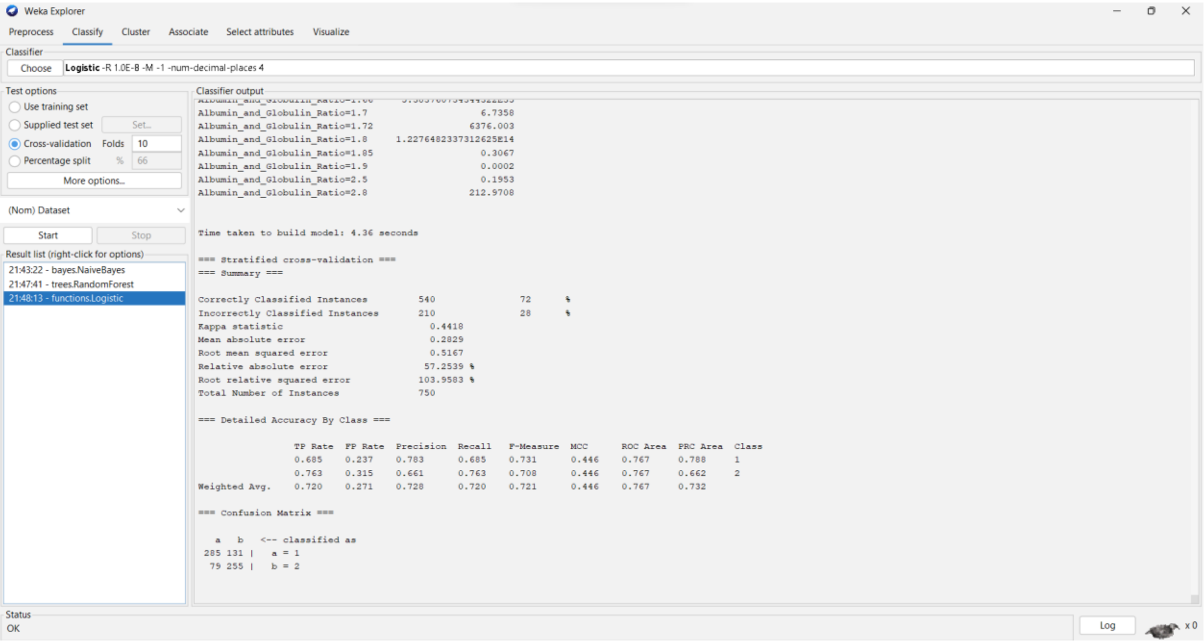
Classifier output

```
Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 0.09 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         422               72.3842 %
Incorrectly Classified Instances       161               27.6158 %
Kappa statistic                          0.0505
Mean absolute error                      0.3989
Root mean squared error                  0.4449
Relative absolute error                 97.504  %
Root relative squared error             98.3991 %
Total Number of Instances              583

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.964    0.721      1.000    0.838      0.161    0.533     0.743     1
                 0.036    0.000    1.000      0.036    0.069      0.161    0.533     0.332     2
Weighted Avg.    0.724    0.688    0.801      0.724    0.618      0.161    0.533     0.625

=== Confusion Matrix ===

   a   b   <-- classified as
 416   0 |   a = 1
 161   6 |   b = 2
```

Status
OK

3. Logistic:

With SMOTE: 72%



Classifier: **Logistic** -R 1.0E-8 -M -1 -num-decimal-places 4

Test options: ○ Use training set ○ Supplied test set ● Cross-validation Folds 10 ○ Percentage split % 66

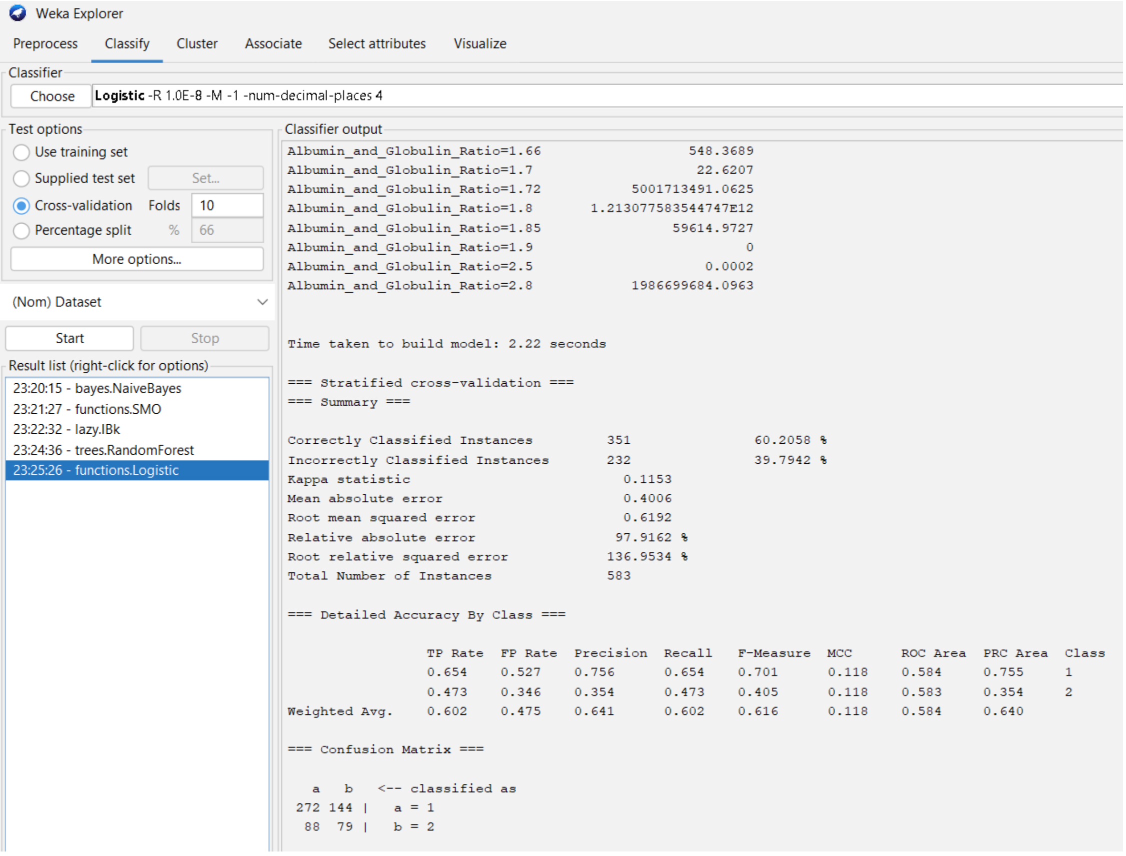
(Nom) Dataset

Result list (right-click for options)
- 21:43:22 - bayes.NaiveBayes
- 21:47:41 - trees.RandomForest
- 21:48:13 - functions.Logistic

Status: OK

```
Albumin_and_Globulin_Ratio=1.7                    6.7358
Albumin_and_Globulin_Ratio=1.72                 6376.003
Albumin_and_Globulin_Ratio=1.8     1.2276482337312625E14
Albumin_and_Globulin_Ratio=1.85                   0.3067
Albumin_and_Globulin_Ratio=1.9                    0.0002
Albumin_and_Globulin_Ratio=2.5                    0.1953
Albumin_and_Globulin_Ratio=2.8                  212.9708


Time taken to build model: 4.36 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         540               72      %
Incorrectly Classified Instances       210               28      %
Kappa statistic                          0.4418
Mean absolute error                      0.2829
Root mean squared error                  0.5167
Relative absolute error                 57.2539 %
Root relative squared error            103.9583 %
Total Number of Instances              750

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.685    0.237    0.783      0.685    0.731      0.446    0.767     0.788     1
                 0.763    0.315    0.661      0.763    0.708      0.446    0.767     0.662     2
Weighted Avg.    0.720    0.271    0.728      0.720    0.721      0.446    0.767     0.732

=== Confusion Matrix ===

   a   b   <-- classified as
 285 131 |   a = 1
  79 255 |   b = 2
```

Without SMOTE: 60.2058%



Classifier: **Logistic** -R 1.0E-8 -M -1 -num-decimal-places 4

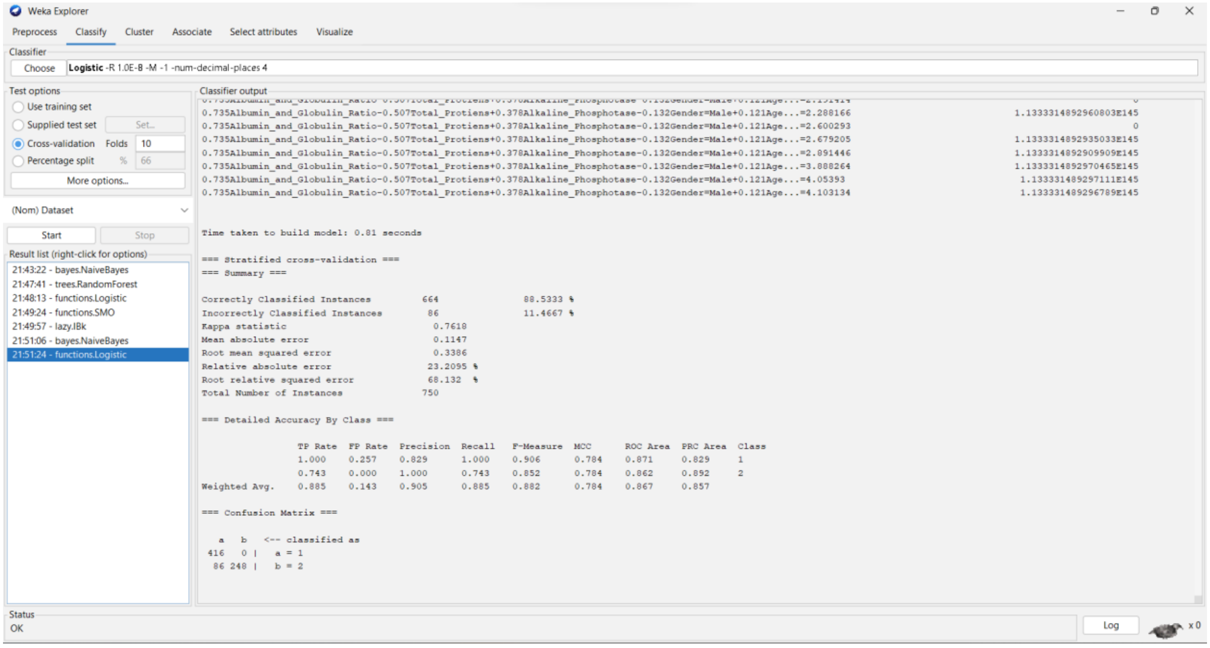
Test options: ○ Use training set ○ Supplied test set ● Cross-validation Folds 10 ○ Percentage split % 66

(Nom) Dataset

Result list (right-click for options)
- 23:20:15 - bayes.NaiveBayes
- 23:21:27 - functions.SMO
- 23:22:32 - lazy.IBk
- 23:24:36 - trees.RandomForest
- 23:25:26 - functions.Logistic

```
Albumin_and_Globulin_Ratio=1.66                 548.3689
Albumin_and_Globulin_Ratio=1.7                   22.6207
Albumin_and_Globulin_Ratio=1.72            5001713491.0625
Albumin_and_Globulin_Ratio=1.8       1.213077583544747E12
Albumin_and_Globulin_Ratio=1.85               59614.9727
Albumin_and_Globulin_Ratio=1.9                         0
Albumin_and_Globulin_Ratio=2.5                    0.0002
Albumin_and_Globulin_Ratio=2.8             1986699684.0963


Time taken to build model: 2.22 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         351               60.2058 %
Incorrectly Classified Instances       232               39.7942 %
Kappa statistic                          0.1153
Mean absolute error                      0.4006
Root mean squared error                  0.6192
Relative absolute error                 97.9162 %
Root relative squared error            136.9534 %
Total Number of Instances              583

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.654    0.527    0.756      0.654    0.701      0.118    0.584     0.755     1
                 0.473    0.346    0.354      0.473    0.405      0.118    0.583     0.354     2
Weighted Avg.    0.602    0.475    0.641      0.602    0.616      0.118    0.584     0.640

=== Confusion Matrix ===

   a   b   <-- classified as
 272 144 |   a = 1
  88  79 |   b = 2
```

