Comparative Analysis of Machine Learning Techniques for Liver Disease Patients

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

MARIA ALEX KUZHIPPALLIL

16BCE2190

CAROLYN JOSEPH

16BCB0098

Under the guidance of

Dr. Kannan A

School of Computer Science and Engineering

VIT, Vellore.

VIT®

Vellore Institute of Technology

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

May, 2020

DECLARATION

I hereby declare that the thesis entitled *Comparative Analysis of Machine Learning Techniques for Liver Disease Patients* submitted by me, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering* to VIT is a record of bonafide work carried out by me under the supervision of Dr. Kannan A.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date : 27th May 2020

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled Comparative Analysis of Machine Learning Techniques for Liver Disease Patients submitted by **Maria Alex Kuzhippallil (16BCE2190) & Carolyn Joseph (16BCB0098)**, **School of Computer Science and Engineering**, **VIT**, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*, is a record of bonafide work carried out by him / her under my supervision during the period, 01. 12. 2018 to 30.04.2019, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Kannan A

Digitally signed by Kannan A
DN: cn=Kannan A, o=VIT, ou, email=kannan.a@vit.ac.in, c=IN
Date: 2020.06.02 12:50:42 +05'30'

Place : Vellore

Date : 27th May 2020

Signature of the Guide

Internal Examiner **External Examiner**

Dr. V. Santhi
School of Computer Science and Engineering

ICACCS 2020

SPONSORED BY AICTE

Event by & at :

Sri Eshwar College of Engineering
An Autonomous Institution
Coimbatore, India

6th 2020 International Conference on
Advanced Computing & Communication Systems

TECHNICAL SPONSORS IEEE IEEE Madras Section

Certificate of Presentation

Certify that

Mr./Ms./Dr. **Maria Alex Kuzhippallil**

Vellore Institute of Technology, Vellore, India

has presented a paper in the International Conference on

Advanced Computing & Communication Systems - ICACCS 2020

on 6th & 7th March 2020 at Sri Eshwar College of Engineering, Coimbatore, TamilNadu, India.

Paper Title :

Comparative Analysis of Machine Learning Techniques for Indian Liver Disease Patients

Dr. H. Anandakumar	Dr. R. Subha	Dr. Sudha Mohanram
Conference Chair	Convener	Patron

ICACCS

ICACCS 2020

SPONSORED BY AICTE

Event by & at :

Sri Eshwar College of Engineering
An Autonomous Institution
Coimbatore, India

6th 2020 International Conference on
Advanced Computing & Communication Systems

TECHNICAL SPONSORS IEEE IEEE Madras Section

Certificate of Presentation

Certify that

Mr./Ms./Dr. **Carolyn Joseph**

Vellore Institute of Technology, Vellore, India

has presented a paper in the International Conference on

Advanced Computing & Communication Systems - ICACCS 2020

on 6th & 7th March 2020 at Sri Eshwar College of Engineering, Coimbatore, TamilNadu, India.

Paper Title :

Comparative Analysis of Machine Learning Techniques for Indian Liver Disease Patients

Dr. H. Anandakumar	Dr. R. Subha	Dr. Sudha Mohanram
Conference Chair	Convener	Patron

ICACCS

ACKNOWLEDGEMENTS

We would like to take the opportunity to thank all the people who helped us. Firstly, we would like to express our sincere gratitude to Dr. Kannan A, Senior Professor, SCOPE, Vellore Institute of Technology, for his valuable guidance, his continuous support and understanding throughout the duration of the project. We are highly indebted for his constant supervision and his knowledge in regards of the field. Working with him was proved to be a very good opportunity for both of us.

We would also like to thank the teaching and non-teaching staff of Vellore Institute of Technology for their non-self-centred enthusiasm and provided and environment to work in which further prompted us to complete the project successfully.

Finally, we would also like to thank our family and friends for their constant support and help for the successful completion of the project.

Maria Alex Kuzhippallil – 16BCE2190

Carolyn Joseph 16BCB0098

Executive Summary

This project proposes a classifier to identify the liver disease at a preliminary stage of Indian liver disease patients. The proposed feature selection is an extension of XGBoost classifier with genetic algorithm.Furthermore, various other classification models and visualization models are compared against each other to find better outcome.

In the project, different data cleaning algorithms are incorporated such as imputation of missing values with median, label encoding to convert categorical into numerical data for easy analysis, duplicate value elimination and outlier detection is used to find out the extreme deviating values and they are eliminated using isolation forest.

The performance of the model proposed in this project is measured in terms of accuracy, precision, recall f-measure and time complexity. The results of various classifiers are obtained by using proposed feature selection algorithm. From the experiments and comparative analysis, it increases classification accuracy and also leads to reduction in classification time and hence aids in the prediction of the disease more efficiently. The results are further depicted with proper tables.

In the project we also made a web application to predict liver disease were the patients could check their chance of getting a liver disease along with the feature of blog where patients can create posts regarding information about liver diseases and prevention ways.

CONTENTS **Page No.**

List of Figures

List of Tables

List of Abbreviations

ML	Machine Learning
AI	Artificial Intelligence
KNN	K Nearest Neighbors
XGBoost	eXtreme Gradient Boosting
DBSCAN	Density based spatial clustering of Applications
WEKA	Waikato Environment for Knowledge Analysis

1. INTRODUCTION

1.1. Objective

1.1.1. *Phase I – Research and Data Analytics*

- Research on the potential of various machine learning concepts and the effective use for prediction of liver disease.
- Performing data visualisation, data cleaning, and data analysis on Indian liver disease data.
- Due to presence of ample amount of outliers, an effective outlier elimination technique has to be applied (Isolation forest).
- Performing improved feature selection technique that combines genetic algorithm with XGBoost.
- Comparing the performance of various algorithms like Neural Networks, KNN, Logistic Regression, Decision Tree, Random Forest Tree, Gradient Boosting, AdaBoost, XGBoost, Light GBM and Stacking Estimator.
- Final aim in phase I is to find the best result in terms of accuracy, precision, recall, f-measure and time complexity among these algorithms.

1.1.2. *Phase II - Web Application development*

- Developing web application using Django framework for liver disease prediction and displays the whether there is risk of liver disease.
- Creating an interactive blog component to facilitate the members to provide comments, share thoughts and suggestion for improving the portal

1.2. Motivation

Liver is vital organ with functionalities like production of bile, detoxification of chemicals and production of important proteins for blood clotting.

Long term drinking habits has been directly linked to the increased risk of having different liver diseases which may further lead to death which can be prevented if the disease is detected early. Fatty liver infiltration is the initial stage and Cirrhosis is the final stage in most chronic liver diseases which may further lead to liver cancer. Many data mining techniques or Medical Data Mining (MDM) techniques help in the detection and predict the presence of liver diseases in the early stage itself and reduce work of doctors to some extent.

1.3. Background/ Literature survey

Machine Learning (ML) a part of Artificial Intelligence (AI) allows the system to obtain knowledge with no explicit knowledge. Supervised algorithms make use of human inputs and outputs for training process and prediction accuracy, and thus used for different classification applications.

Therefore, the application of ML has extended to healthcare as well. One of the major problems in healthcare is the rising number of liver disease patients.

Data mining techniques have widely been used for the prediction of various diseases.

In [6], the authors describe the use of classification algorithms like Decision tree algorithm,Bayes Algorithm and Rule based Algorithm for diabetes disease prediction and they are considered popular classification algorithms at the time. To improve the effectiveness of classification algorithm feature extraction is used.

In [7], various classification algorithms namely Support Vector Machine, K – Nearest Neighbour and Logistic Regression have been used for the prediction of liver disease. On the bases of sensitivity the latter classification algorithm has proved to be more appropriate for the prediction of the disease. Also in [8], the authors explain various unsupervised classification algorithms for the classification of the dataset. The three techniques used in the paper are K-means, DBSCAN and Affinity Propagation. The proposed technique is divided into three stages namely analysis, prediction and comparison. The analysis is being done using K-means, Affinity Propagation and DBSCAN. The data set used contains various levels of enzymes present in the liver system. In order to find the best, performance of the techniques is implemented and is calculated using mainly Silhouette Coefficient. This factor determines accuracy and number of cluster which determines complexity. Finally K-means is found to be the optimal method in comparison to other. The paper further points to future work in determination of other diseases like heart, lung, brain etc.

In [9], the authors implemented genetic algorithm for feature extraction and compared against classification algorithms like Naïve Bayes, Nearest Neighbours and Support Vector Machines. The approach of using feature extraction successfully improved classification and achiever a higher accuracy in classification. Another way to increase the performance and provide better result is the use of hybrid algorithms.

In [10], the authors provide a hybrid algorithm which makes uses of clustering and decision tree induction for the classification where the dataset is first divided into clusters and classified using decision tree. This approach when tested on real life dataset proved to show better result and improved accuracy. Another approach is shown in [11], where the authors propose the use of genetic algorithm along with K-means algorithm for the classification of the dataset where genetic algorithm is used to clean the data to improve the initial cluster centre for the K-means algorithm. And the results proved to be more accurate than just K-means algorithm. Some research paper like in [12], the authors made use of an open source tool called WEKA. WEKA is a tool used to perform data mining work. It is developed at the university in New Zealand. The dataset of sample of 20 patients is collected from BUPA research lab. The dataset has seven attributes which are taken into account. The main criteria set as a parameter is the alcohol consumption. After the use of Bayes theorem into the final result it is found that the alcohol consumption causes more likeliness for liver cancer. In the paper the author establishes the fact that the Bayes model gives the exact prediction. Sometimes the outliers can affect the performance of the algorithms. Therefore it is necessary to detect them and discard if possible.

In [13], the authors explain about outliers detection is essential component of data mining. Usually algorithms used for outlier detection includes depth based, statistical based and cluster based algorithms. This paper clearly highlights recall for various algorithms and the best performance algorithms include Pruning based KNN, KNN and Local Outlier Factor Method.

Already existing models in order to classify the medical dataset is limited. With the rising advancements of data mining and analytics makes it necessary to extend the applications to healthcare dataset. The current researches used in the healthcare data classification have low accuracy ranging in 70s. Hybrid models have proved to improve the classification models for text classification datasets. But these algorithms haven't yet been used in the medical dataset. With the rise in patients suffering from various diseases, it is important to use accurate classification algorithms. These algorithms if proves efficient can reduce the work of the doctors to some extent.

2. PROJECT DESCRIPTION AND GOALS

2.1. INTRODUCTION

2.2. REQUIREMENT ANALYSIS

2.2.1. FUNCTIONAL REQUIREMENTS

2.2.1.1.Product Perspective

The application is based on a client server model and is intended to be launched as a free service. The objective of this web application is help people identify the presence or absence of liver disease. The web interface serves as the access point for all users. The web framework used for prediction is Django which is an open source python web framework used for rapid development.

Data analytics is used to understand Indian liver patient data (ILPD). Various processes are performed to analyze and understand the data. Data visualization, preprocessing, outlier detection and elimination using isolation forest, feature selection using genetic algorithm, classification selection algorithm and measure performance. Data analytics is performed using Jupyter Notebook.

2.2.1.2.Product features

2.2.1.2.1. Data Analytics Features (Phase I)

- Outlier are detected and eliminated using Isolation forest algorithm which improves the overall accuracy of the algorithms.
- Feature selection of the liver data is implemented using combination of XGBoost Algorithm and Genetic Algorithm
- Diverse classification algorithm used to implement are Logistic Regression, KNN, Decision tree, Random Forest Tree, Gradient Boosting Algorithm, XGBoost, LightGBM and stacking estimator.
- Estimating the overall performance of algorithms using measures such as accuracy, precision, recall, f-measure and time complexity.

2.2.1.2.2. Web Framework Features (Phase II)

- Prediction – Users can predict the presence or absence of liver disease based on few attribute values such 'Age', 'Gender', 'TotalBilirubin',

DirectBilirubin','AlkalinePhosphotase','AlamineAminotransferase','AspartateAminotransferase', 'TotalProteins', 'Albumin' and 'AlbuminGlobulinRatio.

- Blogging – Users can share their views and opinions about the disease or how to overcome or handle it.

2.2.1.3.User characteristics

There is only one primary use case target user. Their main objective is to perform predictions of the disease and using the blog concept to share their views and opinions. No specific technical expertise is required for using this application. Users must have access to most basic version of a computer and its related accessories such as keyboard, mouse etc.

2.2.1.4.Assumption & Dependencies

Liver disease web application is dependent on the data collected from machine learning repository (Indian Liver Disease Patient Data) and dependent on machine learning algorithms. Hence it's essential to use the algorithms which yields results with least processing time and better performance.

2.2.2. NON FUNCTIONAL REQUIREMENTS

2.2.2.1.Product Requirements

2.2.2.1.1. Efficiency (in terms of Time and Space)

The proposed algorithm i.e genetic algorithm with XGBoost has provided varying results for different algorithms. For some algorithms the result is proved to be efficient i.e reduces time complexity. While for others, there isn't much of a time difference. Overall the best algorithm for the result has very less time consumption.

Table 1

Time taken by various algorithm before and after data cleaning and feature selection

Algorithms	Before	After
Multilayer Perceptron	2.62	0.00099
KNN	0.01	0.0099
Logistic Regression	0.15	0.001
Decision Tree	0.007	0.0017
Random Forest Tree	0.374	0.017

Gradient Boosting	0.101	0.0019
AdaBoost	0.697	0.0488
XGBoost	0.506	0.5067
LightGBM	0.238	0.0068
Stacking Estimator	0.473	0.473

2.2.2.1.2. Reliability

The proposed model has good performance matrix. The model has improved the already existing classification models to some extent. Algorithms like isolation forest and genetic algorithm has been used to provide better performance. And the results from this model can be considered more reliable than results from the pre-existing models of classical classification algorithms. Therefore, this model is applicable for healthcare data.

2.2.2.1.3. Portability

For our project we have implemented a web application which can be accessed from anywhere with internet availability.

The application of this project can be extended to android application/ ios application/ desktop application. Web pages or mobile applications can be accessed from a small phone anywhere with just the requirement of good internet connectivity.

2.2.2.2.Operational Requirements

- *Economic:* The model can be considered more economic as well as patients can get early detection of the disease i.e. in the preliminary stage itself.
- *Environmental*: Through this we are going digital. The results for the prediction of the disease are digital and less use of paper based reports.
- *Health and Safety:* Disease prediction using Data Analytics can be used to develop an application that can be used in health care systems.
- *Social:* With healthcare being the primary concern in modern day situation, our model provides algorithms for better classification models. This model if implemented or extended into application can be used in healthcare department.

- *Sustainability:* With the advancement in researches and need the web page can constantly be expanded according to the needs. Therefore the system is sustainable.
- *Legality:* This project is developed in compliance for the capstone project under the supervision of faculty of Vellore Institute of Technology.

3. TECHNICAL SPECIFICATION

3.1. SYSTEM REQUIREMENTS

3.1.1. Software Requirements

Operating System Used – Windows 10 Home 64-bit

Language Used for Coding – Python

3.1.1.1. Tools Used

Tools used for visualization and implementation includes Anaconda python and jupyter notebook for effective analysis of liver disease prediction. For building web application, sublime text editor is used.

3.1.1.1.1. *Anaconda* – world's most popular python data science platform. It's an open source software that supports programming languages like python and R for data science, machine learning applications, large scale data processing and predictive analysis.

3.1.1.1.2. *Jupyter* – Notebook documents produced by Jupyter Notebook App integrates both computer code and rich text elements. Computer code can be is multiple languages and rich text elements includes paragraph, equation, figures etc. Thereby enhancing both readability and run executable documents to perform data analysis.

3.1.1.1.3. *Sublime Text Editor* – is a shareware cross platform source code editor with a python application programming interface. It's a free software that supports multiple programming languages and mark-up languages.

3.1.1.2. Libraries

Python libraries is a collection of predefined functions which allows to perform actions without writing the code.

Different python libraries used includes –

3.1.1.2.1. *Pandas* – pandas is a python package which works well with relational and labelled data structures providing fast and flexible solution. Main objective is be fundamental high level building block for real world data analysis in Python.

3.1.1.2.2. *Numpy* – is a library in python which provides support for large and multidimensional arrays which is supported by mathematical functions to operate on arrays.