PCA With SMOTE: 88.5333 %



Classifier: Logistic -R 1.0E-8 -M -1 -num-decimal-places 4

Test options: ○ Use training set ○ Supplied test set ● Cross-validation Folds 10 ○ Percentage split % 66

(Nom) Dataset

Result list (right-click for options)

- 21:43:22 - bayes.NaiveBayes
- 21:47:41 - trees.RandomForest
- 21:48:13 - functions.Logistic
- 21:49:24 - functions.SMO
- 21:49:57 - lazy.IBk
- 21:51:06 - bayes.NaiveBayes
- 21:51:24 - functions.Logistic

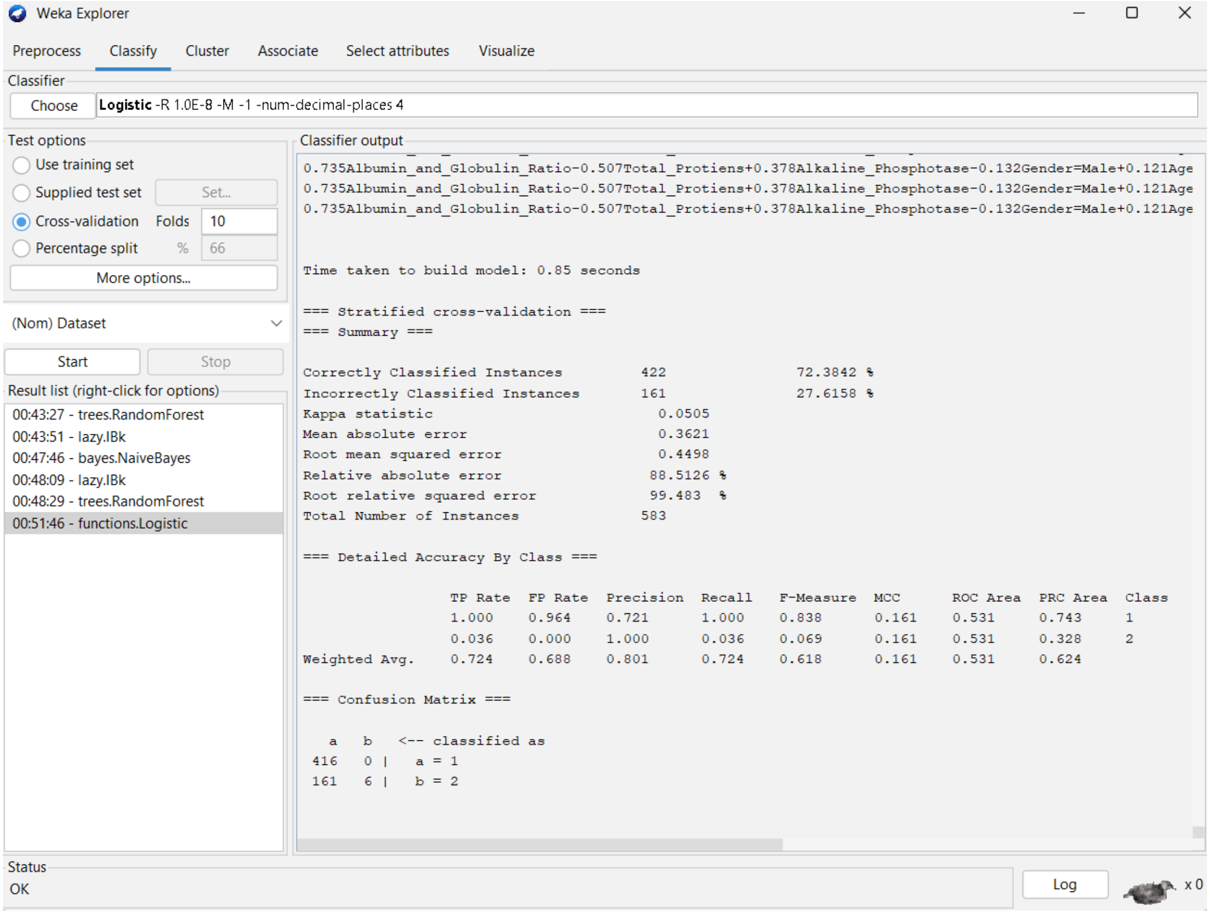
Classifier output

```
0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phosphotase-0.132Gender=Male+0.121Age...=2.288166          1.1333314892960803E145
0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phosphotase-0.132Gender=Male+0.121Age...=2.600293                               0
0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phosphotase-0.132Gender=Male+0.121Age...=2.679205          1.1333314892935033E145
0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phosphotase-0.132Gender=Male+0.121Age...=2.891446          1.1333314892909909E145
0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phosphotase-0.132Gender=Male+0.121Age...=3.888264          1.1333314892970465E145
0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phosphotase-0.132Gender=Male+0.121Age...=4.05393            1.13333148929711E145
0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phosphotase-0.132Gender=Male+0.121Age...=4.103134           1.13333148929678E145


Time taken to build model: 0.81 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         664               88.5333 %
Incorrectly Classified Instances        86               11.4667 %
Kappa statistic                          0.7618
Mean absolute error                      0.1147
Root mean squared error                  0.3386
Relative absolute error                 23.2095 %
Root relative squared error             68.132  %
Total Number of Instances              750

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.257    0.829      1.000    0.906      0.784    0.871     0.829     1
                 0.743    0.000    1.000      0.743    0.852      0.784    0.862     0.892     2
Weighted Avg.    0.885    0.143    0.905      0.885    0.882      0.784    0.867     0.857

=== Confusion Matrix ===

   a   b   <-- classified as
 416   0 |   a = 1
  86 248 |   b = 2
```

Status

OK

PCA Without SMOTE: 72.3842 %



Classifier: Logistic -R 1.0E-8 -M -1 -num-decimal-places 4

Test options: ○ Use training set ○ Supplied test set ● Cross-validation Folds 10 ○ Percentage split % 66

(Nom) Dataset

Result list (right-click for options)

- 00:43:27 - trees.RandomForest
- 00:43:51 - lazy.IBk
- 00:47:46 - bayes.NaiveBayes
- 00:48:09 - lazy.IBk
- 00:48:29 - trees.RandomForest
- 00:51:46 - functions.Logistic

Classifier output

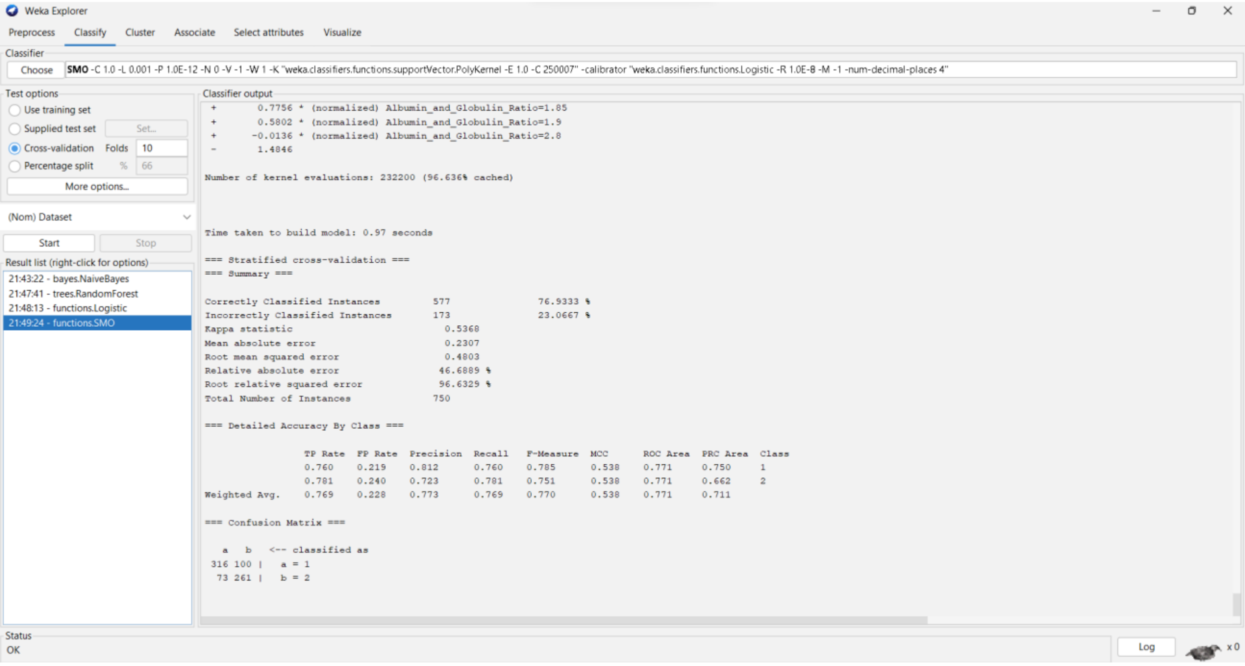
```
0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phosphotase-0.132Gender=Male+0.121Age
0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phosphotase-0.132Gender=Male+0.121Age
0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phosphotase-0.132Gender=Male+0.121Age


Time taken to build model: 0.85 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         422               72.3842 %
Incorrectly Classified Instances       161               27.6158 %
Kappa statistic                          0.0505
Mean absolute error                      0.3621
Root mean squared error                  0.4498
Relative absolute error                 88.5126 %
Root relative squared error             99.483  %
Total Number of Instances              583

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.964    0.721      1.000    0.838      0.161    0.531     0.743     1
                 0.036    0.000    1.000      0.036    0.069      0.161    0.531     0.328     2
Weighted Avg.    0.724    0.688    0.801      0.724    0.618      0.161    0.531     0.624

=== Confusion Matrix ===

   a   b   <-- classified as
 416   0 |   a = 1
 161   6 |   b = 2
```

Status

OK

4. SVM:

With SMOTE: 76.9333 %



Classifier: SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num-decimal-places 4"

Test options: ○ Use training set ○ Supplied test set ● Cross-validation Folds 10 ○ Percentage split % 66

(Nom) Dataset

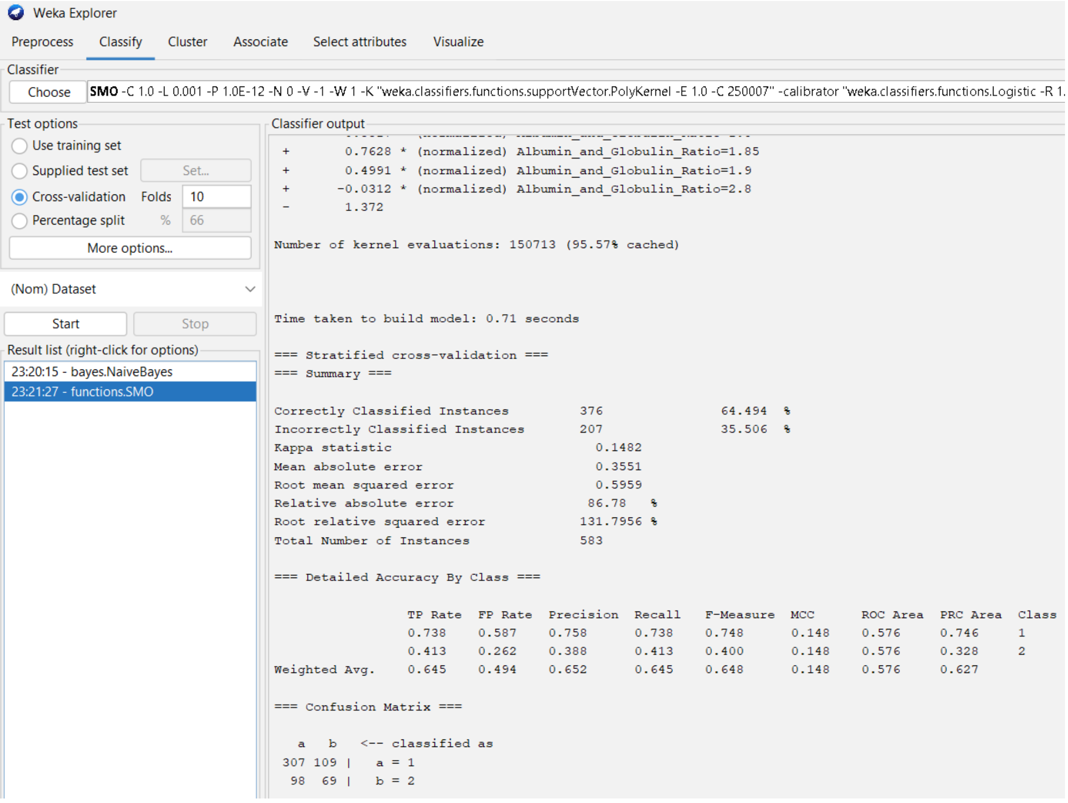
Result list (right-click for options)
- 21:43:22 - bayes.NaiveBayes
- 21:47:41 - trees.RandomForest
- 21:48:13 - functions.Logistic
- 21:49:24 - functions.SMO

Classifier output

```
 +       0.7756 * (normalized) Albumin_and_Globulin_Ratio=1.85
 +       0.5802 * (normalized) Albumin_and_Globulin_Ratio=1.9
 +      -0.0136 * (normalized) Albumin_and_Globulin_Ratio=2.8
 -       1.4846

Number of kernel evaluations: 232200 (96.636% cached)



Time taken to build model: 0.97 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         577               76.9333 %
Incorrectly Classified Instances       173               23.0667 %
Kappa statistic                          0.5368
Mean absolute error                      0.2307
Root mean squared error                  0.4803
Relative absolute error                 46.6889 %
Root relative squared error             96.6329 %
Total Number of Instances              750

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.760    0.219    0.812      0.760    0.785      0.538    0.771     0.750     1
                 0.781    0.240    0.723      0.781    0.751      0.538    0.771     0.662     2
Weighted Avg.    0.769    0.228    0.773      0.769    0.770      0.538    0.771     0.711

=== Confusion Matrix ===

   a   b   <-- classified as
 316 100 |   a = 1
  73 261 |   b = 2
```

Status
OK

Without SMOTE: 64.494%



Classifier: SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.functions.Logistic -R 1.

Test options: ○ Use training set ○ Supplied test set ● Cross-validation Folds 10 ○ Percentage split % 66

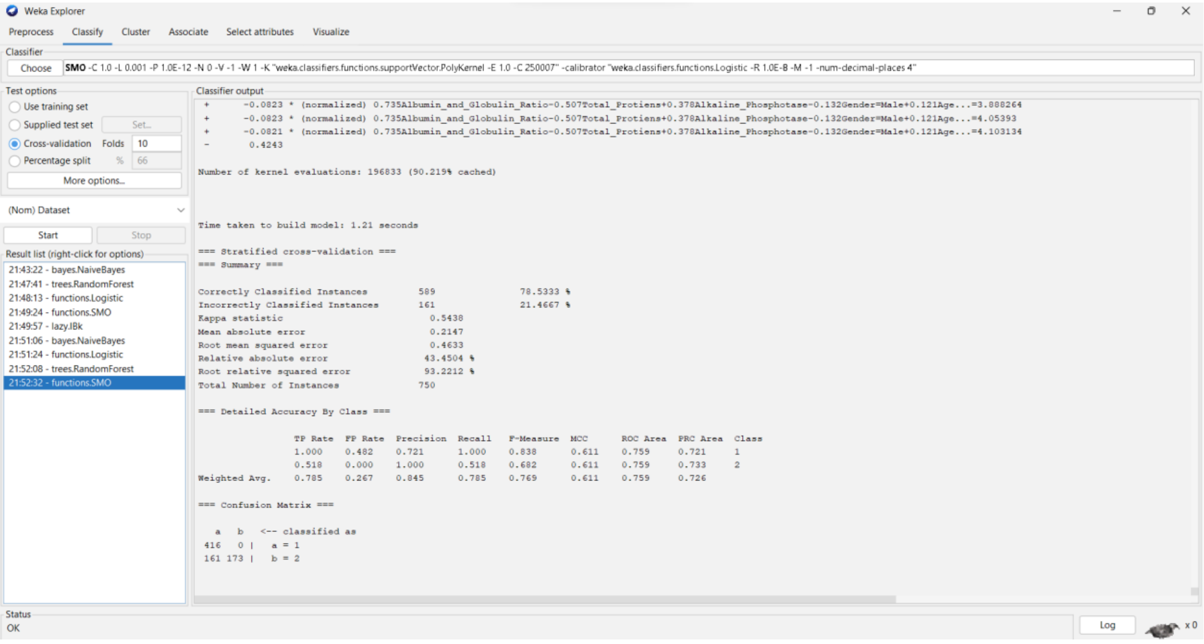
(Nom) Dataset

Result list (right-click for options)
- 23:20:15 - bayes.NaiveBayes
- 23:21:27 - functions.SMO

Classifier output

```
 +       0.7628 * (normalized) Albumin_and_Globulin_Ratio=1.85
 +       0.4991 * (normalized) Albumin_and_Globulin_Ratio=1.9
 +      -0.0312 * (normalized) Albumin_and_Globulin_Ratio=2.8
 -       1.372

Number of kernel evaluations: 150713 (95.57% cached)



Time taken to build model: 0.71 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         376               64.494  %
Incorrectly Classified Instances       207               35.506  %
Kappa statistic                          0.1482
Mean absolute error                      0.3551
Root mean squared error                  0.5959
Relative absolute error                 86.78   %
Root relative squared error            131.7956 %
Total Number of Instances              583

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.738    0.587    0.758      0.738    0.748      0.148    0.576     0.746     1
                 0.413    0.262    0.388      0.413    0.400      0.148    0.576     0.328     2
Weighted Avg.    0.645    0.494    0.652      0.645    0.648      0.148    0.576     0.627

=== Confusion Matrix ===

   a   b   <-- classified as
 307 109 |   a = 1
  98  69 |   b = 2
```

PCA With SMOTE: 78.5333 %



Classifier: SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num-decimal-places 4"

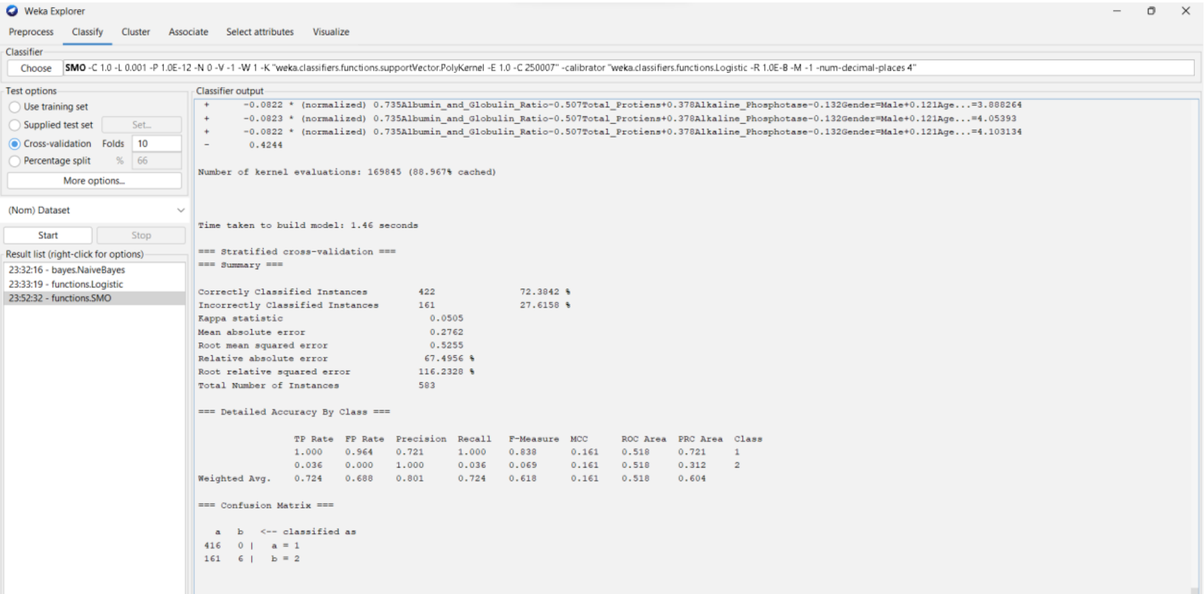
Test options: ○ Use training set ○ Supplied test set ● Cross-validation Folds 10 ○ Percentage split % 66

(Nom) Dataset

Result list (right-click for options):
- 21:43:22 - bayes.NaiveBayes
- 21:47:41 - trees.RandomForest
- 21:48:13 - functions.Logistic
- 21:49:24 - functions.SMO
- 21:49:57 - lazy.IBk
- 21:51:06 - bayes.NaiveBayes
- 21:51:24 - functions.Logistic
- 21:52:08 - trees.RandomForest
- 21:52:32 - functions.SMO

```
 +      -0.0823 * (normalized) 0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phosphotase-0.132Gender=Male+0.121Age...=3.888264
 +      -0.0823 * (normalized) 0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phosphotase-0.132Gender=Male+0.121Age...=4.05393
 +      -0.0821 * (normalized) 0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phosphotase-0.132Gender=Male+0.121Age...=4.103134
 -       0.4243

Number of kernel evaluations: 196833 (90.219% cached)



Time taken to build model: 1.21 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         589               78.5333 %
Incorrectly Classified Instances       161               21.4667 %
Kappa statistic                          0.5438
Mean absolute error                      0.2147
Root mean squared error                  0.4633
Relative absolute error                 43.4504 %
Root relative squared error             93.2212 %
Total Number of Instances              750

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.482    0.721      1.000    0.838      0.611    0.759     0.721     1
                 0.518    0.000    1.000      0.518    0.682      0.611    0.759     0.733     2
Weighted Avg.    0.785    0.267    0.845      0.785    0.769      0.611    0.759     0.726

=== Confusion Matrix ===

   a   b   <-- classified as
 416   0 |   a = 1
 161 173 |   b = 2
```

PCA Without SMOTE: 72.3842 %



Classifier: SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num-decimal-places 4"

Test options: ○ Use training set ○ Supplied test set ● Cross-validation Folds 10 ○ Percentage split % 66