3.1.1.2.3. *Sklearn* – is a machine learning library that supports various algorithms like random forests, adaboost, gradient, logistic regression, KNN etc. It also necessary to for pre-processing, estimating accuracy score and splitting the data into training and testing data. Scikit learn library makes machine learning in python much more robust and easy.

3.1.1.2.4. *Imblearn* – is a library in python that handles imbalanced data by under-sampling an over- sampling.

3.1.1.2.5. *Plotly* – is a data visualization tool which creates plots that are interactive and helps us manually explore the data by panning, zooming and selecting over the graphing surface

3.1.1.2.6. *Time* – time is python library that provides local time and used to measure the efficiency of the code, waiting during the code and other functionalities representing the time.

3.1.2. Hardware Requirements

3.1.2.1. *Processor* – Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz($ CPU), ~ 2.60GHz

3.1.2.2. *Graphic Card* – AMD Radeon Graphics Processor

3.1.2.3. *RAM – 16.0 GB*

3.1.2.4. *ROM* - 512GB SSD storage1

4. DESIGN APPROACH AND DETAILS

4.1.1. DESIGN APPROACH

4.1.1.1.Design Approach for Data Analysis (Liver Data)

The main aim of this project is to propose a method for building predictive model for liver disease using various supervised machine learning algorithms. The comparative analysis of the proposed method has been done and performance is measured using various classification metrics

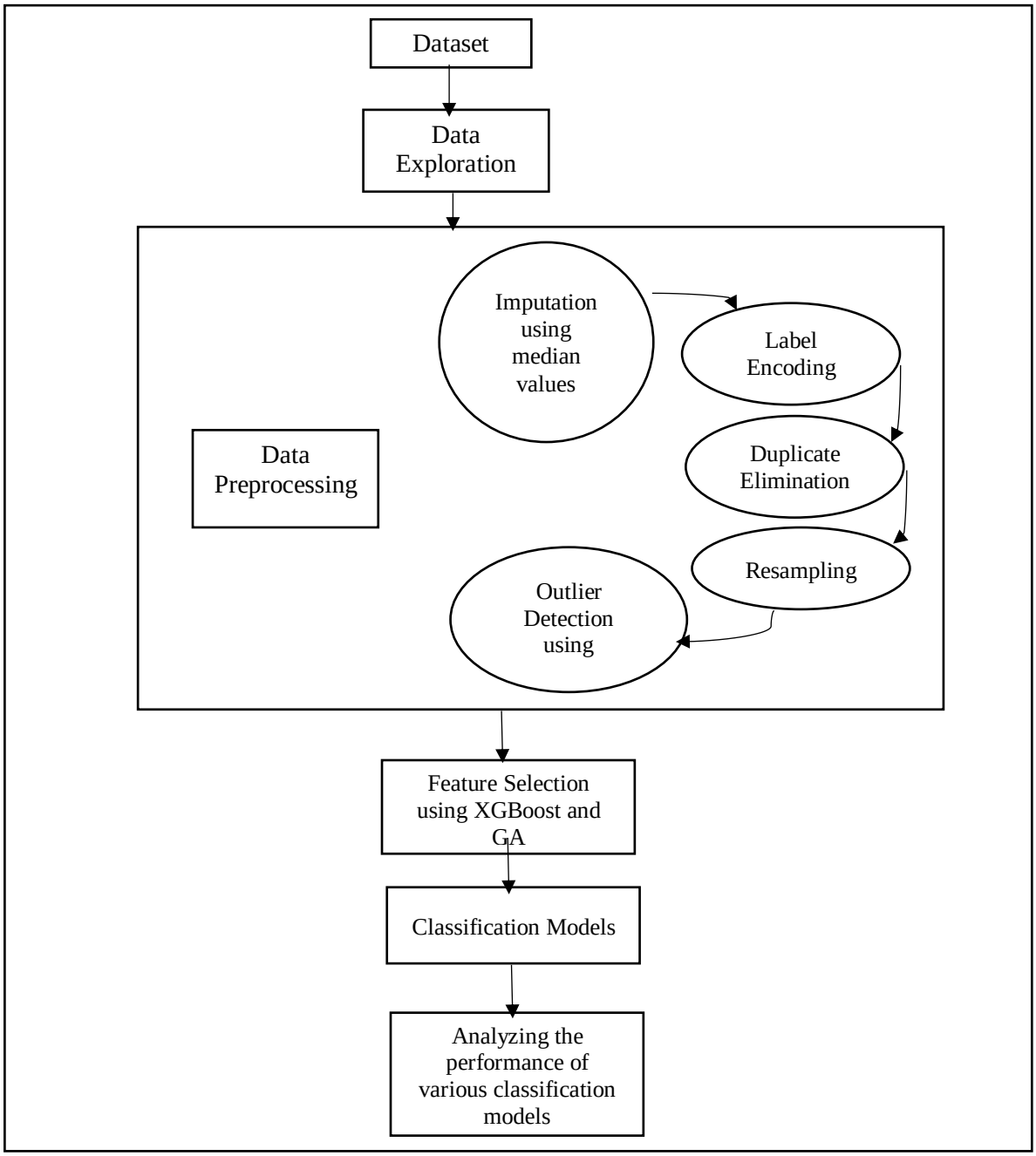

Fig 1. Proposed method for liver disease prediction

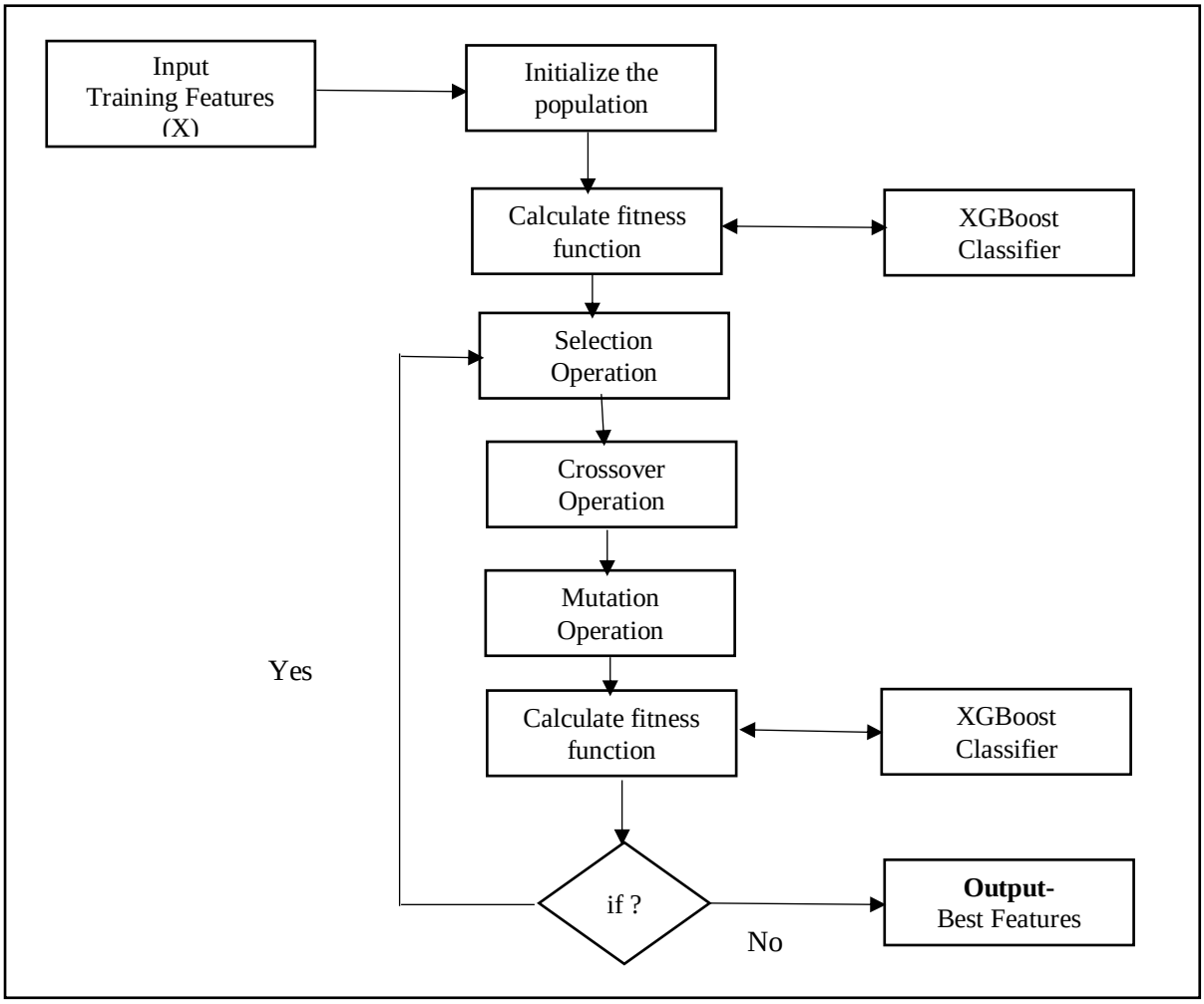

Fig 2. Feature Selection using combination of genetic algorithm and XGBoost

Pseudocode for Feature Selection using Genetic Algorithm and XGBoost classifier:

Input: Training Feature Data (X), Target Data (Y), Classifier used XGBoost algorithm (model)

Start

Read input data X, Y, XGBoost model.

Initializing the population size, generation size and cross validation split.

Evaluating fitness of all Features by averaging cross validation score using XGBoost.

While (termination criteria) do:

Fetching subset of features.

Select best individuals for reproduction.

Generate new individuals through crossover and mutation.

Evaluate the fitness of new individuals by averaging cross validation split using XGBoost.

Replacing least fit population with new individuals.

Return best individual

End

Output: Best Features

4.1.1.2.Design Approach for Web application

Creating a basic web application which allows user to create account and predict the presence and absence of liver disease using various classification algorithms. Also helps the user view the previous history of predictions. The web application is created using Django Framework.

A blog is also created as an interactive platform for the people to put their comments, feedback, suggestions etc.

ER diagram - Entity relationship diagram is high lever conceptual data diagram model. The main aim of this model is to help understand data requirements and produce a well-designed database systematically.

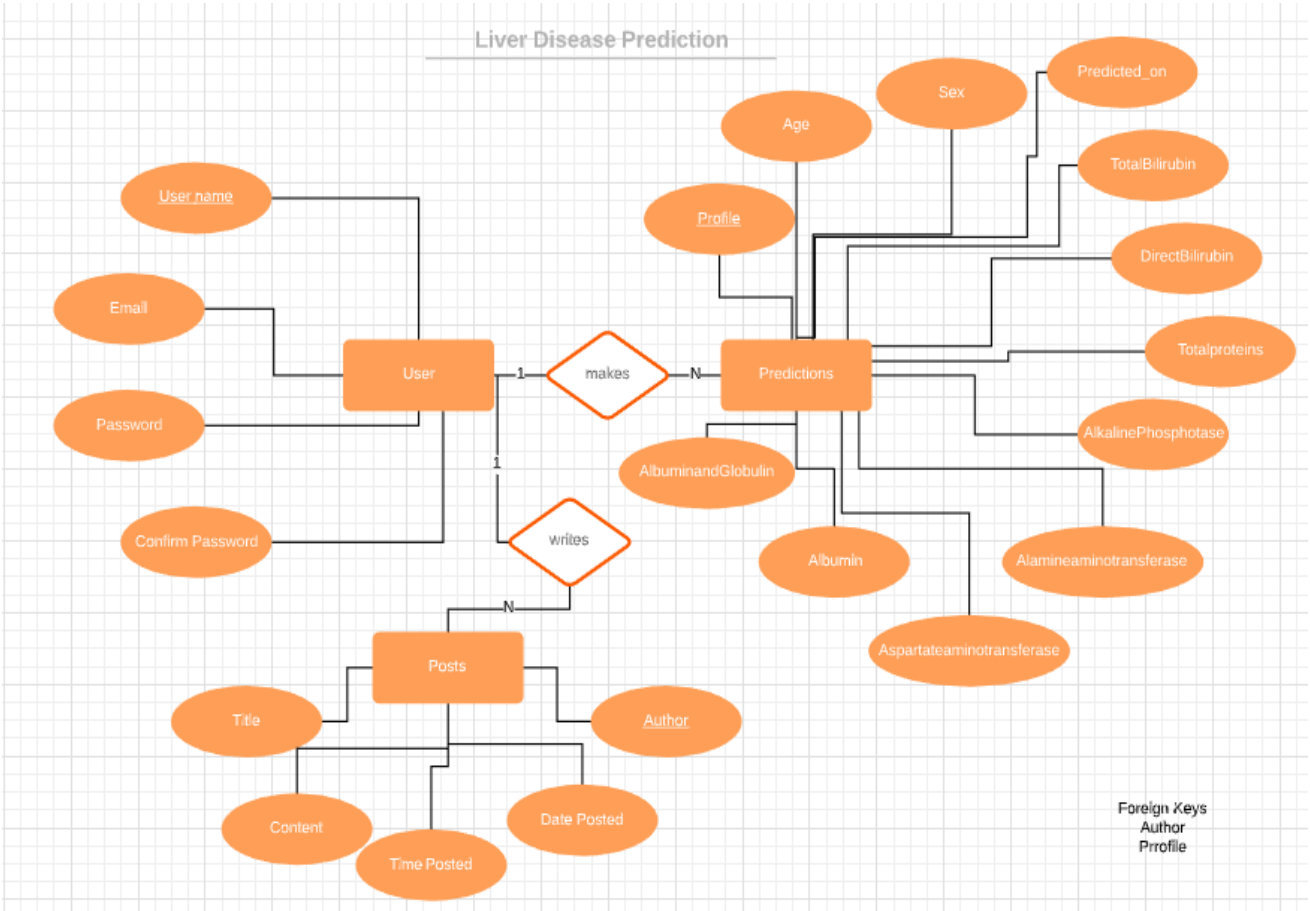

Fig 3. ER Diagram

Class diagram – in Unified Modeling language static structure diagram that shows systems classes their attributes, operations and relationship among objects. The class diagram is main building block for object oriented programming and used for general conceptual modelling

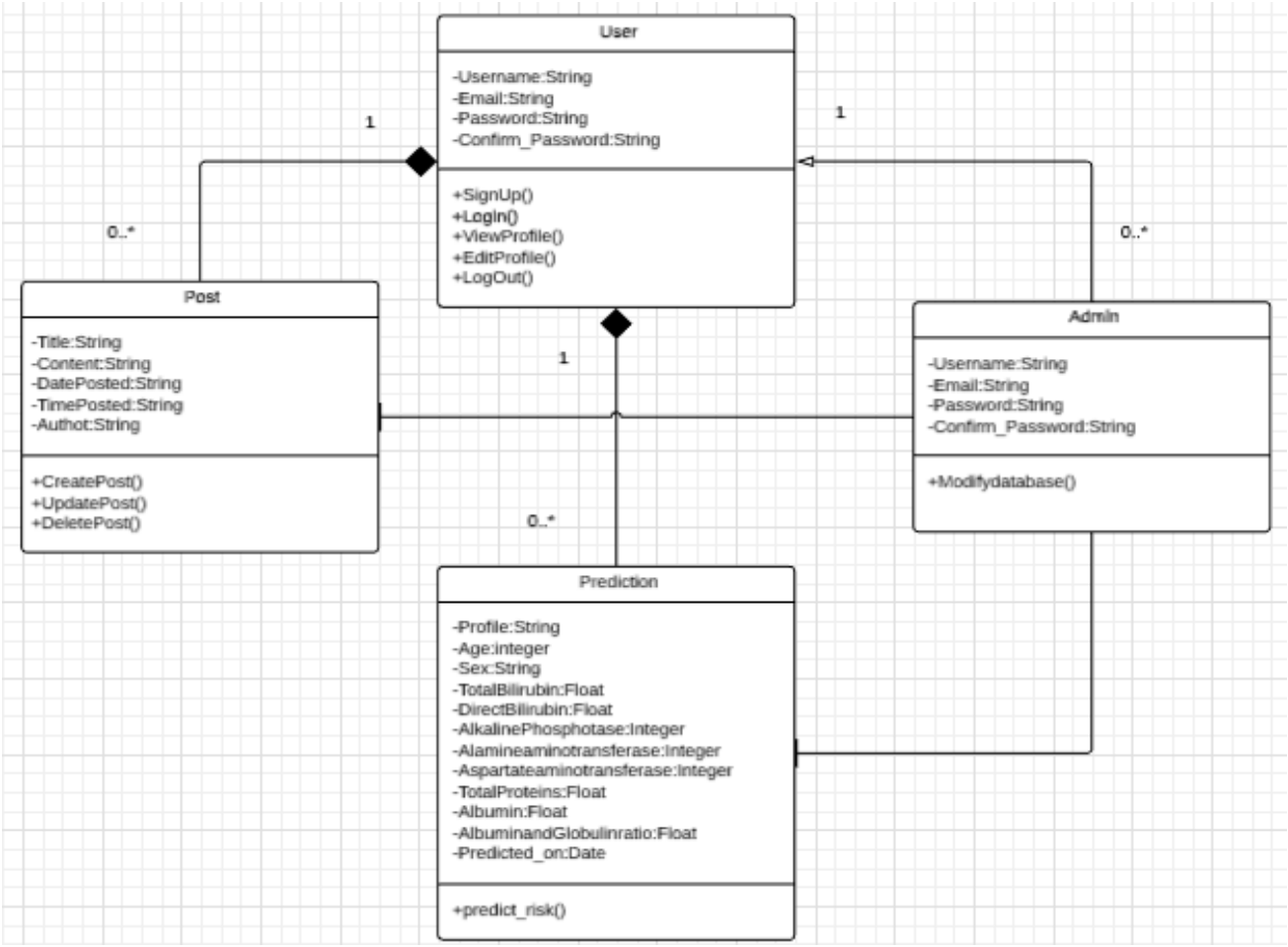

Fig 4. Class Diagram

Activity Diagram – is graphical representation of step wise activities and actions which supports decision making, iteration and concurrency. In Unified modeling language, activity used to model computational and organizational processes and data flows intersecting with related activities.

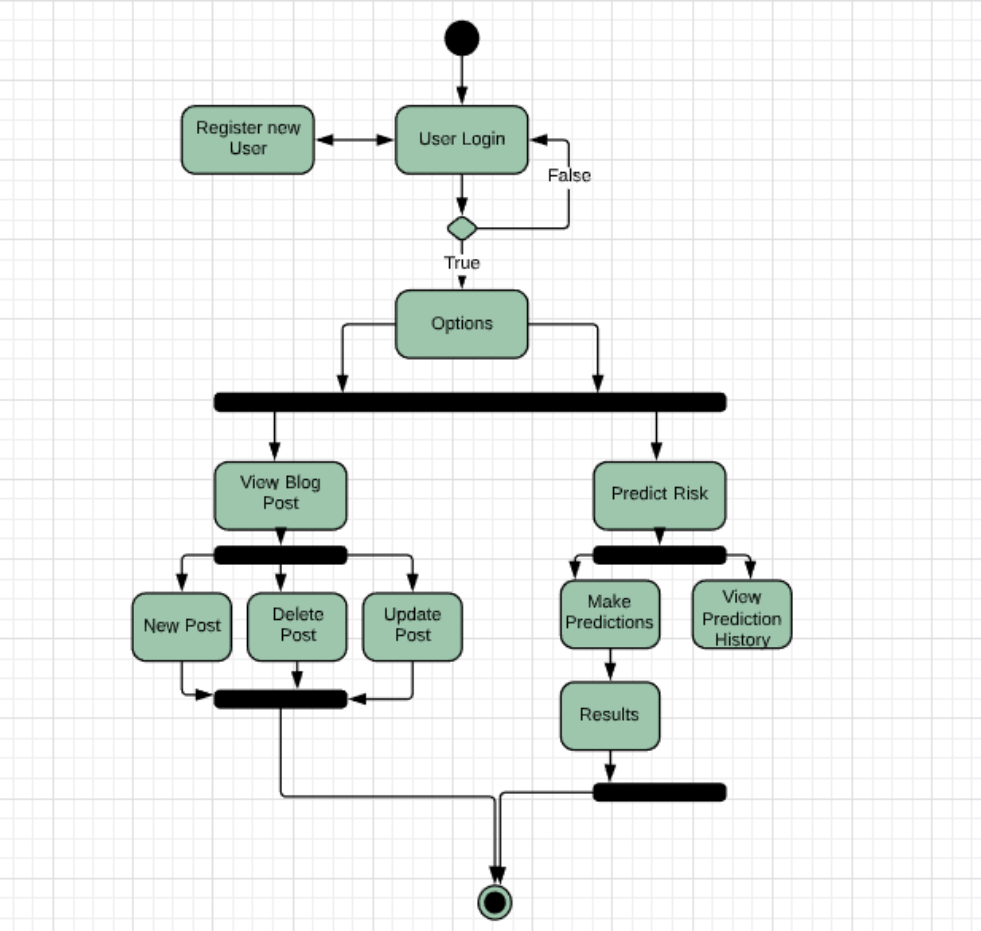

Fig 5. Activity Diagram

4.1.2. METHODS

4.1.2.1. Methods used for data analysis of Liver disease data

A. Data Selection

Data selection process involves the need for selecting appropriate data for analysis and obtaining effective knowledge by performing diverse data mining techniques. The data used for research is Indian Liver Disease Patients (ILDP) from UCI repository.

B. Data Exploration

Data exploration is an initial step of data analysis which inculcates summarizing the data and observing initial patterns in the data and attributes. Various visualization techniques such as histogram and boxplot to identify the extreme and outlier values. Feature correlation of values is assessed in order to identify highly linearly dependent features.

C. Data Pre-processing

- *Imputation of Missing Values* - It refers to identifying missing values in the data and imputing the empty values with median values. For Indian Liver Disease Patients data, Albumin and Globulin ratio has four missing values which is replaced by median values. If there is many missing values then imputation can be implemented using KNN imputation.
- *Label Encoding* - Another data pre-processing technique includes label encoding data which focuses on converting the data into machine readable form. Label encoding converts the labels into numeric forms. In the data used, Gender attribute has labelled data which is converted in to values 1 and 0 for better analysis.
- *Elimination of Duplicate Values* –In order to improve the efficiency and quality of data it's very necessary to eliminate redundant values.
- *Resampling* - Due to the presence of imbalanced data which has majority liver disease and minority non liver disease patients, Synthetic minority over sampling technique used. SMOTE is used to synthesize new samples for the minority. This technique involves blindness problem which is overcome by combining Genetic algorithm with SMOTE known as GASMOTE.
- *Outlier Detection and Elimination* – Outliers are extreme values that significantly deviates from the rest of the values which may be caused due to inappropriate measurement or experimental error. Different types of outliers include univariate and multivariate outliers. Univariate outliers considers a single feature whereas multivariate outliers looks at n-dimensional space consisting of features or attributes of ILPD data. For univariate outlier detection, skewness of attribute is observed and extreme value is replaced. For multivariate outlier detection, isolation forest algorithm is used to identify the contaminated data and it's deleted.

  Isolation Forest is popular technique used to detect anomalies. The technique focuses on mechanism called isolation. This method is highly useful because it doesn't use the commonly used technique based on basic distance and density measure to detect outliers. This algorithm also requires less memory and low linear time complexity.

  Working of Isolation Forest

  1. For each feature there is a range between maximum and minimum to isolate.
  2. Feature is chosen randomly.
  3. Splits value between maximum and minimum vales of selected feature.

4. Number of splitting of a sample is equivalent to path length from root node to terminating node.
5. Measure of normality is estimated by averaging path length of random forest trees.
6. Shorter paths are produced by random partitioning for anomalies.

D. Feature Selection

Feature selection is the process of finding input features for a predictive model which involves removing irrelevant features that don't contribute towards the model. Genetic algorithm is one of the most advanced method for feature selection and optimization method. Which mimics Darwin method of natural selection. Genetic algorithm follows initialization, fitness assignment, selection, crossover and mutation.

Initialization – Population is initialized with the individuals

Fitness Assignment – Fitness is evaluated by training the predictive model and selection error is found out.

Selection – Selection operator finds out the individuals that will recombine for the next generation.

Crossover – Crossover creates a new population by recombining the individuals.

Mutation – Mutation operation is carried out to create a generation varied diversity compared to the previous generation.

E. Classification using machine learning algorithms

Classification is performed using various machine learning algorithms which includes –

- *Logistic Regression* – Used specifically when the target variable which is dependant features is categorical data. Different types logistic regression includes binary logistic regression, multinomial logistic regression and ordinal logistic regression. For liver disease prediction the target variables are presence or absence of liver disease which follows binary logistic regression.
- *KNN* – K-Nearest Neighbors is an algorithm that works based on the close proximity of similar data points.
- *Decision Tree* – Tree based learning algorithm which is a good predictive model which produces better accuracy and ease of interpretation. We use categorical variable

decision tree that creates splits based on gini method. Higher gini value higher homogeneity.

- *Random Forest Tree* – Random forest tree is a machine learning model that is made of many decision trees. Each decision tree trains on different observation. Final outcome prediction is obtained by averaging the predictions of individual decision tree.
- *AdaBoost Classifier* – Adaptive Boosting algorithm which majorly focuses on converting weak classifiers into stronger one. The performance of decision trees is boosted by using the instance of previously trained tree. Additive logistic regression concept is applied in a stage wise fitting.
- *XGBoost Classifier* – Optimised gradient boosting algorithm which improves performance by processing in a parallel manner and uses regularization concept to avoid overfitting issue.
- *LightGBM Classifier* – is one of the high performance gradient boosting algorithm which works based on decision tree algorithm which splits the tree leaf wise rather than depth wise.
- *Multilayer Perceptron* – Feedforwards for artificial neural network which is a deep learning method which is essential in classifying data which are not linearly separable. Perceptron consist of input layer and output layer consisting of multiple hidden layers.
- *Stacking Estimator* – Stacked generalization consist of stacking of individual estimator and using a classifier to compute final prediction. Stacking concept allows to use the strength of each individual and using the output as input of final estimator.

F. Performance metrics analysis

Performance of different machine learning models in analyzed by using metrics such as –

- *Confusion Metric* – is a table that is used for performance analysis which allows easy visualization. Confusion matrix allows distinguish between true positives, false negatives, false positives and true negative.

Table 2
Confusion Metrics

		Actual	
		Positive	**Negative**
Predicted	**Positive**	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

- *Accuracy* – This performance measure is calculated by performing ratio of correctly predicted observation to the total number of observations.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- *Precision* – This is the ratio of relevant instances to total retrieved instances.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- *Recall* – This is a ratio of relevant instances retrieved over the total amount of relevant instances.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- *F-measure* – this measure performs weighted average of precision and recall.

$$\text{F} - \text{measure} = \frac{2 * Precision * Recall}{Precision + Recall}$$

- *Time Complexity* – It is a measure to identify total time taken for predictions.
- *ROC Curve* – Receiving operating characteristic curve is a plot which is used when predicting probability of a binary outcome. The plot displays false positive rate versus (x-axis) true positive rate (y-axis) for threshold values between 0 and 1.

4.2. CODES AND STANDARDS

4.2.1. Data Analysis of Liver Data

```
# instantiate labelencoder object
le = LabelEncoder()
# apply le on categorical feature columns
data['Gender']= le.fit_transform(data['Gender'])
```

Fig 6 – Label Encoding

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='median')
print('Before Imputation - ')
print(data.isnull().sum())
imputer.fit(data)
data=imputer.transform(data)
data=pd.DataFrame(data)
data=data.rename(columns={0:'Age',1:'Gender',2:'Total_Bilirubin',3:'Direct_Bilirubin',4:'Alkaline_Phosphotase ',5:'Alamine_Amino
print('\n\nAfter Imputation - ')
print(data.isnull().sum())
```

Fig 7 – Imputation of Missing values

```
correlation=data.corr()

plt.figure(figsize=(40, 40))
sns.heatmap(correlation, cbar = True,  square = True, annot=True, fmt= '.2f',annot_kws={'size': 15},
           cmap= 'coolwarm')
plt.title('Correlation between features')
```

Fig 8 - Correlation

```
data = data.drop_duplicates()
print(data.shape)
```

Fig 9 – Dropping duplicate values

```
import smote_variants as sv
oversampler= sv.GASMOTE()

Y=np.array(y)
X=np.array(x)


#oversampler= sv.MulticlassOversampling(sv.SMOTE_PSO())

# X_samp and y_samp contain the oversampled dataset
X_samp, y_samp= oversampler.sample(X, Y)
```

Fig 10 - Oversampling

```
from sklearn.ensemble import IsolationForest
iso_forest = IsolationForest(n_estimators=300, contamination=0.20)
iso_forest = iso_forest.fit(x)
isof_outliers = iso_forest.predict(x)
isoF_outliers_values = x[iso_forest.predict(x) == -1]
print(isoF_outliers_values)
```

Fig 11 – Isolation Forest for outlier removal

```python
from deap import creator, base, tools, algorithms
import random
import numpy as np
from deap import tools
import pandas as pd
from sklearn.model_selection import cross_val_score

class FeatureSelectionGA:
    """
    FeatureSelectionGA
    This Class uses Genetic Algorithm to find out the best features for the input model using
    Distributed Evolutionary Algorithms in Python(DEAP) package.
    """

    def __init__(self, model, x, y, cv_split=5, random_state=20, n_pop=25, n_gen=25):
        """
            Parameters
            ----------
            model : scikit-learn supported model,
                x :  {array-like}, shape = [n_samples, n_features]
                     Training vectors, where n_samples is the number of samples
                     and n_features is the number of features.
                y  : {array-like}, shape = [n_samples]
                     Target Values
            cv_split: int
                     Number of splits for cross_validation to calculate fitness.
            Random State: As Specified
            n_pop: It will be Identified to determine the Population Size
            n_gen: It would be Identified to determine the Number of Generation
        """
        self.model = model
        self.n_features = x.shape[1]
        self.cv_split = cv_split
        self.x = x
        self.y = y
        self.random_state = random_state
        self.n_pop = n_pop
        self.n_gen = n_gen
        print("The number of Features received by the system is : {}".format(self.n_features))
        print("The Shape of Training Data is : {} and Target Data is : {}".format(self.x.shape, self.y.shape))

        individual = [1 for i in range(x.shape[1])]
        print("Accuracy For All the features: " + str(self.fitness_test(individual)) + "\n")

        # Applying Genetic Algorithm
        hof = self.evolutionary_algorithm()
        accuracy, individual, header = self.bestIndividual(hof)
        print('Best Accuracy: \t' + str(accuracy[0]))
        print('Number of Features in Subset: \t' + str(individual.count(1)))
        print('Feature Subset: ' + str(header)+'\n')
        print('\n\nKindly Create a New Classifier with the Above Feature Set')
```

```python
    def fitness_test(self, individual):
        """
        The Function Analyses Provides the average Cross Val Score Using All the Features
        :param individual:
        :return: Average Cross Val Score
        """
        if (individual.count(0) != len(individual)):
            # Fetched the Index of the Individual
            cols = [index for index in range(len(individual)) if individual[index] == 0]

            # Fetching Feature Subset
            X_parsed = self.x.drop(self.x.columns[cols], axis=1)
            X_subset = pd.get_dummies(X_parsed)

            # Applying the Classification Algorithm
            classifier = self.model
            cross_v=cross_val_score(classifier, X_subset, self.y, cv=self.cv_split)
            return ((sum(cross_v) / float(len(cross_v))),0)
        else:
            return (0,)

    def evolutionary_algorithm(self):
        """
        Declaring Global Variables for DEAP
        :return:
        """
        # Creating the Individual Using DEAP
        creator.create("FitnessMax", base.Fitness, weights=(1.0,))
        creator.create("Individual", list, fitness=creator.FitnessMax)

        # Creating ToolBox For The DEAP Framework
        toolbox = base.Toolbox()
        toolbox.register("attr_bool", random.randint, 0, 1)
        toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_bool, self.n_features)
        toolbox.register("population", tools.initRepeat, list, toolbox.individual)
        toolbox.register("evaluate", self.fitness_test)
```

Fig 12– Feature Selection using Genetic Algorithm