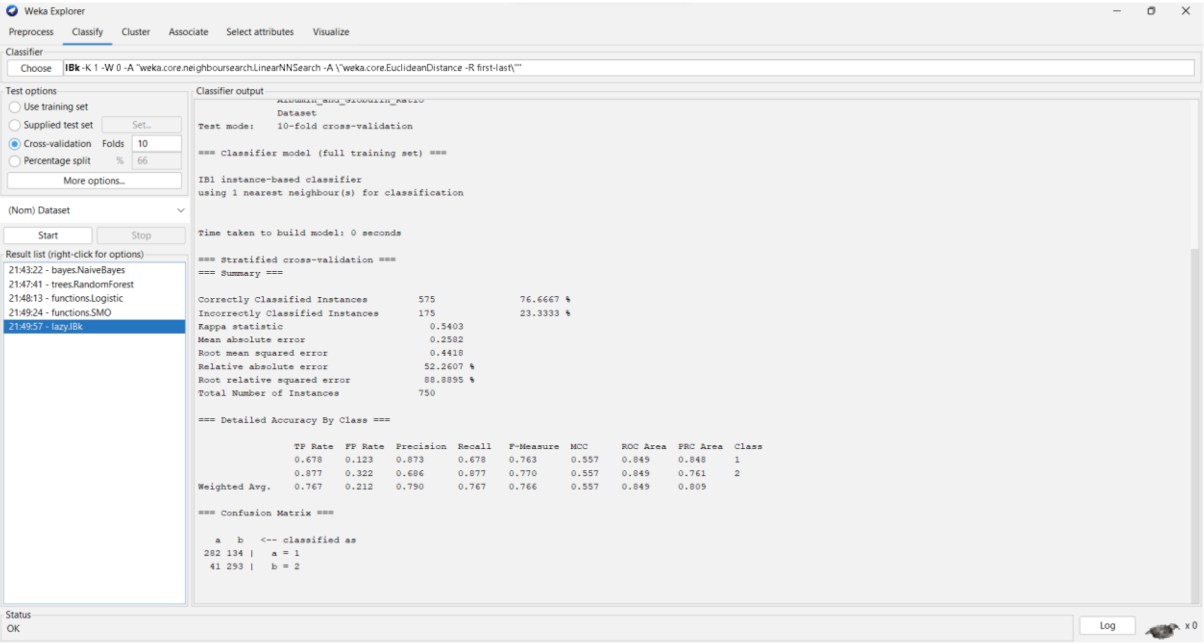
(Nom) Dataset

Result list (right-click for options):
- 23:32:16 - bayes.NaiveBayes
- 23:33:19 - functions.Logistic
- 23:52:32 - functions.SMO

```
 +      -0.0822 * (normalized) 0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phosphotase-0.132Gender=Male+0.121Age...=3.888264
 +      -0.0823 * (normalized) 0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phosphotase-0.132Gender=Male+0.121Age...=4.05393
 +      -0.0822 * (normalized) 0.735Albumin_and_Globulin_Ratio-0.507Total_Protiens+0.378Alkaline_Phosphotase-0.132Gender=Male+0.121Age...=4.103134
 -       0.4244

Number of kernel evaluations: 169845 (88.967% cached)



Time taken to build model: 1.46 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         422               72.3842 %
Incorrectly Classified Instances       161               27.6158 %
Kappa statistic                          0.0505
Mean absolute error                      0.2762
Root mean squared error                  0.5255
Relative absolute error                 67.4956 %
Root relative squared error            116.2328 %
Total Number of Instances              583

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.964    0.721      1.000    0.838      0.161    0.518     0.721     1
                 0.036    0.000    1.000      0.036    0.069      0.161    0.518     0.312     2
Weighted Avg.    0.724    0.688    0.801      0.724    0.618      0.161    0.518     0.604

=== Confusion Matrix ===

   a   b   <-- classified as
 416   0 |   a = 1
 161   6 |   b = 2
```

5. KNN:

With SMOTE: 76.6667 %



Classifier: IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

Test options: ● Cross-validation Folds 10

(Nom) Dataset

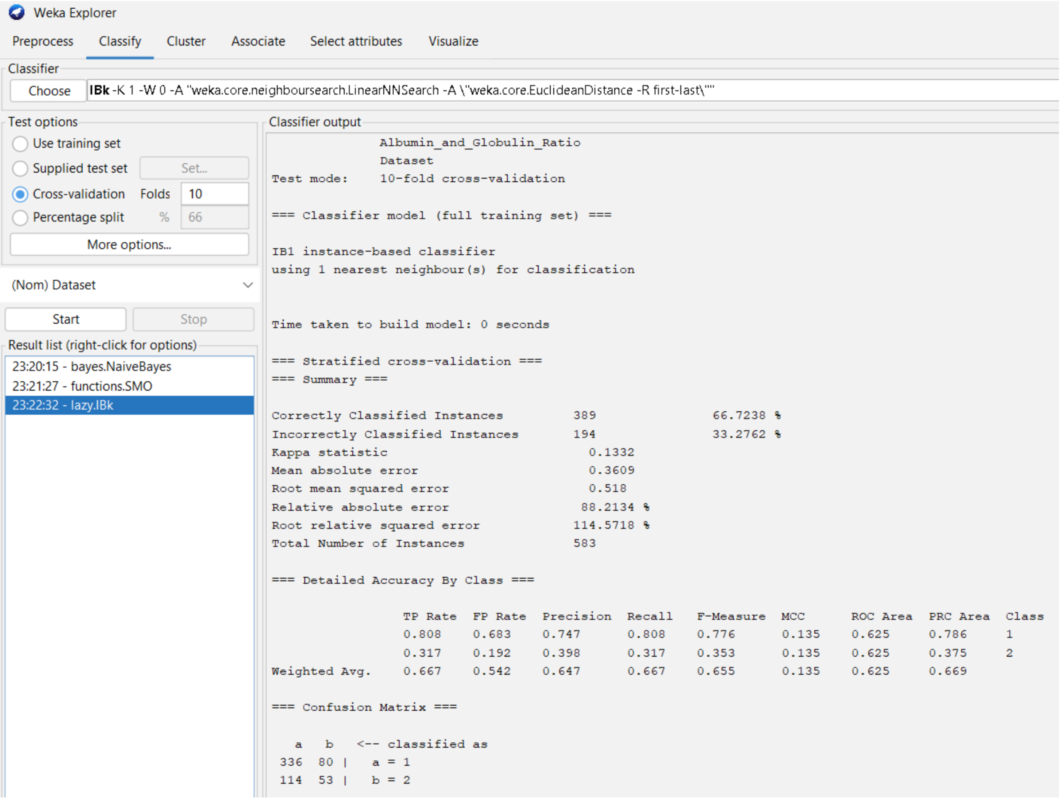
Result list (right-click for options)
- 21:43:22 - bayes.NaiveBayes
- 21:47:41 - trees.RandomForest
- 21:48:13 - functions.Logistic
- 21:49:24 - functions.SMO
- 21:49:57 - lazy.IBk

Classifier output

```
               Albumin_and_Globulin_Ratio
               Dataset
Test mode:     10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         575               76.6667 %
Incorrectly Classified Instances       175               23.3333 %
Kappa statistic                          0.5403
Mean absolute error                      0.2582
Root mean squared error                  0.4418
Relative absolute error                 52.2607 %
Root relative squared error             88.8895 %
Total Number of Instances              750

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.678    0.123    0.873      0.678    0.763      0.557    0.849     0.848     1
                 0.877    0.322    0.686      0.877    0.770      0.557    0.849     0.761     2
Weighted Avg.    0.767    0.212    0.790      0.767    0.766      0.557    0.849     0.809

=== Confusion Matrix ===

   a   b   <-- classified as
 282 134 |   a = 1
  41 293 |   b = 2
```

Status
OK

Without SMOTE: 66.7238%



Classifier: IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

Test options: ● Cross-validation Folds 10

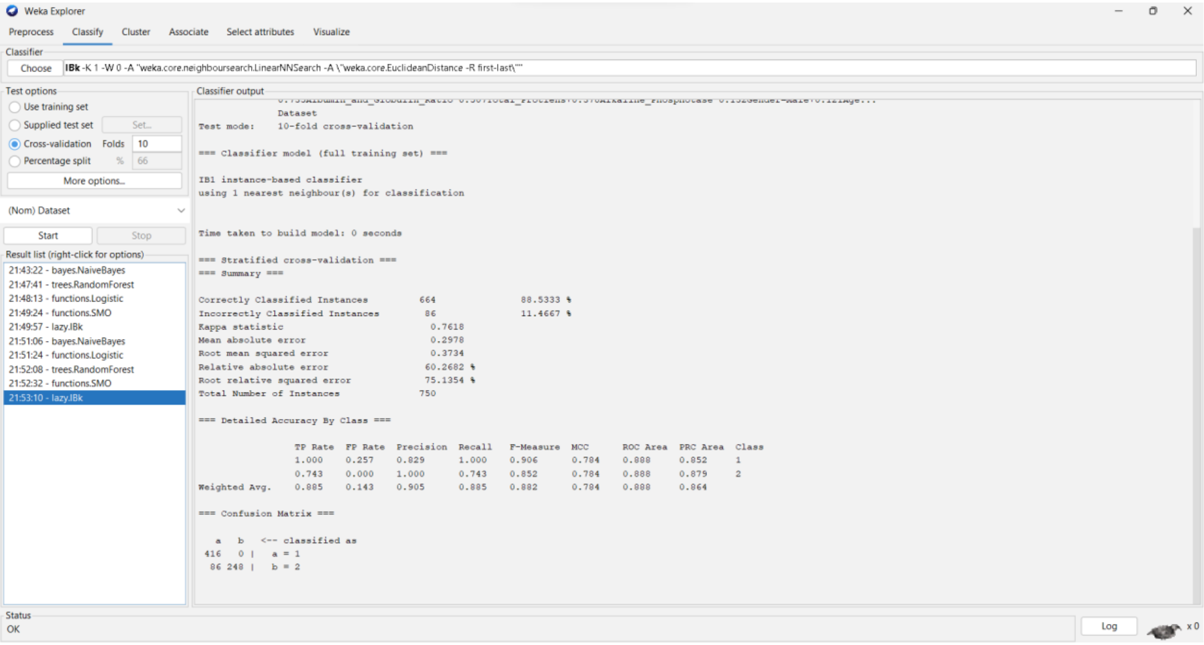
(Nom) Dataset

Result list (right-click for options)
- 23:20:15 - bayes.NaiveBayes
- 23:21:27 - functions.SMO
- 23:22:32 - lazy.IBk

Classifier output

```
               Albumin_and_Globulin_Ratio
               Dataset
Test mode:     10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         389               66.7238 %
Incorrectly Classified Instances       194               33.2762 %
Kappa statistic                          0.1332
Mean absolute error                      0.3609
Root mean squared error                  0.518
Relative absolute error                 88.2134 %
Root relative squared error            114.5718 %
Total Number of Instances              583

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.808    0.683    0.747      0.808    0.776      0.135    0.625     0.786     1
                 0.317    0.192    0.398      0.317    0.353      0.135    0.625     0.375     2
Weighted Avg.    0.667    0.542    0.647      0.667    0.655      0.135    0.625     0.669

=== Confusion Matrix ===

   a   b   <-- classified as
 336  80 |   a = 1
 114  53 |   b = 2
```

PCA With SMOTE: 88.5333 %



IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

Test options

○ Use training set

○ Supplied test set Set...