```
        # Creating ToolBox For The DEAP Framework
        toolbox = base.Toolbox()
        toolbox.register("attr_bool", random.randint, 0, 1)
        toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_bool, self.n_features)
        toolbox.register("population", tools.initRepeat, list, toolbox.individual)
        toolbox.register("evaluate", self.fitness_test)
        toolbox.register("mate", tools.cxOnePoint)
        toolbox.register("mutate", tools.mutFlipBit, indpb=0.05)
        toolbox.register("select", tools.selTournament, tournsize=3)

        # Initialize Parameters
        pop = toolbox.population(n=self.n_pop)
        hof = tools.HallOfFame(self.n_pop * self.n_gen)
        stats = tools.Statistics(lambda ind: ind.fitness.values)
        stats.register("avg", np.mean)
        stats.register("min", np.min)
        stats.register("max", np.max)

        # Genetic Algorithm
        pop, log = algorithms.eaSimple(pop, toolbox, cxpb=0.5, mutpb=0.2, ngen=self.n_gen, stats=stats, halloffame=hof,verbose=T

        # Return Fall Of Home
        return hof

    def bestIndividual(self, hof):
        """
        Get the best individual
        """
        maxAccurcy = 0.0
        for individual in hof:
            if (individual.fitness.values[0] > maxAccurcy):
                maxAccurcy = individual.fitness.values[0]
                _individual = individual

        _individualHeader = [list(self.x)[i] for i in range(len(_individual)) if _individual[i] == 1]
        return _individual.fitness.values, _individual, _individualHeader
```

Fig 13 – Feature Selection using Genetic Algorithm

```
training_accuracy = []
test_accuracy = []
neighbors_settings = range(1,10)
for n_neighbors in neighbors_settings:
# build the model
    clf = KNeighborsClassifier(n_neighbors=n_neighbors)
    clf.fit(xtrain, ytrain)
# record training set accuracy
    training_accuracy.append(clf.score(xtrain, ytrain))
# record generalization accuracy
    test_accuracy.append(clf.score(xtest, ytest))
plt.plot(neighbors_settings, training_accuracy, label="training accuracy")
plt.plot(neighbors_settings, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("n_neighbors")
plt.legend()


knn=KNeighborsClassifier(n_neighbors=9)
st=time.time()
knn.fit(xtrain,ytrain)
test=knn.predict(xtest)
et=time.time()
time_knn=et-st
knn.score(xtrain,ytrain)
score_knn=accuracy_score(test,ytest)
accuracy_score(test,ytest)
print(classification_report(ytest,test))
```

Fig 14 – KNN Algorithm

```
training_accuracy = []
test_accuracy = []
cvalue = [0.001,0.01,0.1,10,100]
for c in cvalue:
    log1=LogisticRegression(C=c,max_iter=10000,penalty='l2').fit(xtrain,ytrain)
    test=log1.predict(xtest)
    log1.score(xtrain,ytrain)
    accuracy_score(test,ytest)
    training_accuracy.append(log1.score(xtrain, ytrain))
    test_accuracy.append(log1.score(xtest, ytest))
plt.plot(cvalue, training_accuracy, label="training accuracy")
plt.plot(cvalue, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("c value")
plt.legend()

log1=LogisticRegression(C=100,max_iter=10000,random_state=25)
#rfe = RFE(log1, 9)
st=time.time()
log1=log1.fit(xtrain,ytrain)
test=log1.predict(xtest)
et=time.time()
time_log=et-st
#log1.score(xtrain,ytrain)
score_log=accuracy_score(test,ytest)
accuracy_score(test,ytest)
print(classification_report(ytest,test))
```

Fig 15 – Logistic Regression

```
tree=DecisionTreeClassifier(criterion="gini",max_depth=30,random_state=50)
st=time.time()
tree.fit(xtrain,ytrain)
test=tree.predict(xtest)
et=time.time()
time_dt=et-st
tree.score(xtrain,ytrain)
accuracy_score(test,ytest)
score_dt=accuracy_score(test,ytest)
print(classification_report(ytest,test))

print(tree.feature_importances_)

plt.barh(range(10),tree.feature_importances_,align='center')
plt.yticks(np.arange(10),x.columns)
plt.xlabel('Feature importance')
plt.ylabel('Feature')
```

Fig 16 – Decision Tree Classifier

```
forest=RandomForestClassifier(n_estimators=100,max_depth=30,random_state=50)
st=time.time()
forest.fit(xtrain,ytrain)
test=forest.predict(xtest)
et=time.time()
time_rf=et-st
forest.score(xtrain,ytrain)
accuracy_score(test,ytest)
score_rf=accuracy_score(test,ytest)
print(classification_report(ytest,test))

plt.barh(range(10),forest.feature_importances_,align='center')
plt.yticks(np.arange(10),x.columns)
plt.xlabel('Feature importance')
plt.ylabel('Feature')
```

Fig 17 – Random Forest Tree

```
gradient=GradientBoostingClassifier(n_estimators=50,learning_rate=1,max_depth=30,random_state=50)
st=time.time()
gradient.fit(xtrain,ytrain)
test=gradient.predict(xtest)
et=time.time()
time_gba=et-st
gradient.score(xtrain,ytrain)
score_gba=accuracy_score(test,ytest)
accuracy_score(test,ytest)
print(classification_report(ytest,test))

plt.barh(range(10),gradient.feature_importances_,align='center')
plt.yticks(np.arange(10),x.columns)
plt.xlabel('Feature importance')
plt.ylabel('Feature')
```

Fig 18 – Gradient Boosting Algorithm

```
from sklearn.ensemble import AdaBoostClassifier
model = AdaBoostClassifier(n_estimators=200,learning_rate=1,random_state=50)
st=time.time()
model.fit(xtrain, ytrain)
test=model.predict(xtest)
et=time.time()
time_ab=et-st
model.score(xtrain,ytrain)
score_ab=accuracy_score(test,ytest)
accuracy_score(test,ytest)
print(classification_report(ytest,test))

plt.barh(range(10),model.feature_importances_,align='center')
plt.yticks(np.arange(10),x.columns)
plt.xlabel('Feature importance')
plt.ylabel('Feature')
```

Fig 19 – AdaBoost Classifier

```
import xgboost as xgb
model=xgb.XGBClassifier(n_estimators=150,max_depth=30,random_state=50)
st=time.time()
model.fit(xtrain, ytrain)
test=model.predict(xtest)
et=time.time()
time_xgb=et-st
model.score(xtrain,ytrain)
accuracy_score(test,ytest)
score_xgb=accuracy_score(test,ytest)
print(classification_report(ytest,test))

plt.barh(range(10),model.feature_importances_,align='center')
plt.yticks(np.arange(10),x.columns)
plt.xlabel('Feature importance')
plt.ylabel('Feature')
```

Fig 20 – XGBoosst Classifier

```
from lightgbm import LGBMClassifier
model=LGBMClassifier(n_estimators=100,max_depth=30,random_state=50)
st=time.time()
model.fit(xtrain, ytrain)
test=model.predict(xtest)
model.score(xtrain,ytrain)
et=time.time()
time_gbm=et-st
accuracy_score(test,ytest)
score_gbm=accuracy_score(test,ytest)
print(classification_report(ytest,test))

plt.barh(range(10),model.feature_importances_,align='center')
plt.yticks(np.arange(10),x.columns)
plt.xlabel('Feature importance')
plt.ylabel('Feature')
```

Fig 21 – LightGBM Classifier

```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import make_pipeline, make_union
from tpot.builtins import StackingEstimator
from xgboost import XGBClassifier


features = X_samp
training_features, testing_features, training_target, testing_target = \
            train_test_split(features, y_samp, random_state=None)

exported_pipeline = make_pipeline(
    StackingEstimator(estimator=RandomForestClassifier(bootstrap=True, criterion="gini", max_features=0.8500000000000001, min_sa
    StackingEstimator(estimator=SGDClassifier(alpha=0.01, eta0=0.01, fit_intercept=False, l1_ratio=0.75, learning_rate="constant
    StackingEstimator(estimator=XGBClassifier(learning_rate=0.1, max_depth=8, min_child_weight=10, n_estimators=100, nthread=1,
    KNeighborsClassifier(n_neighbors=1, p=1, weights="distance")
)
st=time.time()
exported_pipeline.fit(training_features, training_target)
results = exported_pipeline.predict(testing_features)
et=time.time()
time_stack1=et-st
score_stack=accuracy_score(testing_target,results)
print(classification_report(testing_target,results))
```

Fig 22 – Stacking Estimator 1

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline, make_union
from sklearn.preprocessing import StandardScaler
from tpot.builtins import StackingEstimator

# NOTE: Make sure that the outcome column is labeled 'target' in the data file
features = X_samp
training_features, testing_features, training_target, testing_target = \
            train_test_split(features, y_samp, random_state=None)


# Average CV score on the training set was: 0.8808420607214641
exported_pipeline = make_pipeline(
    StackingEstimator(estimator=SGDClassifier(alpha=0.0, eta0=1.0, fit_intercept=True, l1_ratio=1.0, learning_rate="invscaling",
    StackingEstimator(estimator=MultinomialNB(alpha=100.0, fit_prior=False)),
    StandardScaler(),
    GradientBoostingClassifier(learning_rate=0.5, max_depth=8, max_features=0.8500000000000001, min_samples_leaf=14, min_samples
)

st1=time.time()
exported_pipeline.fit(training_features,np.ravel(training_target,order='C'))
results = exported_pipeline.predict(testing_features)
et1=time.time()
time_se1=et1-st1
score_stack1=accuracy_score(testing_target,results)
print(classification_report(testing_target,results))
```

Fig 23 – Stacking Estimator 2

```
import time
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(activation='logistic',hidden_layer_sizes=(20, 2), max_iter=1500, alpha=1e-4,
                    solver='adam', verbose=10, tol=0.000001, random_state=50,
                    learning_rate_init=0.001,learning_rate='adaptive',momentum=0.8)
st=time.time()
mlp.fit(xtrain,ytrain)
print("Training set score: %f" % mlp.score(xtrain,ytrain))
test=mlp.predict(xtest)
et=time.time()
time_mlp=et-st
score_nn=accuracy_score(test,ytest)
print(accuracy_score(test,ytest))
#print("Test set score: %f" % mlp.score(learnset, learnlabels))
print(classification_report(ytest,test))
```

Fig 24 – MLP

4.2.2. Code for Web Application

```
from django.conf.urls import url
from . import views
app_name = 'accounts'
urlpatterns=[
    url(r'^register/$', views.register, name='register'),
    url(r'^logout/$',views.user_logout,name='logout'),
    url(r'^profile/(?P<pk>\d+)/$', views.ProfileDetailView.as_view(), name='profile'),
#url(r'^profile/(?P<pk>\d+)/edit/$', views.profile_update, name='edit_profile'),
]
```

Fig 25 – URL for Accounts

```
from django.db import models
from django.contrib.auth.models import User
from django.conf.urls import url
from django.shortcuts import resolve_url


class UserProfileInfo(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, related_name='profile')
    profile_pic = models.ImageField(upload_to='profile_pics', default="1.jpg")

    def __str__(self):
        return self.user.username
```

Fig 26 – Models for accounts

```python
from django import forms
from .models import UserProfileInfo
from django.contrib.auth.models import User

class UserForm(forms.ModelForm):
    username = forms.CharField(widget=forms.TextInput(
        attrs={'class': 'form-control', 'placeholder': 'Enter username'}
    ), required=True, max_length=50)

    email = forms.CharField(widget=forms.EmailInput(
        attrs={'class': 'form-control', 'placeholder': 'Enter Email Id'}
    ), required=True, max_length=50)

    password = forms.CharField(widget=forms.PasswordInput(
        attrs={'class': 'form-control', 'placeholder': 'Enter password'}
    ), required=True, max_length=50)

    confirm_password = forms.CharField(widget=forms.PasswordInput(
        attrs={'class': 'form-control', 'placeholder': 'Confirm password'}
    ), required=True, max_length=50)

    class Meta:
        model = User
        fields = ('username', 'email', 'password')

    def clean(self):
        cleaned_data = super(UserForm, self).clean()
        password = cleaned_data.get("password")
        confirm_password = cleaned_data.get("confirm_password")

        if password != confirm_password:
            raise forms.ValidationError(
                "Password and Confirm password does not match"
            )


class UserProfileInfoForm(forms.ModelForm):
    class Meta:
        model = UserProfileInfo
        fields = ('profile_pic',)


class UpdateProfileForm(forms.ModelForm):
    class Meta:
        model = User
        fields = ('username', 'first_name', 'last_name', 'email')
        widgets = {'username': forms.TextInput(attrs={'readonly': 'True'})}
```

Fig 27 – Forms for accounts

```python
from django.apps import AppConfig


class AccountsConfig(AppConfig):
    name = 'accounts'

    def ready(self):
        import users.signals
```

Fig 28 – App config for Accounts

```python
from django.contrib import admin
from accounts.models import UserProfileInfo

class UserProfile(admin.ModelAdmin):
    list_display = ('user', 'profile_pic')
admin.site.register(UserProfileInfo,UserProfile)
# Register your models here.
```

Fig 29 – Admin for Accounts

```python
import csv,io
from django.shortcuts import render
from .forms import Predict_Form
from predict_risk.data_provider import *
from accounts.models import UserProfileInfo
from django.shortcuts import get_object_or_404, redirect, render
from django.http import HttpResponseRedirect, HttpResponse
from django.contrib.auth.decorators import login_required,permission_required
from django.urls import reverse
from django.contrib import messages

@login_required(login_url='/')
def PredictRisk(request,pk):
    predicted = False
    predictions={}
    if request.session.has_key('user_id'):
        u_id = request.session['user_id']

    if request.method == 'POST':
        form = Predict_Form(data=request.POST)
        profile = get_object_or_404(UserProfileInfo, pk=pk)


        print(form.errors)
        if form.is_valid():
            features = [[ form.cleaned_data['age'], form.cleaned_data['sex'], form.cleaned_data['tb'], form.cleaned_data['db'], form.cleaned_data['alp'],
            form.cleaned_data['ala'], form.cleaned_data['apa'], form.cleaned_data['tp'], form.cleaned_data['alb'],
            form.cleaned_data['aag']]]


            #standard_scalar = GetStandardScalarForHeart()
            #features = standard_scalar.transform(features)
            XGBClassifier,LogisticRegressionClassifier,NaiveBayesClassifier,DecisionTreeClassifier,GradientBoostClassifier,RandomForestTreeClassifier,GBMClassifier,AdaBoo

            #'SVC': str(SVCClassifier.predict(features)[0]),
            predictions = {
              'XGB':str(XGBClassifier.predict(features)[0]),
              'LogisticRegression': str(LogisticRegressionClassifier.predict(features)[0]),
              'NaiveBayes': str(NaiveBayesClassifier.predict(features)[0]),
              'DecisionTree': str(DecisionTreeClassifier.predict(features)[0]),
              'GradientBoost':str(GradientBoostClassifier.predict(features)[0]),
              'RandomForest':str(RandomForestTreeClassifier.predict(features)[0]),
              'GBM':str(GBMClassifier.predict(features)[0]),
              'AdaBoost':str(AdaBoostClassifier.predict(features)[0]),
              'KNN':str(KNNClassifier.predict(features)[0]),
              'MLP':str(MLPClassifier.predict(features)[0]),
              'StackingEstimator1':str(StackingEstimator1.predict(features)[0]),
              'StackingEstimator2':str(StackingEstimator2.predict(features)[0]),
               }
            pred = form.save(commit=False)

            #predictions['SVC'],
            l=[predictions['XGB'],predictions['LogisticRegression'],predictions['NaiveBayes'],predictions['DecisionTree'],predictions['GradientBoost'],predictions['Rando
            count=l.count('1.0')
```

Fig 30 –Views for predicting risk

```python
from django.conf.urls import url
from . import views
app_name='predict'
urlpatterns=[
url(r'^(?P<pk>\d+)$',views.PredictRisk,name='predict')
]
```

Fig 31 – URL for predicting risk

```python
from django.db import models
from accounts.models import UserProfileInfo
from django.utils import timezone
from django.urls import reverse
# Create your models here.
sex_choices=((0, 'Female'),(1, 'Male'))

class Predictions(models.Model):
    profile = models.ForeignKey(UserProfileInfo, on_delete=models.CASCADE, related_name='predict')
    age = models.IntegerField()
    sex = models.IntegerField(choices=sex_choices, default=0)
    tb = models.DecimalField(max_digits=4, decimal_places=2)
    db = models.DecimalField(max_digits=4, decimal_places=2)
    alp = models.IntegerField()
    ala = models.IntegerField()
    apa = models.IntegerField()
    tp = models.DecimalField(max_digits=4, decimal_places=2)
    alb = models.DecimalField(max_digits=4, decimal_places=2)
    aag = models.DecimalField(max_digits=4, decimal_places=2)


    predicted_on = models.DateTimeField(default=timezone.now)
    num=models.IntegerField()

    def get_absolute_url(self):
        return reverse('predict:predict', kwargs={'pk': self.profile.pk})
```

Fig 32 – Models for predicting risk

```
config = {
    'liver': {

        'LogisticRegression': 'production/Logistic_regression_model.pkl',
        'NaiveBayes': 'production/naive_bayes_model.pkl',
        'DecisionTree':'production/decision_tree_model.pkl',
        'GradientBoost':'production/gradient_model.pkl',
        'RandomForest':'production/randomforest_model.pkl',
        'XGB':'production/xgboost_model.pkl',
        'GBM':'production/lightgbm_model.pkl',
        'AdaBoost':'production/adaboost_model.pkl',
        'KNN':'production/knn_model.pkl',
        'MLP':'production/MLP_model.pkl',
        'stackingestimator1':'production/StackingEstimator1_model.pkl',
        'stackingestimator2':'production/StackingEstimator2_model.pkl',

        #'scalar_file': 'production/standard_scalar.pkl',
    }}

dir = os.path.dirname(__file__)

def GetJobLibFile(filepath):
    if os.path.isfile(os.path.join(dir, filepath)):
        return joblib.load(os.path.join(dir, filepath))
    return None

def GetPickleFile(filepath):
    if os.path.isfile(os.path.join(dir, filepath)):
        return pickle.load( open(os.path.join(dir, filepath), "rb" ) )
    return None

def GetStandardScalarForHeart():
    return GetPickleFile(config['liver']['scalar_file'])

def GetAllClassifiersForLiver():
    return (XGBClassifier(),GetLogisticRegressionClassifier(),GetNaiveBayesClassifier(),GetDecisionTreeClassifier(),GetGradientBoostClassifier(),GetRandomForestClassifie


def GetLogisticRegressionClassifier():
    return GetJobLibFile(config['liver']['LogisticRegression'])

def GetNaiveBayesClassifier():
    return GetJobLibFile(config['liver']['NaiveBayes'])

def GetDecisionTreeClassifier():
    return GetJobLibFile(config['liver']['DecisionTree'])

def GetGradientBoostClassifier():
    return GetJobLibFile(config['liver']['GradientBoost'])

def GetRandomForestClassifier():
    return GetJobLibFile(config['liver']['RandomForest'])
```

Fig 33 – Classifier Analysis

```
from django.contrib import admin
from predict_risk.models import Predictions
from django import forms

class Prediction(admin.ModelAdmin):
    list_display=('profile','age','sex','tb','db','alp','ala','apa','tp','alb','aag','predicted_on','num')
admin.site.register(Predictions,Prediction)
# Register your models here.
```

Fig 34 – Admin for predicting risk

```
from django import forms
from predict_risk.models import Predictions

class Predict_Form(forms.ModelForm):
    class Meta:
        model = Predictions
        fields = ('age','sex','tb','db','alp','ala','apa','tp','alb','aag')
        widgets = {'age': forms.TextInput(attrs={'class': 'form-control'}),
                   'sex': forms.Select(attrs={'class': 'form-control'}),
                   'tb':forms.TextInput(attrs={'class': 'form-control'}),
                   'db':forms.TextInput(attrs={'class': 'form-control'}),
                   'alp':forms.TextInput(attrs={'class': 'form-control'}),
                   'ala':forms.TextInput(attrs={'class': 'form-control'}),
                   'apa':forms.TextInput(attrs={'class': 'form-control'}),
                   'tp':forms.TextInput(attrs={'class': 'form-control'}),
                   'alb':forms.TextInput(attrs={'class': 'form-control'}),
                   'aag':forms.TextInput(attrs={'class': 'form-control'}),

                   }
```

Fig 35 – Forms for predicting risk

```python
def home(request):
    context = {
        'posts': Post.objects.all()
    }
    return render(request, 'blog/home.html', context)


class PostListView(ListView):
    model = Post
    template_name = 'blog/home.html'  # <app>/<model>_<viewtype>.html
    context_object_name = 'posts'
    ordering = ['-date_posted']


class PostDetailView(DetailView):
    model = Post


class PostCreateView(LoginRequiredMixin, CreateView):
    model = Post
    fields = ['title', 'content']

    def form_valid(self, form):
        form.instance.author = self.request.user
        return super().form_valid(form)


class PostUpdateView(LoginRequiredMixin, UserPassesTestMixin, UpdateView):
    model = Post
    fields = ['title', 'content']

    def form_valid(self, form):
        form.instance.author = self.request.user
        return super().form_valid(form)

    def test_func(self):
        post = self.get_object()
        if self.request.user == post.author:
            return True
        return False


class PostDeleteView(LoginRequiredMixin, UserPassesTestMixin, DeleteView):
    model = Post
    success_url = '/'

    def test_func(self):
        post = self.get_object()
        if self.request.user == post.author:
            return True
        return False


def about(request):
    return render(request, 'blog/about.html', {'title': 'About'})
```

Fig 36 – Views for blog

```python
from django.urls import path
from .views import (
    PostListView,
    PostDetailView,
    PostCreateView,
    PostUpdateView,
    PostDeleteView
)
from . import views


urlpatterns = [
    path('', PostListView.as_view(), name='blog-home'),
    path('post/<int:pk>/', PostDetailView.as_view(), name='post-detail'),
    path('post/new/', PostCreateView.as_view(), name='post-create'),
    path('post/<int:pk>/update/', PostUpdateView.as_view(), name='post-update'),
    path('post/<int:pk>/delete/', PostDeleteView.as_view(), name='post-delete'),
    path('about/', views.about, name='blog-about'),
]
```

Fig 37 – URL for blog

```python
from django.contrib import admin

# Register your models here.
from django.contrib import admin
from .models import Post

admin.site.register(Post)
```

Fig 38 – Admin for blog post

```python
from django.db import models
from django.utils import timezone
from django.contrib.auth.models import User
from django.urls import reverse


class Post(models.Model):
    title = models.CharField(max_length=100)
    content = models.TextField()
    date_posted = models.DateTimeField(default=timezone.now)
    author = models.ForeignKey(User, on_delete=models.CASCADE)

    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return reverse('post-detail', kwargs={'pk': self.pk})
```

Fig 39 – Models for blog

4.3. CONSTRAINTS, ALTERNATIVES AND TRADEOFFS

4.3.1. Lack of Data

Large amounts of data is required for most of the machine learning algorithms to produce accurate results. Reusing data is not the best option instead oversampling techniques can be applied, but preferred option is to have more data.

4.3.2. Quality of Data

It's very necessary to ensure that quality of data collected is reliable, understandable, relevant, complete, high quality and up to date that be confidently used for meaningful research purpose.

- Reliability – Data should reflect real time scenario and in some cases can contain approximation but guessing should be avoided.
- Understandability – There should be clarity in the data and it should be clear to the intended users.
- Relevance – The data collected should be complete and updated.

5. SCHEDULES, TASKS AND MILESTONES

Table 3
Task Description

Week Number	Task Scheduled	Description of work carried
Week 1	Problem Identification	Project title identified and how to incorporate new features and bring improvements in existing methodologies. Also identifying gaps in the existing methodologies.
Week 2	Literature Survey	Conducting literature survey by reading multiple research papers and understanding how the existing methodologies can be improved.
Week 3	Requirement Analysis	Collecting all the requirements by analyzing the literature surveys. Identifying the data set and source of the data. The data selected is Indian Liver Disease Patient Data collected from UCI Machine learning repository.
Week 4	Project Discussion	Discussing the problem statement, preliminary objectives and scope of the project with the guide.
Week 5	Idea Brainstorming	Identifying new techniques for performing data analysis of liver disease data. Consolidating the machine learning techniques that has to be applied to the Liver disease data. Start preparing the basic document for research paper publication.
Week 6,7,8	Data Analytics	Implementing the code in Jupyter Notebook. Performing data selection, data exploration, data visualization, data cleaning, feature selection, model selection and performance analysis. Genetic Algorithm, Isolation Forest and GASMOTE is explored and performed.
Week 9,10,11	Research paper submission	Preparing the contents of research paper for publication. Paper is prepared in IEEE form and submitted for plagiarism check.

Week 12	Project Discussion	Project Progress is discussed in terms of paper presentation and code for data analytics is explained to the guide.
Week 13,14	Preparing for the research paper presentation and Presenting the research	Research paper final proof reading is done and preparing the presentation. Presenting the research before the committee.
Week 15,16,17	Web application development	Learning Django and how to develop a web application. Creating a web application for prediction of liver disease. Creating HTML, CSS and database code along with creation of models and forms in Django for the application.
Week 18	Final Bug Fixing	Running the data science model and Django web application again and check whether there are any errors and fixing them.
Week 19, 20	Project Report	Preparing the content for final project report.
Week 21	Paper published in IEEE digital Xplore	Paper published - https://ieeexplore.ieee.org/document/9074368/authors#authors

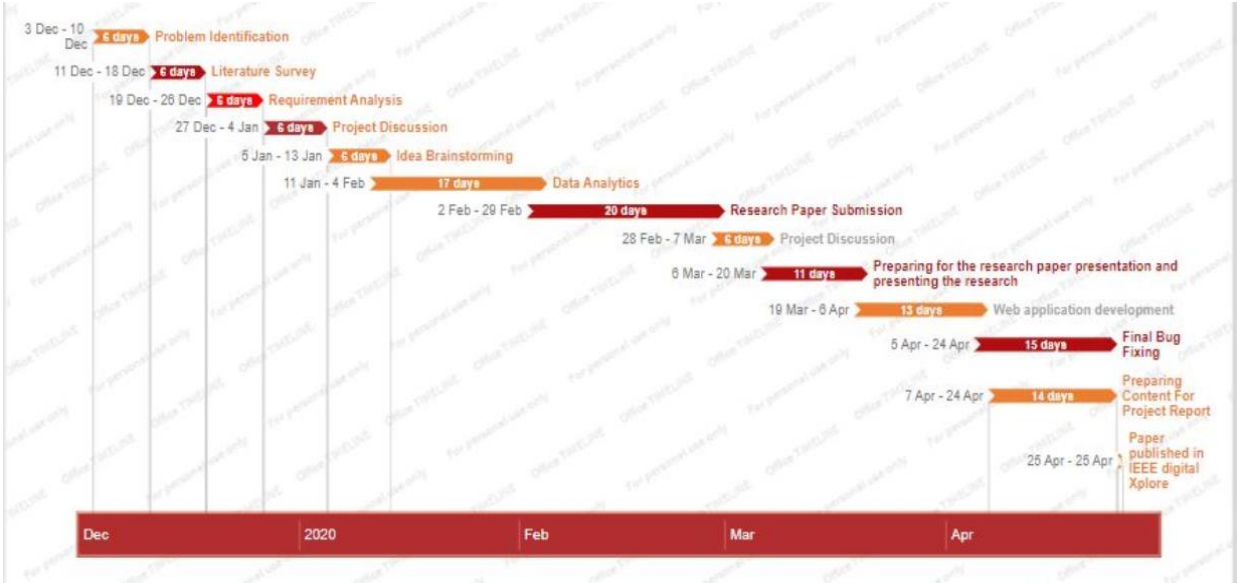

Fig 40 – Gantt Chart

6. PROJECT DEMONSTRATION

Implementation of this project is carried out in two phases. Phase 1 consist of data analysis and phase 2 involves web application development.

Phase 1 - Data analysis of deals with understanding of Indian Liver Patient Data using diverse techniques such as data exploration, visualization, cleaning, feature selection, classifier selection and evaluation measures.

Phase 2 – Web application development for liver disease prediction which facilitates the user to login/ sign up, viewing and editing the profile, predict the presence or absence liver disease using various parameter values which is calculated using machine learning algorithms and view the history of predictions of liver disease.

6.1. Phase 1 – Data Analysis

6.1.1. Statistical Analysis of Indian Liver Patient Dataset (ILPD)

Table 4
Statistics Values for different attributes

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Proteins	Albumin	Albumin_and_Globulin_Ratio
count	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000
mean	44.746141	0.756432	3.298799	1.486106	290.576329	80.713551	109.910806	6.483190	3.141852	0.946947
std	16.189833	0.429603	6.209522	2.808498	242.937989	182.620356	288.918529	1.085451	0.795519	0.318495
min	4.000000	0.000000	0.400000	0.100000	63.000000	10.000000	10.000000	2.700000	0.900000	0.300000
25%	33.000000	1.000000	0.800000	0.200000	175.500000	23.000000	25.000000	5.800000	2.600000	0.700000
50%	45.000000	1.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000	3.100000	0.930000
75%	58.000000	1.000000	2.600000	1.300000	298.000000	60.500000	87.000000	7.200000	3.800000	1.100000
max	90.000000	1.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.600000	5.500000	2.800000

Table clearly describes different statistic measures such as count, mean, standard deviation, quartiles, minimum value and maximum value.

From this table we can observe high variation in mean value and maximum value indicating the presence of outlier or anomaly value which deviates significantly.

6.1.2. Histogram view of data

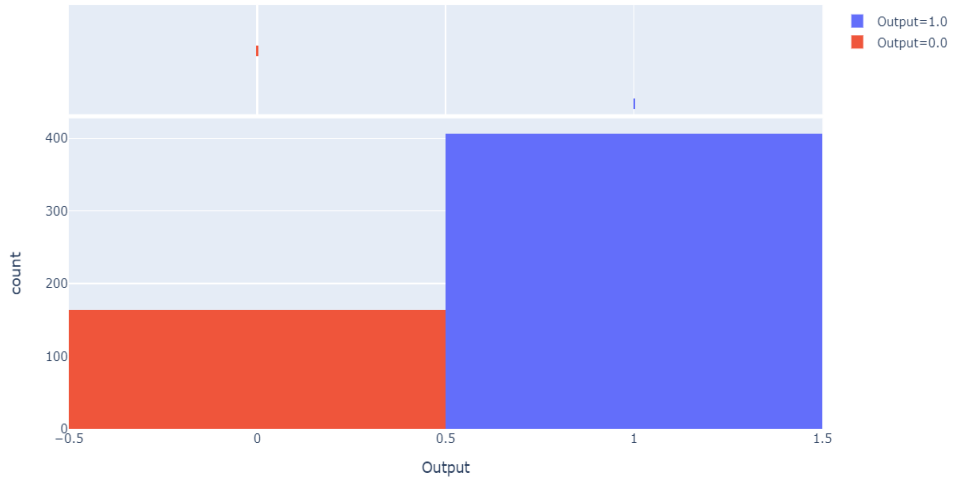

Fig 41 - Patients with and without liver disease

The above figure represents the patients affected and not affected by liver disease. From the plotted histogram, 416 are liver disease patients and 167 non liver disease patients. The data is biased towards liver disease patients hence there is high chances that predictions will be highly biased towards liver disease patients. To overcome this problem resampling has to be performed to create better training and better predictions.

For the histograms, Output Value=1.0 represents patients with liver disease and Output value=0.0 represents patients without liver disease.