● Cross-validation Folds 10

○ Percentage split % 66

Result list (right-click for options)

- 21:43:22 - bayes.NaiveBayes
- 21:47:41 - trees.RandomForest
- 21:48:13 - functions.Logistic
- 21:49:24 - functions.SMO
- 21:49:57 - lazy.IBk
- 21:51:06 - bayes.NaiveBayes
- 21:51:24 - functions.Logistic
- 21:52:08 - trees.RandomForest
- 21:52:32 - functions.SMO
- 21:53:10 - lazy.IBk

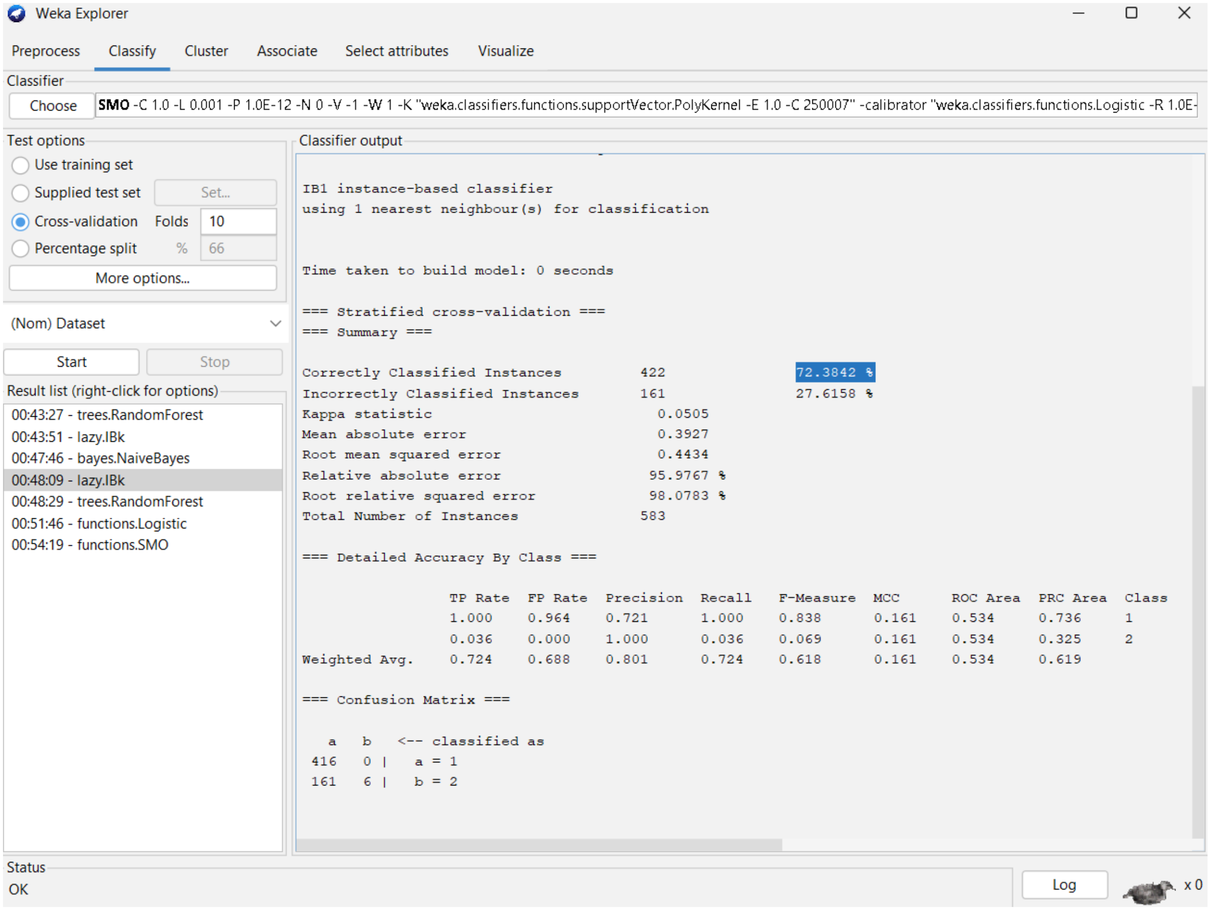
Classifier output

```
              0.735Albumin_and_Globulin_Ratio 0.507Total_Protiens+0.376Alkaline_Phosphotase 0.152Gender=Male+0.121Age...
              Dataset
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         664               88.5333 %
Incorrectly Classified Instances        86               11.4667 %
Kappa statistic                          0.7618
Mean absolute error                      0.2978
Root mean squared error                  0.3734
Relative absolute error                 60.2682 %
Root relative squared error             75.1354 %
Total Number of Instances              750

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.257    0.829      1.000    0.906      0.784    0.888     0.852     1
                 0.743    0.000    1.000      0.743    0.852      0.784    0.888     0.879     2
Weighted Avg.    0.885    0.143    0.905      0.885    0.882      0.784    0.888     0.864

=== Confusion Matrix ===

   a   b   <-- classified as
 416   0 |   a = 1
  86 248 |   b = 2
```

Status

OK

PCA Without SMOTE: 72.3842 %



SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.functions.Logistic -R 1.0E-

Test options

○ Use training set

○ Supplied test set Set...

● Cross-validation Folds 10

○ Percentage split % 66

Result list (right-click for options)

- 00:43:27 - trees.RandomForest
- 00:43:51 - lazy.IBk
- 00:47:46 - bayes.NaiveBayes
- 00:48:09 - lazy.IBk
- 00:48:29 - trees.RandomForest
- 00:51:46 - functions.Logistic
- 00:54:19 - functions.SMO

Classifier output

```
IB1 instance-based classifier
using 1 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         422               72.3842 %
Incorrectly Classified Instances       161               27.6158 %
Kappa statistic                          0.0505
Mean absolute error                      0.3927
Root mean squared error                  0.4434
Relative absolute error                 95.9767 %
Root relative squared error             98.0783 %
Total Number of Instances              583

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.964    0.721      1.000    0.838      0.161    0.534     0.736     1
                 0.036    0.000    1.000      0.036    0.069      0.161    0.534     0.325     2
Weighted Avg.    0.724    0.688    0.801      0.724    0.618      0.161    0.534     0.619

=== Confusion Matrix ===

   a   b   <-- classified as
 416   0 |   a = 1
 161   6 |   b = 2
```

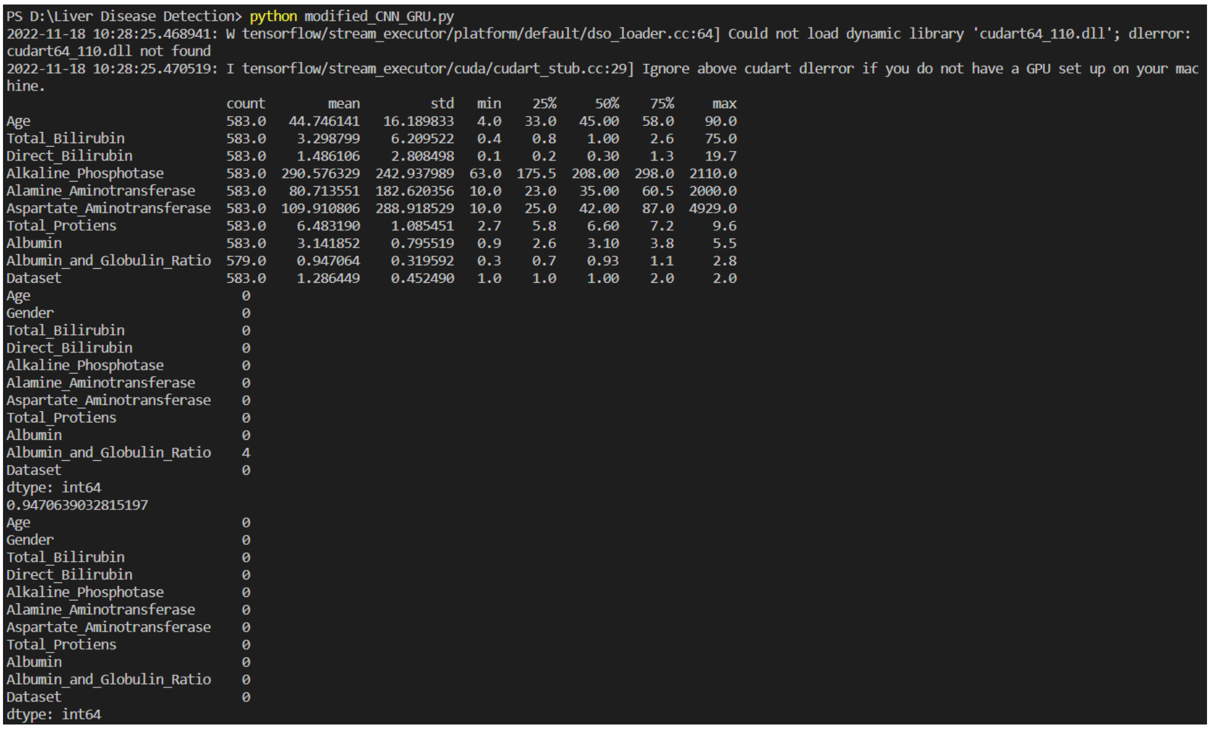
Status

OK

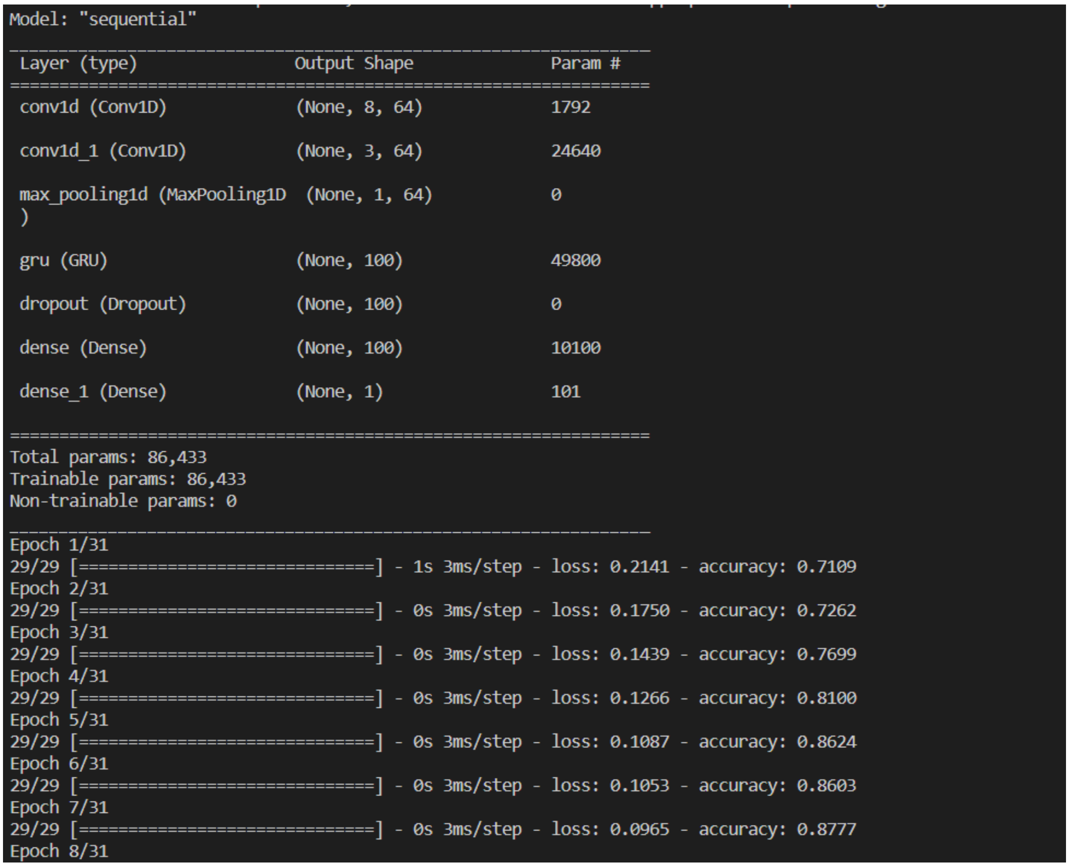
CODE IMPLEMENTATION

The following series of images show the code for the project

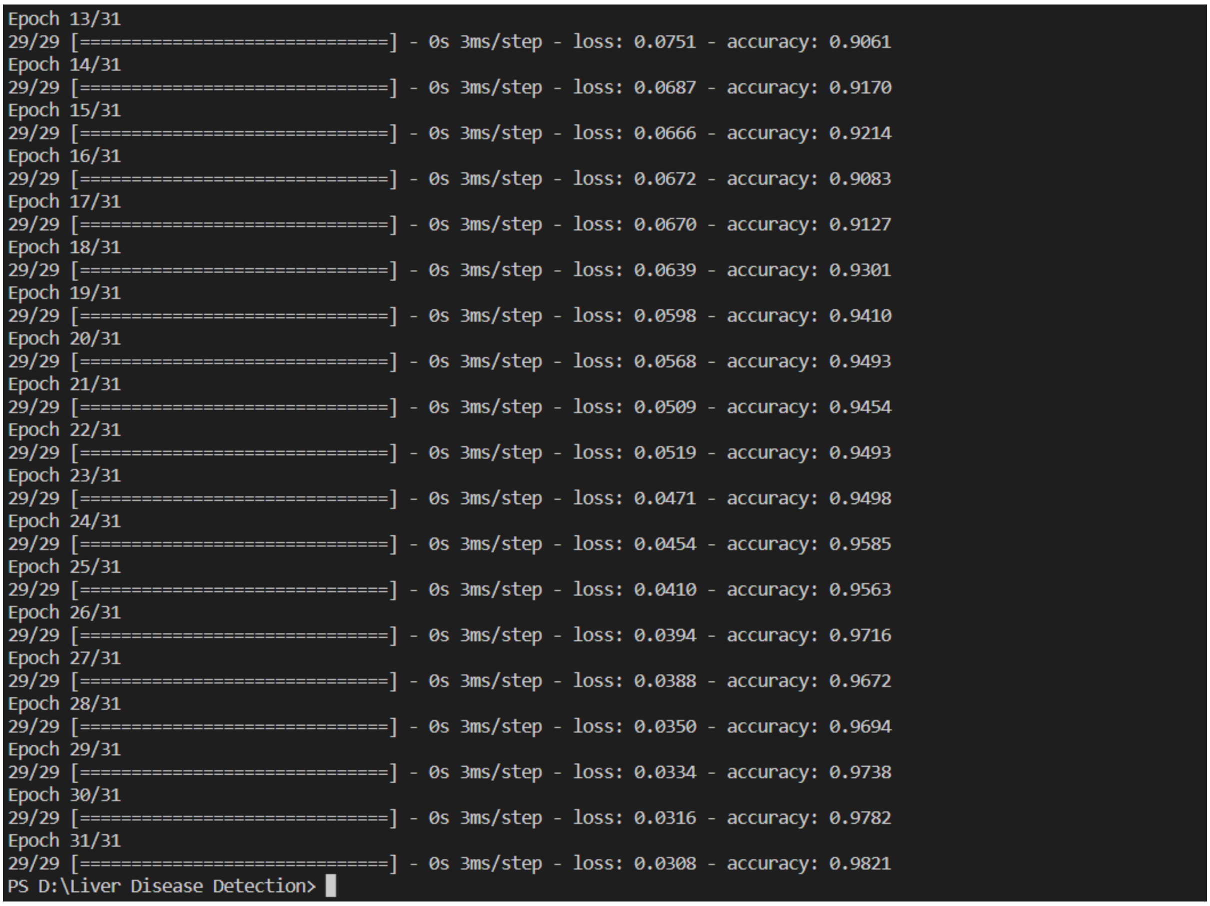
CNN + GRU



```
PS D:\Liver Disease Detection> python modified_CNN_GRU.py
2022-11-18 10:28:25.468941: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror:
cudart64_110.dll not found
2022-11-18 10:28:25.470519: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your mac
hine.
                            count        mean         std   min    25%     50%    75%     max
Age                         583.0   44.746141   16.189833   4.0   33.0   45.00   58.0    90.0
Total_Bilirubin             583.0    3.298799    6.209522   0.4    0.8    1.00    2.6    75.0
Direct_Bilirubin            583.0    1.486106    2.808498   0.1    0.2    0.30    1.3    19.7
Alkaline_Phosphotase        583.0  290.576329  242.937989  63.0  175.5  208.00  298.0  2110.0
Alamine_Aminotransferase    583.0   80.713551  182.620356  10.0   23.0   35.00   60.5  2000.0
Aspartate_Aminotransferase  583.0  109.910806  288.918529  10.0   25.0   42.00   87.0  4929.0
Total_Protiens              583.0    6.483190    1.085451   2.7    5.8    6.60    7.2     9.6
Albumin                     583.0    3.141852    0.795519   0.9    2.6    3.10    3.8     5.5
Albumin_and_Globulin_Ratio  579.0    0.947064    0.319592   0.3    0.7    0.93    1.1     2.8
Dataset                     583.0    1.286449    0.452490   1.0    1.0    1.00    2.0     2.0
Age                           0
Gender                        0
Total_Bilirubin               0
Direct_Bilirubin              0
Alkaline_Phosphotase          0
Alamine_Aminotransferase      0
Aspartate_Aminotransferase    0
Total_Protiens                0
Albumin                       0
Albumin_and_Globulin_Ratio    4
Dataset                       0
dtype: int64
0.9470639032815197
Age                           0
Gender                        0
Total_Bilirubin               0
Direct_Bilirubin              0
Alkaline_Phosphotase          0
Alamine_Aminotransferase      0
Aspartate_Aminotransferase    0
Total_Protiens                0
Albumin                       0
Albumin_and_Globulin_Ratio    0
Dataset                       0
dtype: int64
```

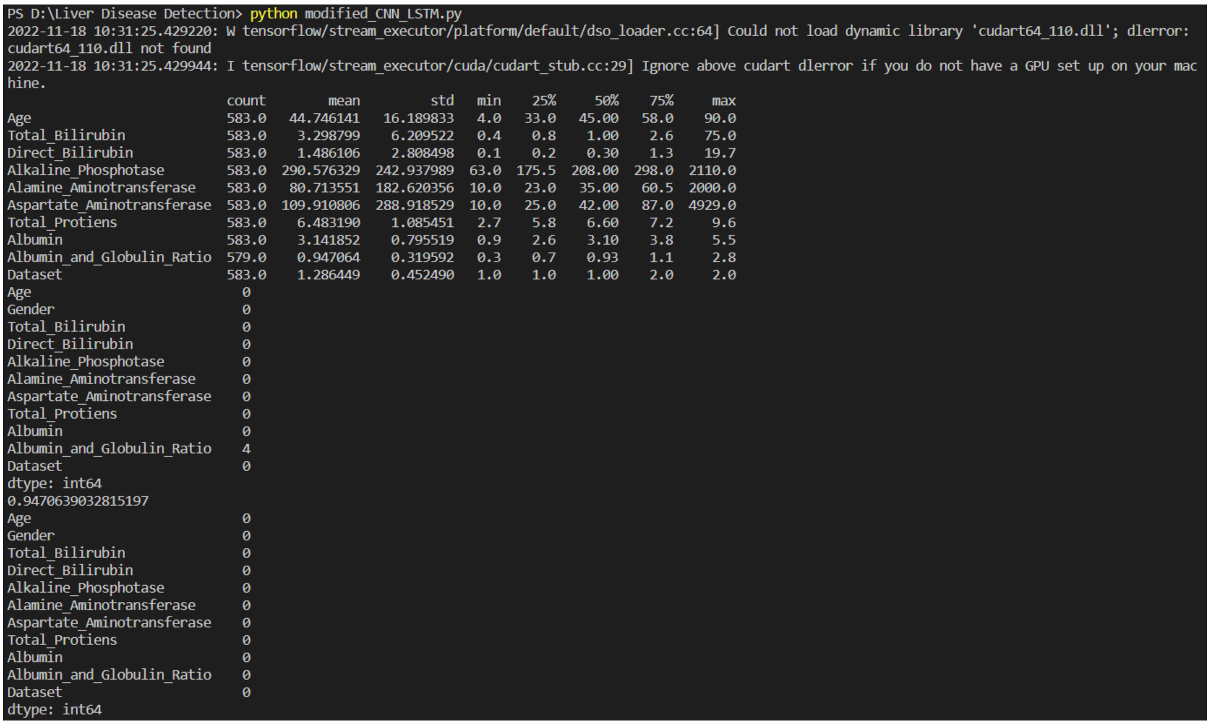


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 8, 64)             1792

 conv1d_1 (Conv1D)           (None, 3, 64)             24640

 max_pooling1d (MaxPooling1D  (None, 1, 64)            0
 )

 gru (GRU)                   (None, 100)               49800

 dropout (Dropout)           (None, 100)               0

 dense (Dense)               (None, 100)               10100

 dense_1 (Dense)             (None, 1)                 101

=================================================================
Total params: 86,433
Trainable params: 86,433
Non-trainable params: 0
_________________________________________________________________
Epoch 1/31
29/29 [==============================] - 1s 3ms/step - loss: 0.2141 - accuracy: 0.7109
Epoch 2/31
29/29 [==============================] - 0s 3ms/step - loss: 0.1750 - accuracy: 0.7262
Epoch 3/31
29/29 [==============================] - 0s 3ms/step - loss: 0.1439 - accuracy: 0.7699
Epoch 4/31
29/29 [==============================] - 0s 3ms/step - loss: 0.1266 - accuracy: 0.8100
Epoch 5/31
29/29 [==============================] - 0s 3ms/step - loss: 0.1087 - accuracy: 0.8624
Epoch 6/31
29/29 [==============================] - 0s 3ms/step - loss: 0.1053 - accuracy: 0.8603
Epoch 7/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0965 - accuracy: 0.8777
Epoch 8/31
```

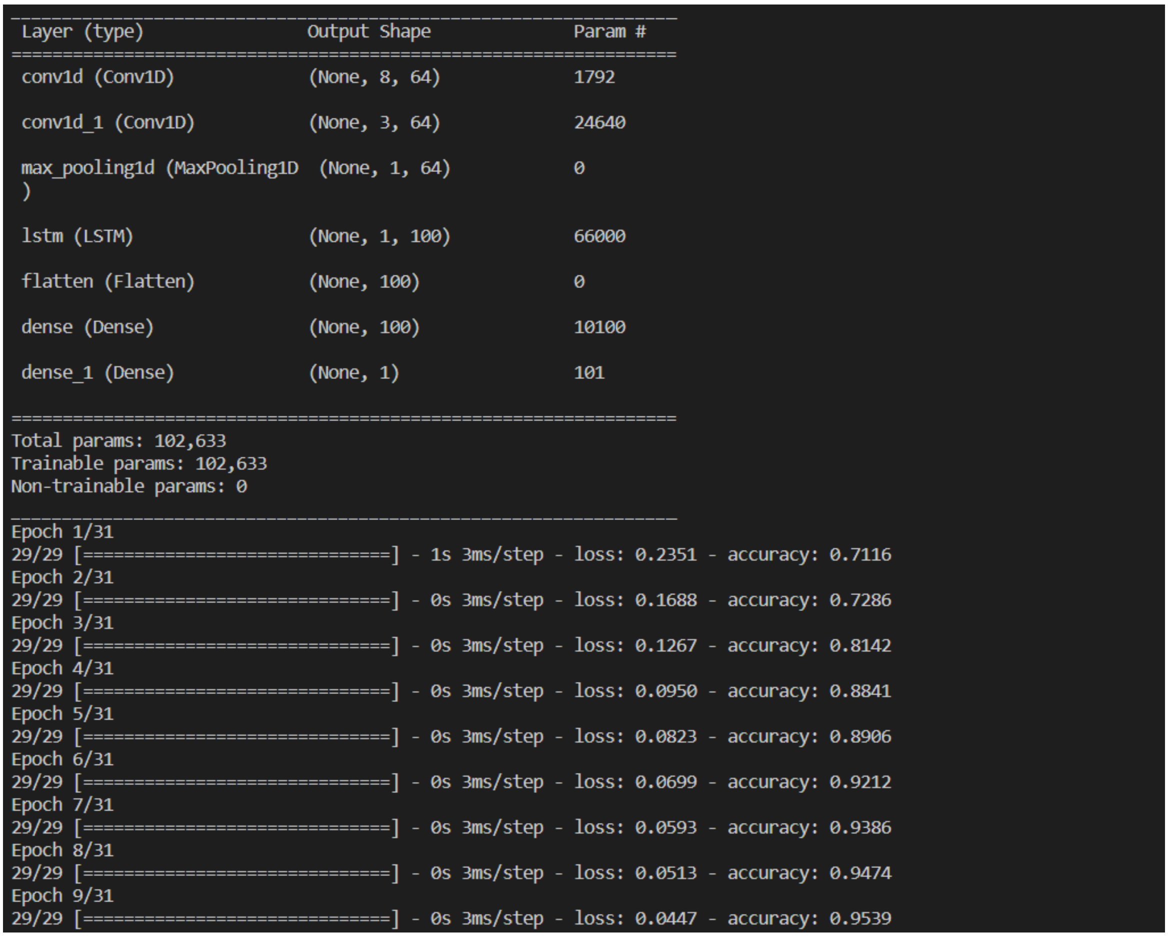