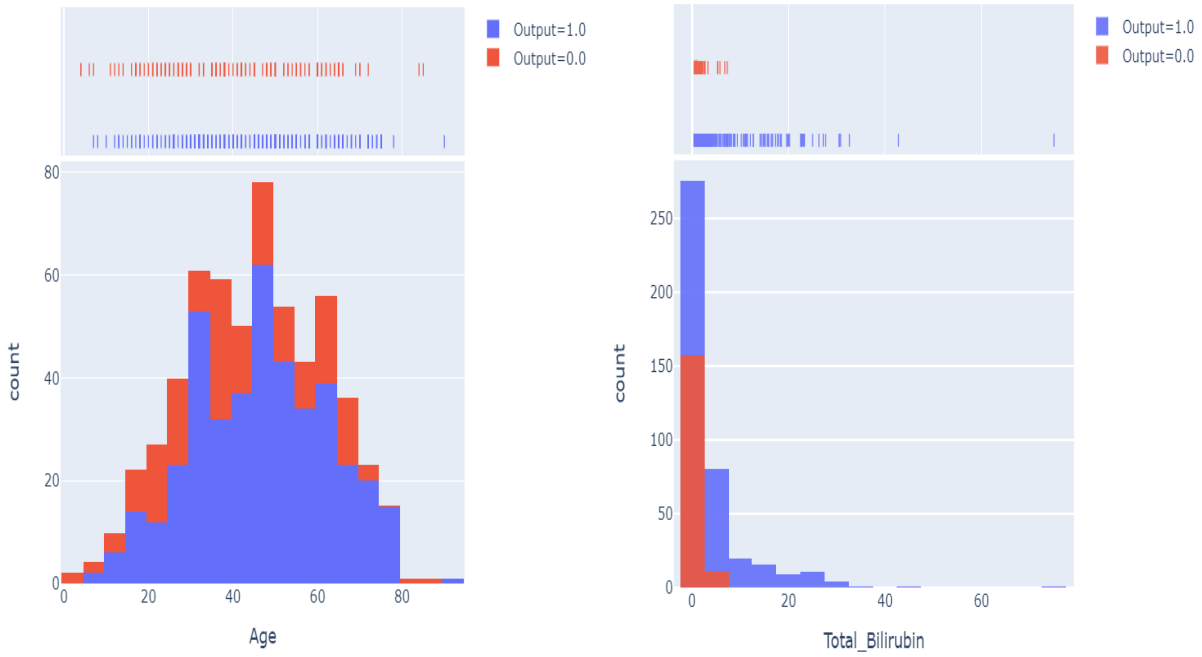

Fig 42 - Histogram of Age and Total_Bilirubin

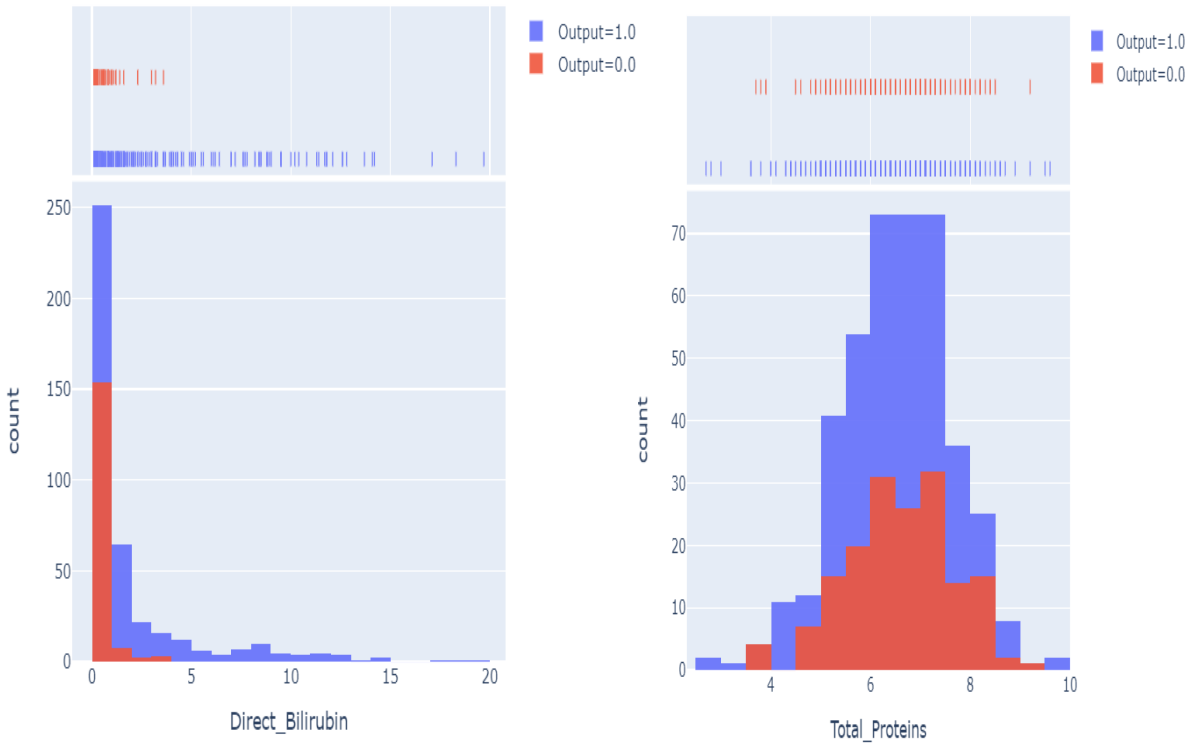

Fig 43 - Histograms of Direct_Bilirubin and Total_Proteins

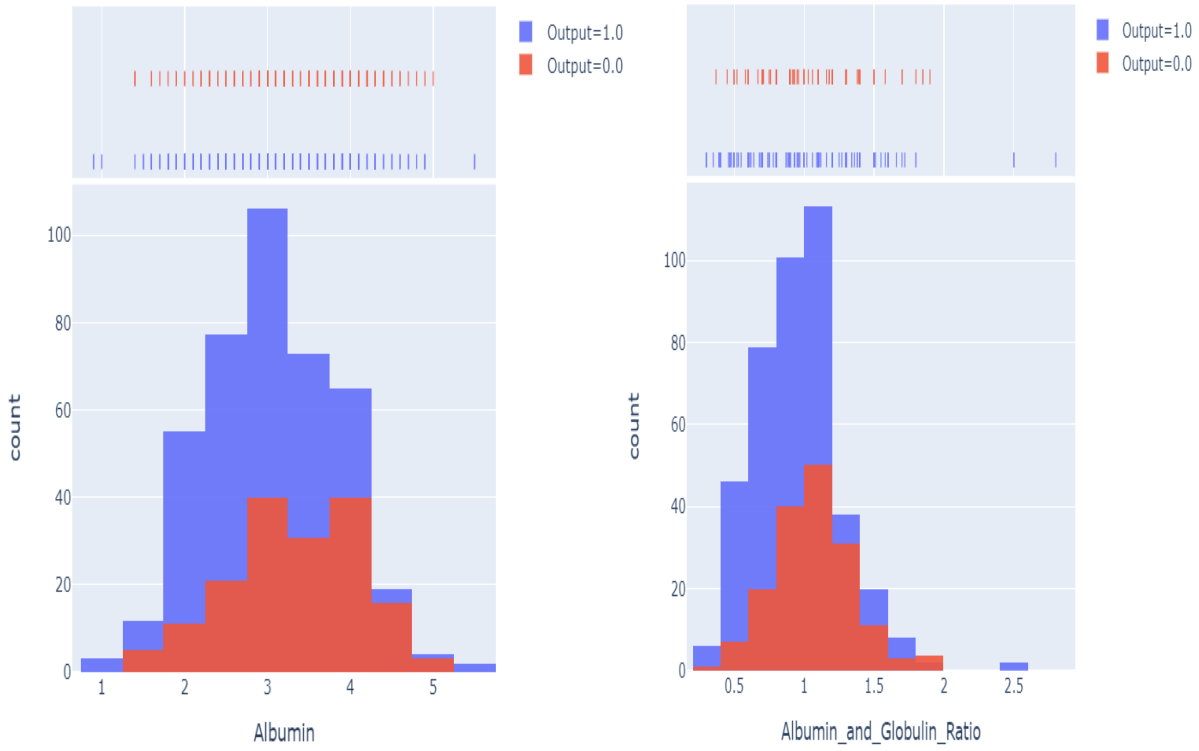

Fig 44 - Histogram of Albumin and Albumin_and_Globulin_Ratio

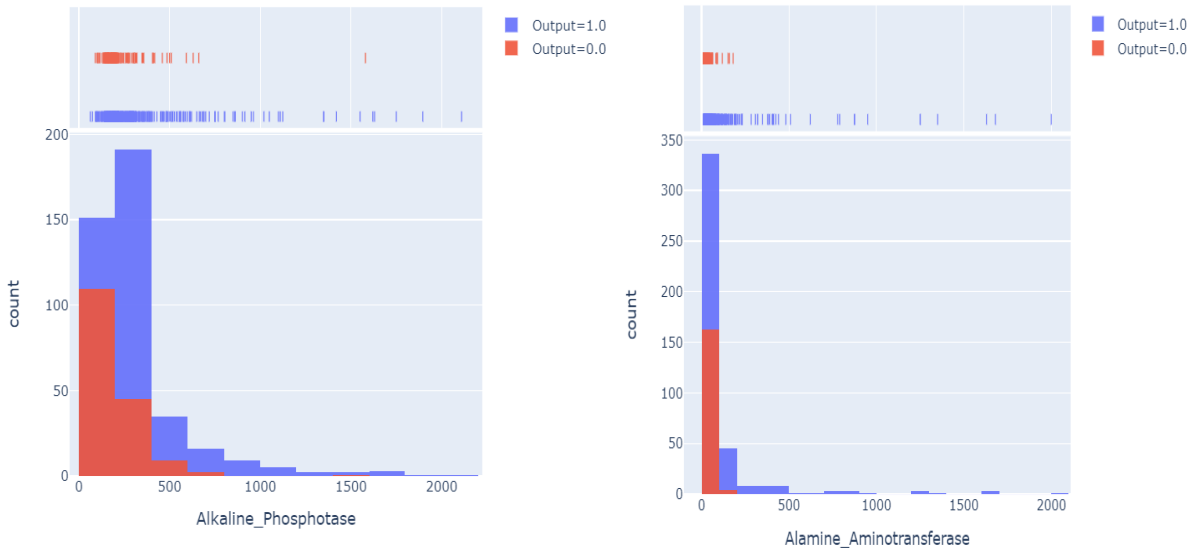

Fig 45 - Histograms of Alkaline_Phosphotase and Alamine_Aminotransferase

The above histograms, helps understand the presence of outlier values in few attributes and also there is presence of skewness for few attributes such as Total_Bilirubin, Direct_Bilirubin, Alkaline_Phosphotase and Alamine_Aminotransferase.

Imputation of missing values

```
Before Imputation -
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
Output                       0
dtype: int64
```

```
After Imputation -
Age                          0
Gender                       0
Total_Bilirubin              0
Direct_Bilirubin             0
Alkaline_Phosphotase         0
Alamine_Aminotransferase     0
Aspartate_Aminotransferase   0
Total_Proteins               0
Albumin                      0
Albumin_and_Globulin_Ratio   0
Output                       0
dtype: int64
```

Fig 46 - Imputation of missing values

Missing data reduces the statistical power hence it's always better to replace null values. In liver disease data set contains only 4 missing values for the attribute Albumin and Globulin Ratio which is filled with median values.

Feature Correlation

Correlation is a statistical technique that depicts relationship between variable pairs. Correlated features will not always worsen your model, but they will not always improve it

either. Removing highly correlated will help to make the learning algorithm faster, decrease harmful bias and interpretability of your model (Occam's razor)

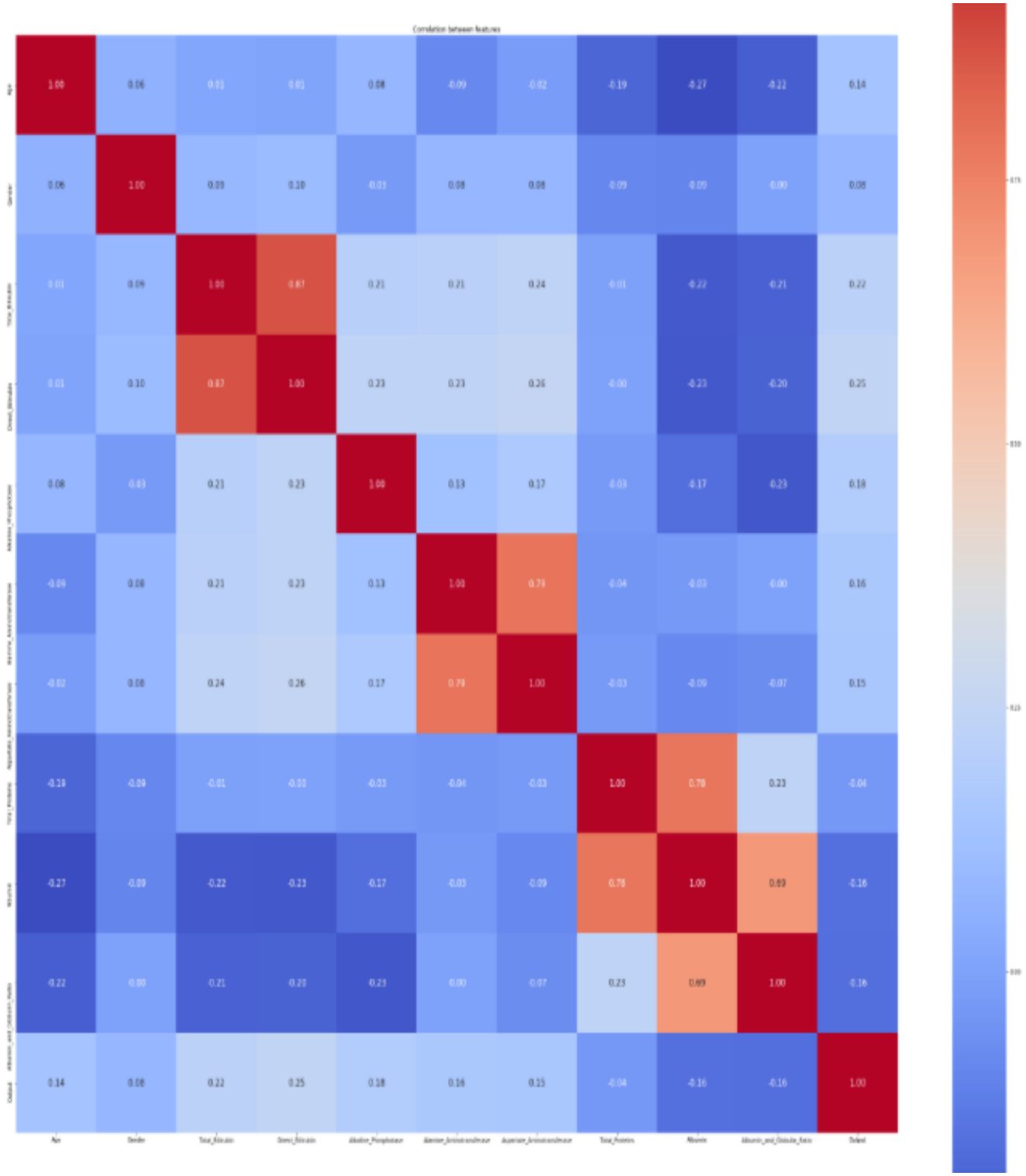

Fig 47 - Feature Correlation

From the figure, few pairs of variables are highly correlated. Highly positively correlated pair is Total_Bilirubin - Direct_Bilirubin with correlation value 0.87. Other

positively correlated value pairs are Alamine_Aminotransferase - Aspartate_Aminotransferase and Albumin - Total_Proteins with correlation value 0.79 and 0.78 respectively.