```
Epoch 13/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0751 - accuracy: 0.9061
Epoch 14/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0687 - accuracy: 0.9170
Epoch 15/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0666 - accuracy: 0.9214
Epoch 16/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0672 - accuracy: 0.9083
Epoch 17/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0670 - accuracy: 0.9127
Epoch 18/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0639 - accuracy: 0.9301
Epoch 19/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0598 - accuracy: 0.9410
Epoch 20/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0568 - accuracy: 0.9493
Epoch 21/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0509 - accuracy: 0.9454
Epoch 22/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0519 - accuracy: 0.9493
Epoch 23/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0471 - accuracy: 0.9498
Epoch 24/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0454 - accuracy: 0.9585
Epoch 25/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0410 - accuracy: 0.9563
Epoch 26/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0394 - accuracy: 0.9716
Epoch 27/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0388 - accuracy: 0.9672
Epoch 28/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0350 - accuracy: 0.9694
Epoch 29/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0334 - accuracy: 0.9738
Epoch 30/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0316 - accuracy: 0.9782
Epoch 31/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0308 - accuracy: 0.9821
PS D:\Liver Disease Detection> █
```

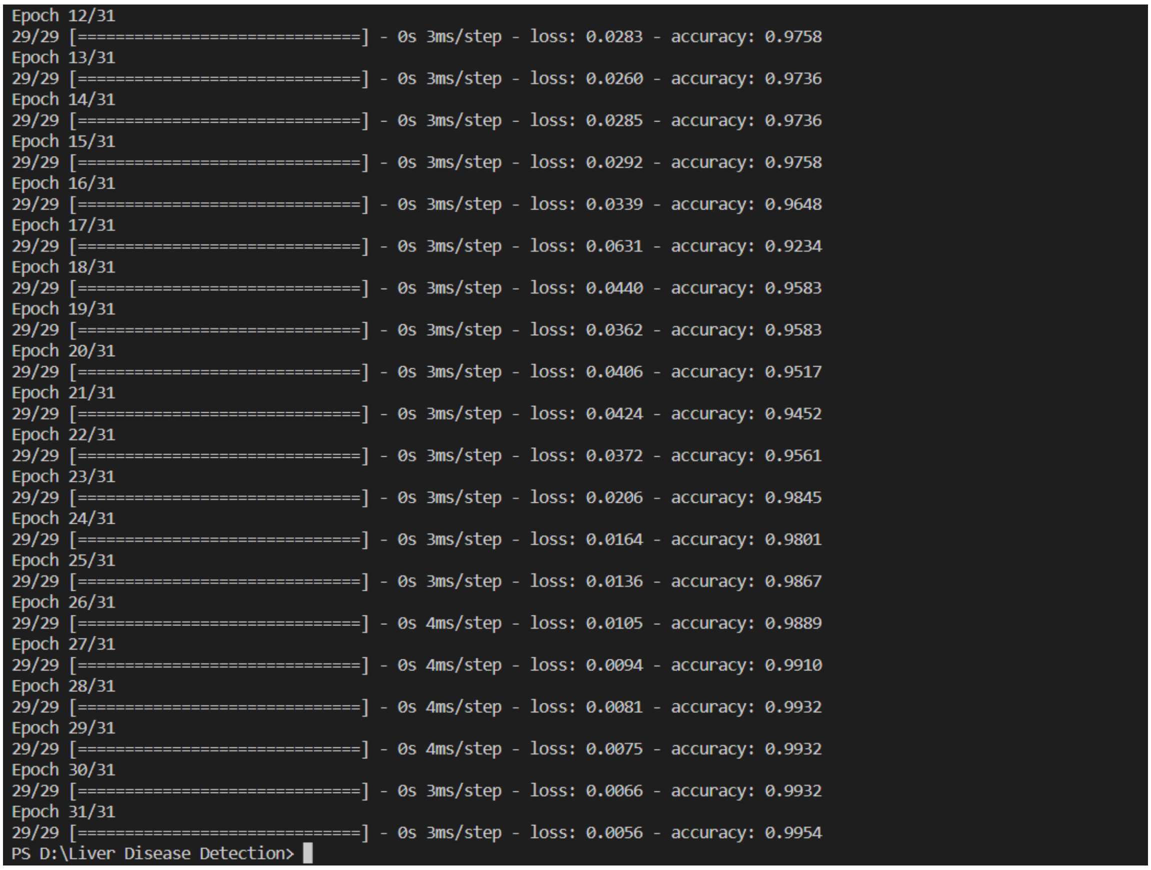
CNN + LSTM



```
PS D:\Liver Disease Detection> python modified_CNN_LSTM.py
2022-11-18 10:31:25.429220: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror:
cudart64_110.dll not found
2022-11-18 10:31:25.429944: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your mac
hine.
                            count        mean         std   min    25%     50%    75%     max
Age                         583.0   44.746141   16.189833   4.0   33.0   45.00   58.0    90.0
Total_Bilirubin             583.0    3.298799    6.209522   0.4    0.8    1.00    2.6    75.0
Direct_Bilirubin            583.0    1.486106    2.808498   0.1    0.2    0.30    1.3    19.7
Alkaline_Phosphotase        583.0  290.576329  242.937989  63.0  175.5  208.00  298.0  2110.0
Alamine_Aminotransferase    583.0   80.713551  182.620356  10.0   23.0   35.00   60.5  2000.0
Aspartate_Aminotransferase  583.0  109.910806  288.918529  10.0   25.0   42.00   87.0  4929.0
Total_Protiens              583.0    6.483190    1.085451   2.7    5.8    6.60    7.2     9.6
Albumin                     583.0    3.141852    0.795519   0.9    2.6    3.10    3.8     5.5
Albumin_and_Globulin_Ratio  579.0    0.947064    0.319592   0.3    0.7    0.93    1.1     2.8
Dataset                     583.0    1.286449    0.452490   1.0    1.0    1.00    2.0     2.0
Age                           0
Gender                        0
Total_Bilirubin               0
Direct_Bilirubin              0
Alkaline_Phosphotase          0
Alamine_Aminotransferase      0
Aspartate_Aminotransferase    0
Total_Protiens                0
Albumin                       0
Albumin_and_Globulin_Ratio    4
Dataset                       0
dtype: int64
0.9470639032815197
Age                           0
Gender                        0
Total_Bilirubin               0
Direct_Bilirubin              0
Alkaline_Phosphotase          0
Alamine_Aminotransferase      0
Aspartate_Aminotransferase    0
Total_Protiens                0
Albumin                       0
Albumin_and_Globulin_Ratio    0
Dataset                       0
dtype: int64
```

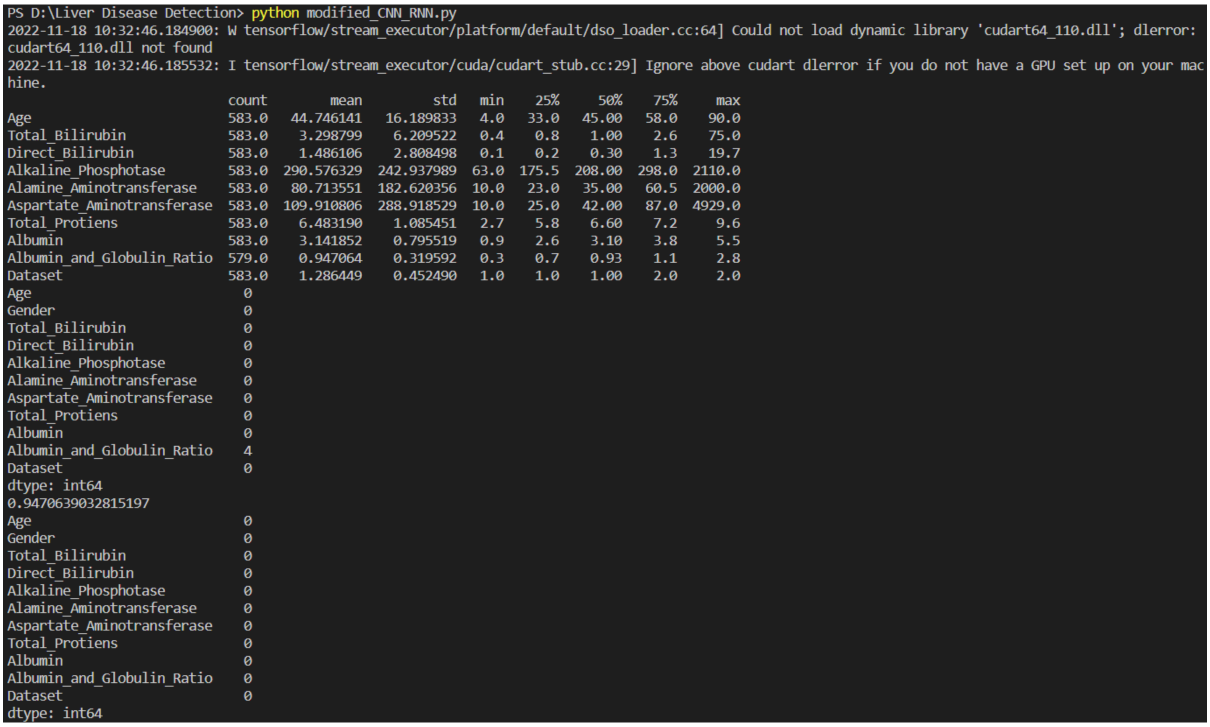


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 8, 64)             1792

 conv1d_1 (Conv1D)           (None, 3, 64)             24640

 max_pooling1d (MaxPooling1D  (None, 1, 64)            0
 )

 lstm (LSTM)                 (None, 1, 100)            66000

 flatten (Flatten)           (None, 100)               0

 dense (Dense)               (None, 100)               10100

 dense_1 (Dense)             (None, 1)                 101

=================================================================
Total params: 102,633
Trainable params: 102,633
Non-trainable params: 0
_________________________________________________________________
Epoch 1/31
29/29 [==============================] - 1s 3ms/step - loss: 0.2351 - accuracy: 0.7116
Epoch 2/31
29/29 [==============================] - 0s 3ms/step - loss: 0.1688 - accuracy: 0.7286
Epoch 3/31
29/29 [==============================] - 0s 3ms/step - loss: 0.1267 - accuracy: 0.8142
Epoch 4/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0950 - accuracy: 0.8841
Epoch 5/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0823 - accuracy: 0.8906
Epoch 6/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0699 - accuracy: 0.9212
Epoch 7/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0593 - accuracy: 0.9386
Epoch 8/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0513 - accuracy: 0.9474
Epoch 9/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0447 - accuracy: 0.9539
```

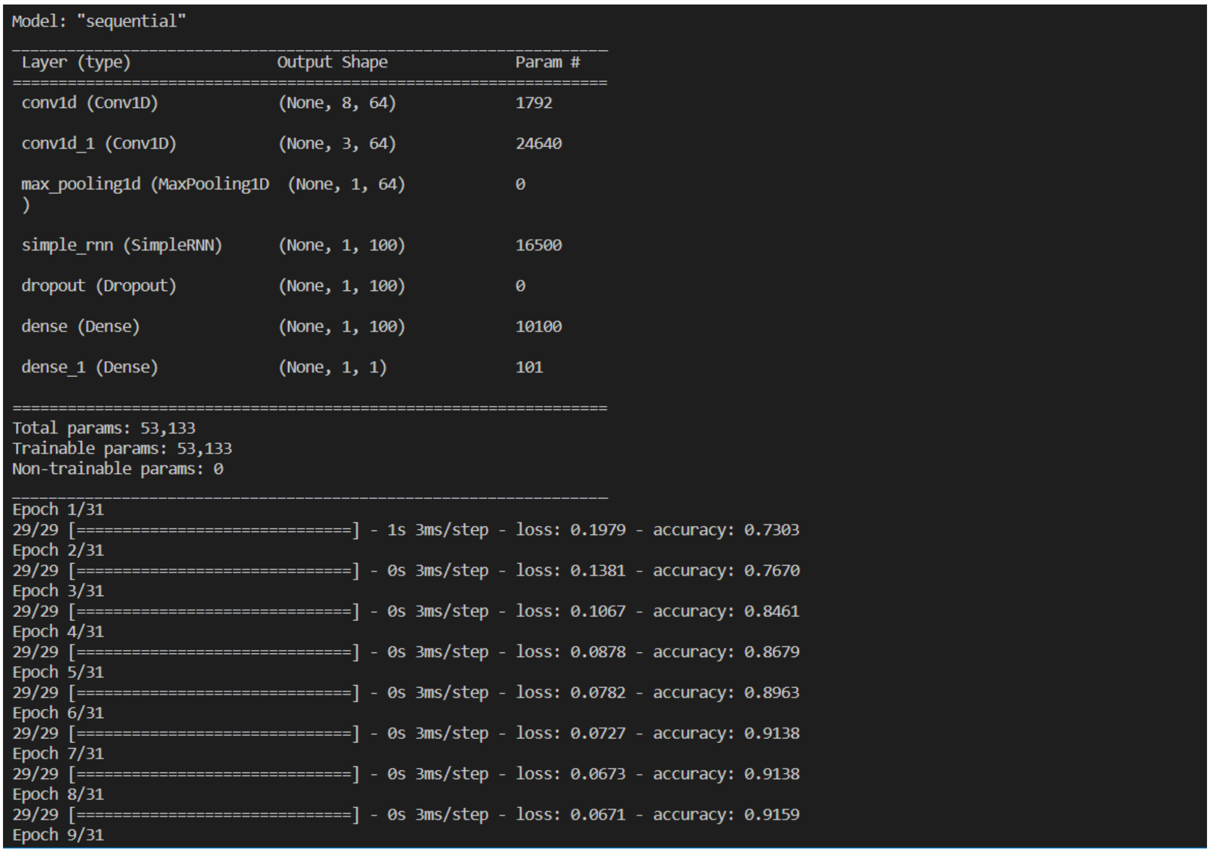


```
Epoch 12/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0283 - accuracy: 0.9758
Epoch 13/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0260 - accuracy: 0.9736
Epoch 14/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0285 - accuracy: 0.9736
Epoch 15/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0292 - accuracy: 0.9758
Epoch 16/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0339 - accuracy: 0.9648
Epoch 17/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0631 - accuracy: 0.9234
Epoch 18/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0440 - accuracy: 0.9583
Epoch 19/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0362 - accuracy: 0.9583
Epoch 20/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0406 - accuracy: 0.9517
Epoch 21/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0424 - accuracy: 0.9452
Epoch 22/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0372 - accuracy: 0.9561
Epoch 23/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0206 - accuracy: 0.9845
Epoch 24/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0164 - accuracy: 0.9801
Epoch 25/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0136 - accuracy: 0.9867
Epoch 26/31
29/29 [==============================] - 0s 4ms/step - loss: 0.0105 - accuracy: 0.9889
Epoch 27/31
29/29 [==============================] - 0s 4ms/step - loss: 0.0094 - accuracy: 0.9910
Epoch 28/31
29/29 [==============================] - 0s 4ms/step - loss: 0.0081 - accuracy: 0.9932
Epoch 29/31
29/29 [==============================] - 0s 4ms/step - loss: 0.0075 - accuracy: 0.9932
Epoch 30/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0066 - accuracy: 0.9932
Epoch 31/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0056 - accuracy: 0.9954
PS D:\Liver Disease Detection>
```

CNN + RNN



```
PS D:\Liver Disease Detection> python modified_CNN_RNN.py
2022-11-18 10:32:46.184900: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror:
cudart64_110.dll not found
2022-11-18 10:32:46.185532: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your mac
hine.
                            count        mean         std   min    25%     50%    75%     max
Age                         583.0   44.746141   16.189833   4.0   33.0   45.00   58.0    90.0
Total_Bilirubin             583.0    3.298799    6.209522   0.4    0.8    1.00    2.6    75.0
Direct_Bilirubin            583.0    1.486106    2.808498   0.1    0.2    0.30    1.3    19.7
Alkaline_Phosphotase        583.0  290.576329  242.937989  63.0  175.5  208.00  298.0  2110.0
Alamine_Aminotransferase    583.0   80.713551  182.620356  10.0   23.0   35.00   60.5  2000.0
Aspartate_Aminotransferase  583.0  109.910806  288.918529  10.0   25.0   42.00   87.0  4929.0
Total_Protiens              583.0    6.483190    1.085451   2.7    5.8    6.60    7.2     9.6
Albumin                     583.0    3.141852    0.795519   0.9    2.6    3.10    3.8     5.5
Albumin_and_Globulin_Ratio  579.0    0.947064    0.319592   0.3    0.7    0.93    1.1     2.8
Dataset                     583.0    1.286449    0.452490   1.0    1.0    1.00    2.0     2.0
Age                           0
Gender                        0
Total_Bilirubin               0
Direct_Bilirubin              0
Alkaline_Phosphotase          0
Alamine_Aminotransferase      0
Aspartate_Aminotransferase    0
Total_Protiens                0
Albumin                       0
Albumin_and_Globulin_Ratio    4
Dataset                       0
dtype: int64
0.9470639032815197
Age                           0
Gender                        0
Total_Bilirubin               0
Direct_Bilirubin              0
Alkaline_Phosphotase          0
Alamine_Aminotransferase      0
Aspartate_Aminotransferase    0
Total_Protiens                0
Albumin                       0
Albumin_and_Globulin_Ratio    0
Dataset                       0
dtype: int64
```



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 8, 64)             1792

 conv1d_1 (Conv1D)           (None, 3, 64)             24640

 max_pooling1d (MaxPooling1D  (None, 1, 64)            0
 )

 simple_rnn (SimpleRNN)      (None, 1, 100)            16500

 dropout (Dropout)           (None, 1, 100)            0

 dense (Dense)               (None, 1, 100)            10100

 dense_1 (Dense)             (None, 1, 1)              101

=================================================================
Total params: 53,133
Trainable params: 53,133
Non-trainable params: 0
_________________________________________________________________
Epoch 1/31
29/29 [==============================] - 1s 3ms/step - loss: 0.1979 - accuracy: 0.7303
Epoch 2/31
29/29 [==============================] - 0s 3ms/step - loss: 0.1381 - accuracy: 0.7670
Epoch 3/31
29/29 [==============================] - 0s 3ms/step - loss: 0.1067 - accuracy: 0.8461
Epoch 4/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0878 - accuracy: 0.8679
Epoch 5/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0782 - accuracy: 0.8963
Epoch 6/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0727 - accuracy: 0.9138
Epoch 7/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0673 - accuracy: 0.9138
Epoch 8/31
29/29 [==============================] - 0s 3ms/step - loss: 0.0671 - accuracy: 0.9159
Epoch 9/31
```

VII. COMPARATIVE STUDY / RESULTS AND DISCUSSION

In the existing method[ref. 18] the researchers have used MLCNN-LDPS which provided an accuracy of 90.75%. We have used three hybrid algorithms: CNN+LSTM(99.54%), CNN+GRU(98.23%), CNN+RNN(99.45%) and have achieved an accuracy as high as 99.48% using filters like upscaling and PCA. We also got used various algorithms and got the following accuracies- naive bayes: 76%, random forest: 80.26%, logistic: 72%, svm: 76.93%, knn: 76.67%.

Accuracy table

Table 3: The table contains the accuracy of all the algorithms run on weka before and after applying pca and smote

	With SMOTE	Without SMOTE	PCA With SMOTE	PCA Without SMOTE
Naïve Bayes	76 %	62.6072%	47.4667%	31.38%
Random Forest	80.2667 %	72.0412%	84.2667 %	72.3842 %
Logistic	72%	60.2058%	88.5333 %	72.3842 %
SVM	76.9333 %	64.494%	78.5333 %	72.3842 %
KNN	76.6667 %	66.7238%	88.5333 %	72.3842 %

VIII. CONCLUSION AND FUTURE WORK

With the help of performance measure and analysis the performance of various deep algorithms are evaluated. The dataset was obtained from UCI repository on which data pre-processing techniques such as filling missing values, replacing duplicate values was performed. Oversampling was performed using SMOTE and with the help of data visualization the model was trained to understand the duplicate values present. Feature selection showed a linear relationship on certain attributes of the dataset. The highest accuracy was obtained by using CNN+RNN model and thus the performance was measured based on a classification report and performance measures such as accuracy and precision.

This result which we obtained relies upon various deep learning algorithms which provides high accuracy and consumes very less time for the entire processing. The process includes data analysis, data pre-processing which includes filling of missing values with mean, identifying duplicate value, and resampling to improve the performance. Accuracy is effectively utilized to analyse the performance of various classification algorithms. Thus, we can conclude that CNN+RNN model proved its worthiness in prediction of liver patients by achieving high accuracy amongst the other hybrid algorithms.

IX. REFERENCES.

[1]. Jeyalakshmi, K., & Rangaraj, R. (2021). Accurate liver disease prediction system using convolutional neural network. *Indian Journal of Science and Technology*, *14*(17), 1406-1421. (Base Paper)

[2]. Mutlu, E. N., Devim, A., Hameed, A. A., & Jamil, A. (2022). Deep Learning for Liver Disease Prediction. In *Mediterranean Conference on Pattern Recognition and Artificial Intelligence* (pp. 95-107). Springer, Cham.

[3]. Nahar, N., & Ara, F. (2018). Liver disease prediction by using different decision tree techniques. *International Journal of Data Mining & Knowledge Management Process*, *8*(2), 01-09.

[4]. Kefelegn, S., & Kamat, P. (2018). Prediction and analysis of liver disorder diseases by using data mining technique: survey. *International Journal of pure and applied mathematics*, *118*(9), 765-770.

[5]. Nahar, N., Ara, F., Neloy, M. A. I., Barua, V., Hossain, M. S., & Andersson, K. (2019, December). A comparative analysis of the ensemble method for liver disease prediction. In *2019 2nd International Conference on Innovation in Engineering and Technology (ICIET)* (pp. 1-6). IEEE.