Boxplot before outlier removal

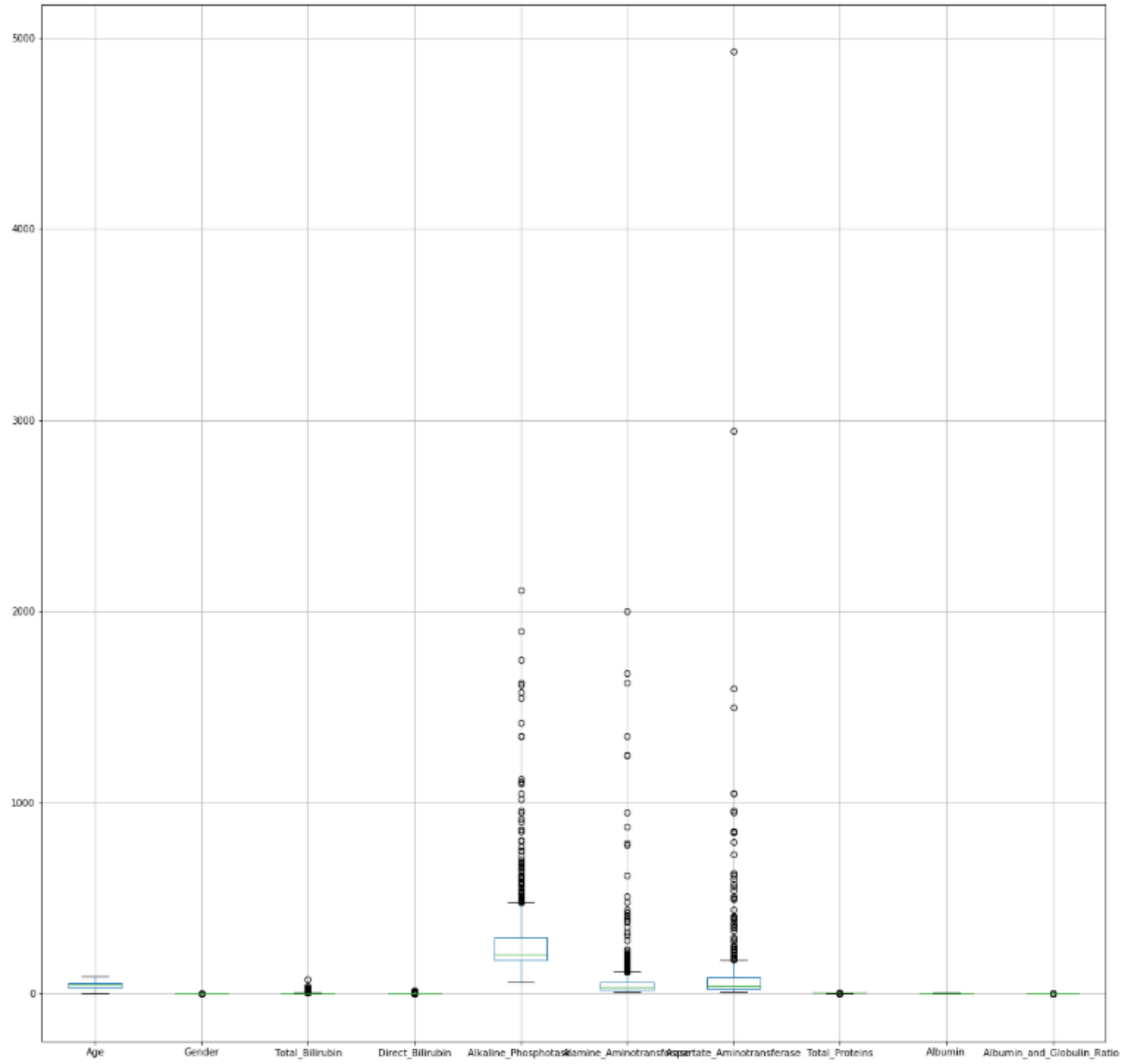

Fig 48 - Boxplot before outlier removal

Boxplot presented above displays the presence of anomalous value present in various attributes. Alkaline_Phosphotase, Alamine_Aminotransferase and Aspartate_Aminotranseferase indicates highly deviating values which has to be removed to or adjusted in order to create better prediction.

Two approached carried out Isolation Forest algorithm and adjusting the skewness of extreme values of certain attributes.

Outlier elimination using Isolation Forest

	Age	Gender	TotalBilirubin	DirectBilirubin
1	62.000000	1.000000	10.900000	5.500000
2	62.000000	1.000000	7.300000	4.100000
4	72.000000	1.000000	3.900000	2.000000
13	74.000000	0.000000	1.100000	0.400000
16	38.000000	1.000000	1.800000	0.800000
..	...	...	...	...
832	64.910361	0.000000	0.782960	0.200000
880	25.664104	0.204931	2.013096	0.656548
898	17.670010	0.167503	0.533501	0.116750
899	17.300880	0.000000	0.590264	0.130088
919	44.906261	0.000000	0.698661	0.198661

	AlkalinePhosphotase	AlamineAminotransferase	AspartateAminotransferase
1	699.000000	64.000000	100.000000
2	490.000000	60.000000	68.000000
4	195.000000	27.000000	59.000000
13	214.000000	22.000000	30.000000
16	342.000000	168.000000	441.000000
..	...	...	...
832	147.192811	41.829598	65.228754
880	504.286585	27.385207	41.950689
898	206.837513	25.654965	21.335005
899	203.893839	29.805280	24.008801
919	164.013391	21.013391	52.745566

	TotalProteins	Albumin	AlbuminGlobulinRatio
1	7.500000	3.200000	0.740000
2	7.000000	3.300000	0.890000
4	7.300000	2.400000	0.400000
13	8.100000	4.100000	1.000000
16	7.600000	4.400000	1.300000
..	...	...	...
832	7.735434	4.439916	1.314799
880	6.695069	2.879507	0.740986
898	7.133501	4.432999	1.616249
899	6.919472	4.199120	1.537525
919	4.518748	1.420087	0.456026

[203 rows x 10 columns]

Fig 49 - Screenshot of outlier values detected using Isolation forest

Using isolation forest, 203 outlier value records where found out in Indian Liver Patient Dataset. Data skewness for extreme values was adjusted using quartile range to help learning better.

Resampling using Genetic SMOTE

```
Before resampling (Features) -  (570, 10)
Before resampling (Target) -  (570,)
After resampling (Features) -  (1050, 10)
After resampling (Target) -  (1050,)
```

Boxplot after outlier removal

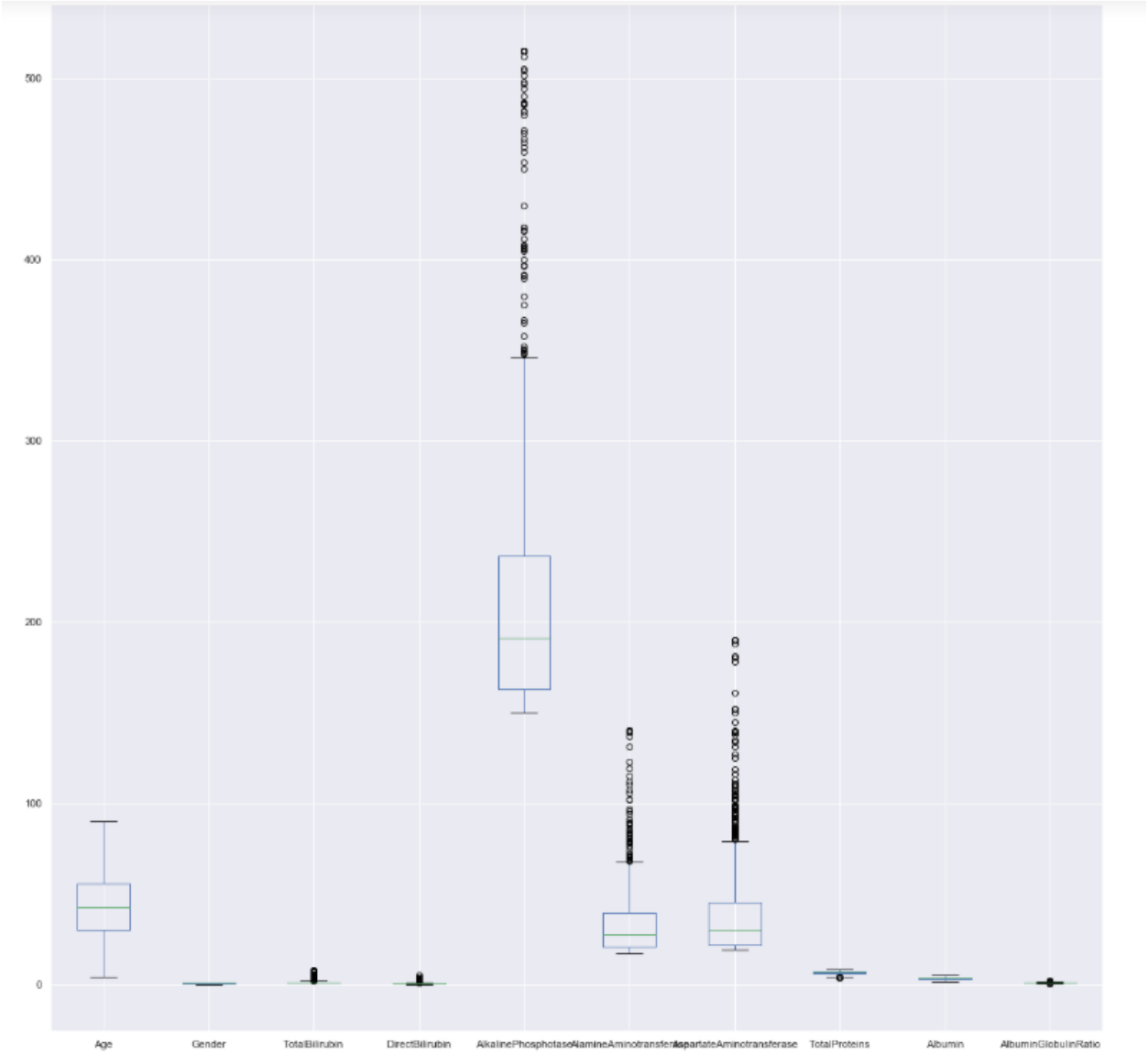

Fig 50 - Boxplot after outlier removal

After outlier removal using isolation forest and reducing data skewness due to extreme values. It can be observed that there is considerable reduction in the anomaly value for the attributes Alkaline_Phosphotase, Alamine_Aminotransferase and Aspartate_Aminotranseferase.

Genetic Algorithm and XGBoost classifier for feature selection

```
gen     nevals  avg             min             max
0       25      0.762005        0.724862        0.784171
1       12      0.772841        0.744657        0.792365
2       11      0.783179        0.77249         0.792365
3       12      0.78787         0.769102        0.794677
4       13      0.790356        0.77497         0.794677
5       15      0.787528        0.7574          0.794677
6       9       0.790735        0.769082        0.794677
7       20      0.791844        0.764451        0.794677
8       13      0.794629        0.792365        0.795806
9       15      0.792813        0.769082        0.795806
10      12      0.79434         0.783082        0.795806
11      16      0.793906        0.776005        0.795806
12      21      0.795204        0.783008        0.795806
13      9       0.794694        0.776106        0.795806
14      17      0.795198        0.780616        0.795806
15      18      0.795344        0.792318        0.798125
16      16      0.795944        0.792318        0.798125
17      14      0.79515         0.7656          0.798125
18      17      0.795849        0.776005        0.798125
19      16      0.794922        0.764478        0.798125
20      18      0.798125        0.798125        0.798125
21      16      0.795248        0.77257         0.798125
22      13      0.798124        0.798105        0.798125
23      18      0.794646        0.77257         0.798125
24      18      0.796775        0.77257         0.798125
25      15      0.794506        0.763302        0.798125
Best Accuracy:  0.7981247479499933
Number of Features in Subset:   8
Feature Subset: ['Age', 'Gender', 'TotalBilirubin', 'DirectBilirubin', 'AlamineAminotransferase', 'AspartateAminotransferase',
'Albumin', 'AlbuminGlobulinRatio']
```

Fig 51 - Screenshots of Feature Selection

Feature Selection in implemented using combination of Genetic Algorithm and XGBoost algorithm. The algorithm is applied using population size = 25 and generation size = 25 with a cross validation split = 5. This is carried out to fetch the best features required for the analysis thereby reducing the time and increasing the performance. TotalProteins and AlkalinePhosphotase are eliminated using feature selection. TotalBilirubin is eliminated due to its high correlation value with Directbilirubin.

Performance metrics of various machine learning algorithms before feature selection using genetic algorithm and anomaly elimination

	precision	recall	f1-score	support
0.0	0.53	0.30	0.38	33
1.0	0.76	0.89	0.82	81
accuracy			0.72	114
macro avg	0.64	0.60	0.60	114
weighted avg	0.69	0.72	0.69	114

Fig 52 – KNN

	precision	recall	f1-score	support
0.0	0.60	0.27	0.37	33
1.0	0.76	0.93	0.83	81
accuracy			0.74	114
macro avg	0.68	0.60	0.60	114
weighted avg	0.71	0.74	0.70	114

Fig 53 – Logistic Regression

	precision	recall	f1-score	support
0.0	0.43	0.48	0.46	33
1.0	0.78	0.74	0.76	81
accuracy			0.67	114
macro avg	0.61	0.61	0.61	114
weighted avg	0.68	0.67	0.67	114

Fig 54 – Decision Tree

	precision	recall	f1-score	support
0.0	0.57	0.39	0.46	33
1.0	0.78	0.88	0.83	81
accuracy			0.74	114
macro avg	0.67	0.64	0.64	114
weighted avg	0.72	0.74	0.72	114

Fig 54 – Random Forest Tree

	precision	recall	f1-score	support
0.0	0.42	0.45	0.43	33
1.0	0.77	0.74	0.75	81
accuracy			0.66	114
macro avg	0.59	0.60	0.59	114
weighted avg	0.67	0.66	0.66	114

Fig 56 – Gradient Boosting Algorithm

	precision	recall	f1-score	support
0.0	0.41	0.27	0.33	33
1.0	0.74	0.84	0.79	81
accuracy			0.68	114
macro avg	0.57	0.56	0.56	114
weighted avg	0.64	0.68	0.65	114

Fig 57 – AdaBoost Algorithm

	precision	recall	f1-score	support
0.0	0.48	0.36	0.41	33
1.0	0.76	0.84	0.80	81
accuracy			0.70	114
macro avg	0.62	0.60	0.61	114
weighted avg	0.68	0.70	0.69	114

Fig 58 – XGBoost Algorithm

	precision	recall	f1-score	support
0.0	0.48	0.48	0.48	33
1.0	0.79	0.79	0.79	81
accuracy			0.70	114
macro avg	0.64	0.64	0.64	114
weighted avg	0.70	0.70	0.70	114

Fig 59 – LightGBM Algorithm

	precision	recall	f1-score	support
0.0	0.82	0.92	0.87	161
1.0	0.85	0.70	0.77	109
accuracy			0.83	270
macro avg	0.84	0.81	0.82	270
weighted avg	0.83	0.83	0.83	270

Fig 60 – Stacking Estimator

```
Training set score: 0.712719
0.7105263157894737
```

	precision	recall	f1-score	support
0.0	0.00	0.00	0.00	33
1.0	0.71	1.00	0.83	81
accuracy			0.71	114
macro avg	0.36	0.50	0.42	114
weighted avg	0.50	0.71	0.59	114

Fig 61 – Multilayer Perceptron

Performance metrics of various machine learning algorithms after feature selection using genetic algorithm and anomaly elimination

	precision	recall	f1-score	support
0.0	0.83	0.96	0.89	130
1.0	0.77	0.40	0.52	43
accuracy			0.82	173
macro avg	0.80	0.68	0.71	173
weighted avg	0.81	0.82	0.80	173

Fig 61 – Multilayer Perceptron

	precision	recall	f1-score	support
0.0	0.79	0.97	0.87	130
1.0	0.71	0.23	0.35	43
accuracy			0.79	173
macro avg	0.75	0.60	0.61	173
weighted avg	0.77	0.79	0.74	173

Fig 62 – KNN

	precision	recall	f1-score	support
0.0	0.78	0.94	0.85	130
1.0	0.53	0.21	0.30	43
accuracy			0.76	173
macro avg	0.66	0.57	0.58	173
weighted avg	0.72	0.76	0.72	173

Fig 63 – Logistic Regression

	precision	recall	f1-score	support
0.0	0.89	0.89	0.89	130
1.0	0.67	0.67	0.67	43
accuracy			0.84	173
macro avg	0.78	0.78	0.78	173
weighted avg	0.84	0.84	0.84	173

Fig 64 – Decision Tree

	precision	recall	f1-score	support
0.0	0.90	0.95	0.93	130
1.0	0.83	0.67	0.74	43
accuracy			0.88	173
macro avg	0.86	0.81	0.83	173
weighted avg	0.88	0.88	0.88	173

Fig 65 – Random Forest Tree

	precision	recall	f1-score	support
0.0	0.89	0.89	0.89	130
1.0	0.67	0.67	0.67	43
accuracy			0.84	173
macro avg	0.78	0.78	0.78	173
weighted avg	0.84	0.84	0.84	173

Fig 66 - Gradient Boosting Algorithm

	precision	recall	f1-score	support
0.0	0.88	0.90	0.89	130
1.0	0.68	0.63	0.65	43
accuracy			0.83	173
macro avg	0.78	0.76	0.77	173
weighted avg	0.83	0.83	0.83	173

Fig 67 – AdaBoost Algorithm

	precision	recall	f1-score	support
0.0	0.91	0.90	0.91	130
1.0	0.71	0.74	0.73	43
accuracy			0.86	173
macro avg	0.81	0.82	0.82	173
weighted avg	0.86	0.86	0.86	173

Fig 68 – XGBoost Algorithm

	precision	recall	f1-score	support
0.0	0.90	0.91	0.90	130
1.0	0.71	0.70	0.71	43
accuracy			0.86	173
macro avg	0.81	0.80	0.81	173
weighted avg	0.85	0.86	0.85	173

Fig 69 – Light GBM Algorithm

	precision	recall	f1-score	support
0.0	0.87	0.94	0.90	154
1.0	0.80	0.65	0.71	62
accuracy			0.85	216
macro avg	0.83	0.79	0.81	216
weighted avg	0.85	0.85	0.85	216

Fig 70– Stacking Estimator

Confusion Matrix of various classification algorithms

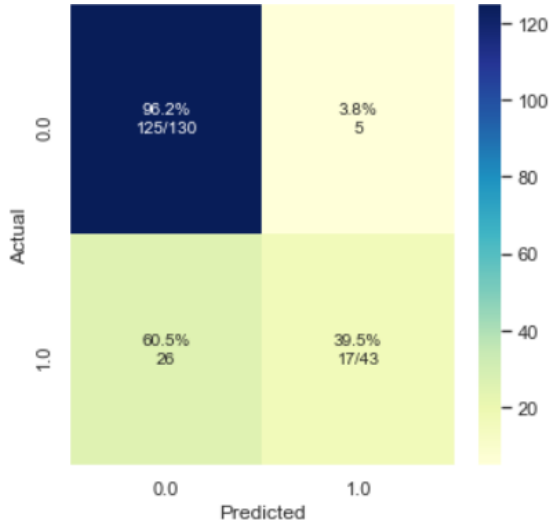

Fig 71 – Multilayer Perceptron

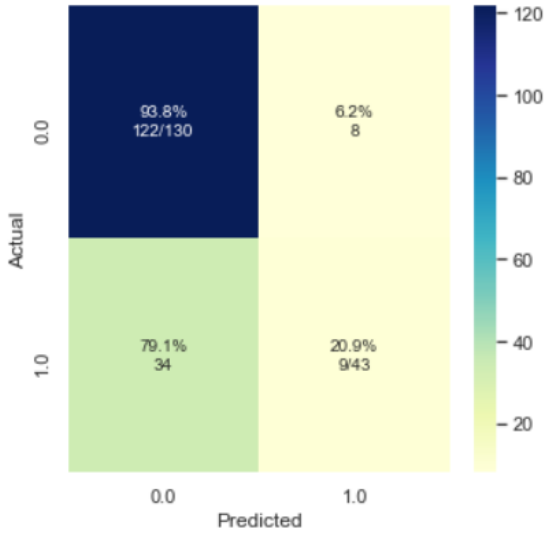

Fig 72– Logistic Regression

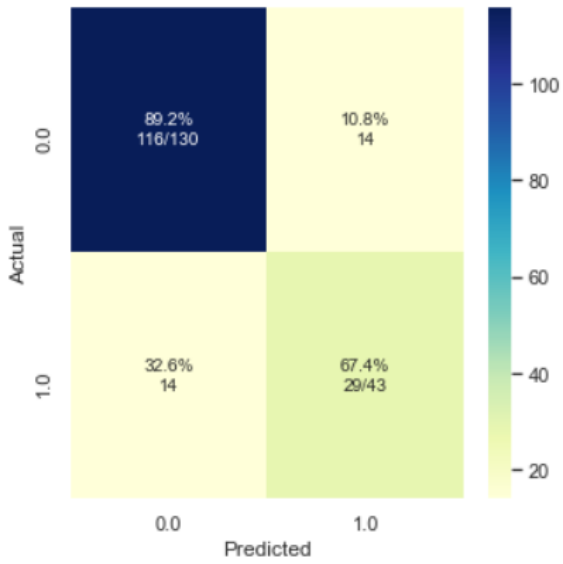

Fig 73– Decision Tree

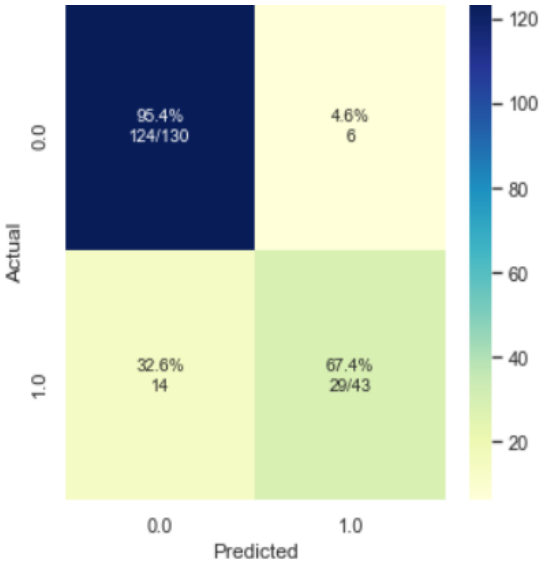

Fig 74– Random Forest Tree

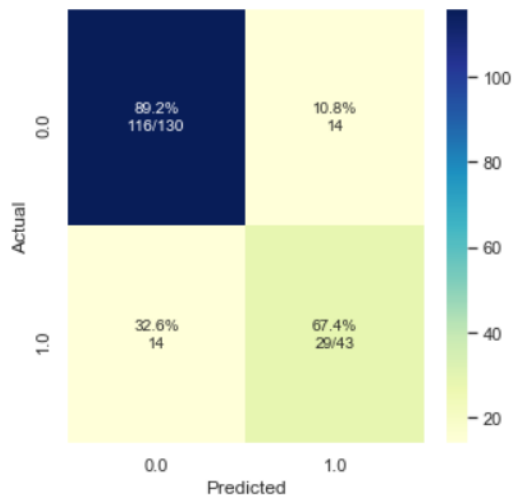

Fig 75 – Gradient Boosting Algorithm

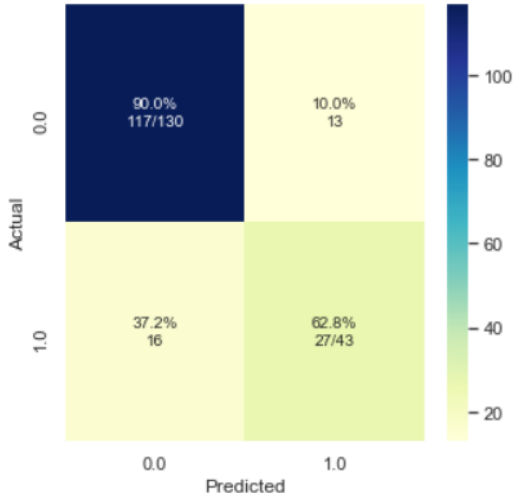

Fig 76 – AdaBoost Algorithm

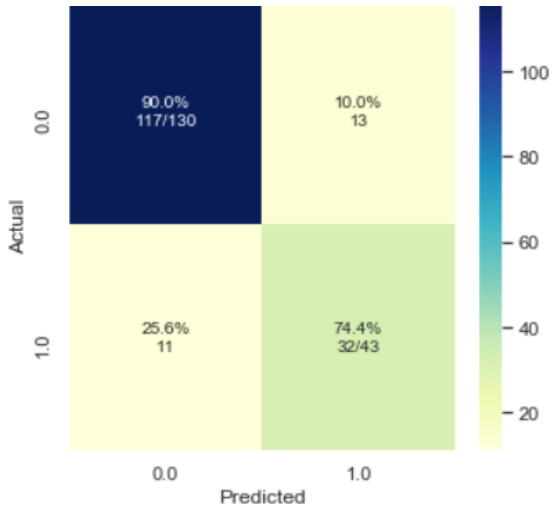

Fig 77 – XGBoost Algorithm

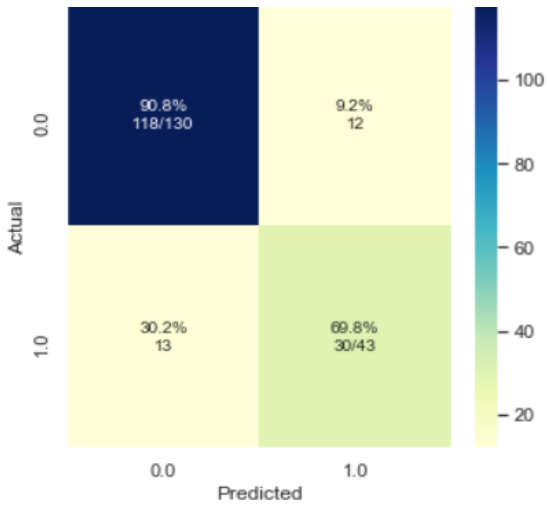

Fig 78 – Light GBM Algorithm

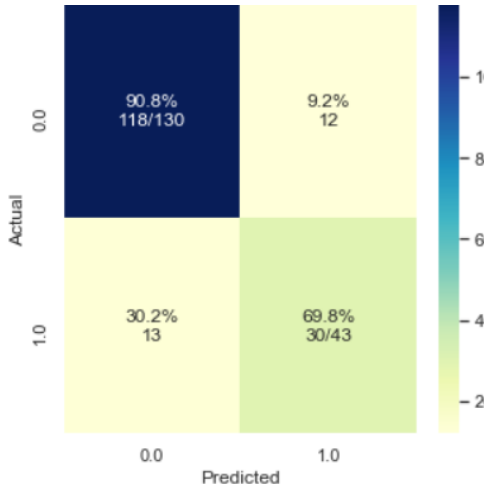

Fig 79 – Stacking Estimator

ROC Curve of Classification Algorithms

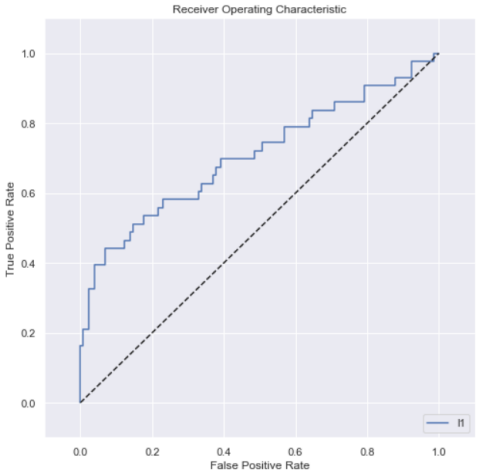

Fig 80 – MLP

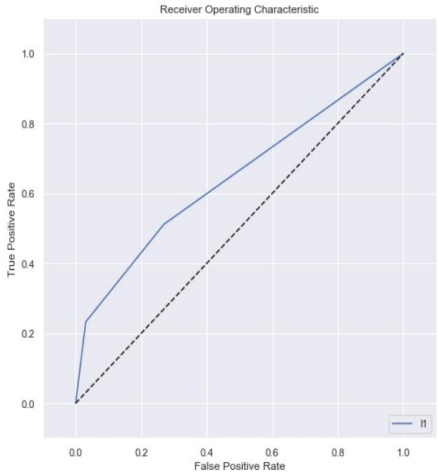

Fig 81 – KNN

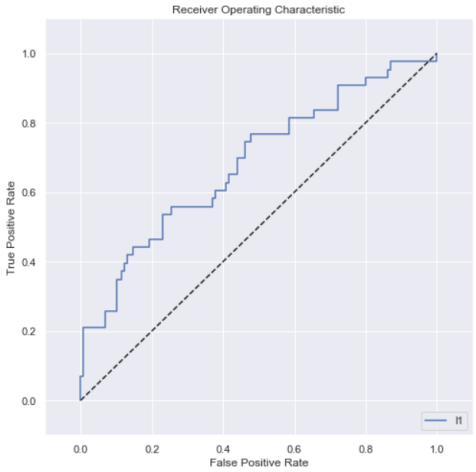

Fig 82 – Logistic Regression

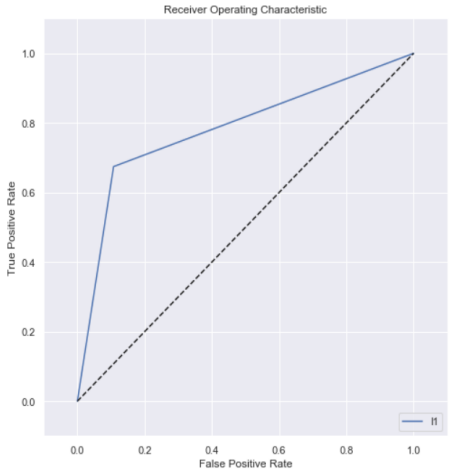

Fig 83 – Decision Tree

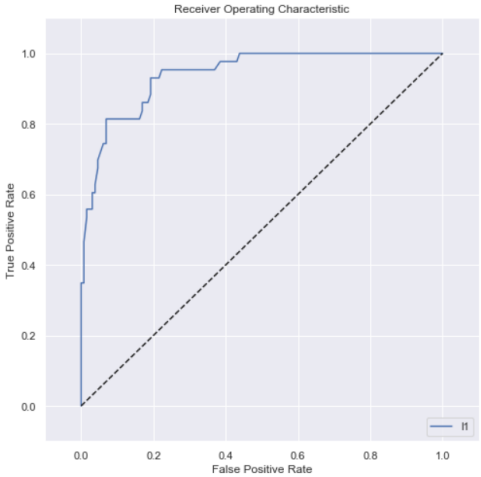

Fig 84 – Random Forest Tree

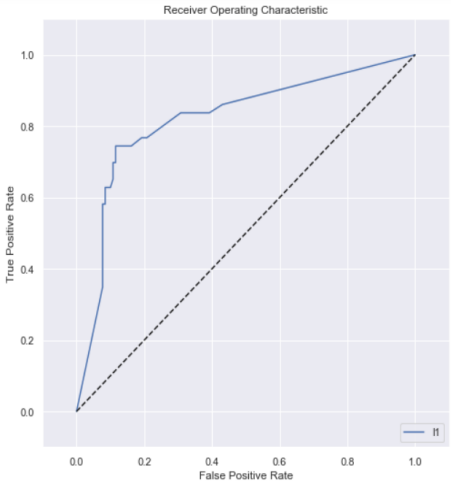

Fig 85 – Gradient Boosting

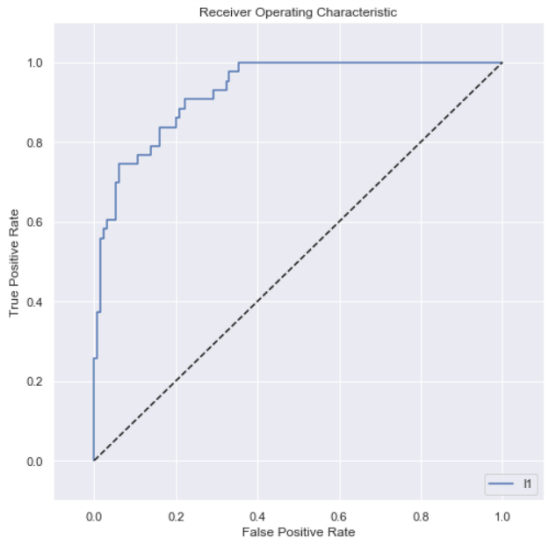

Fig 86 – XGBoost Algorithm

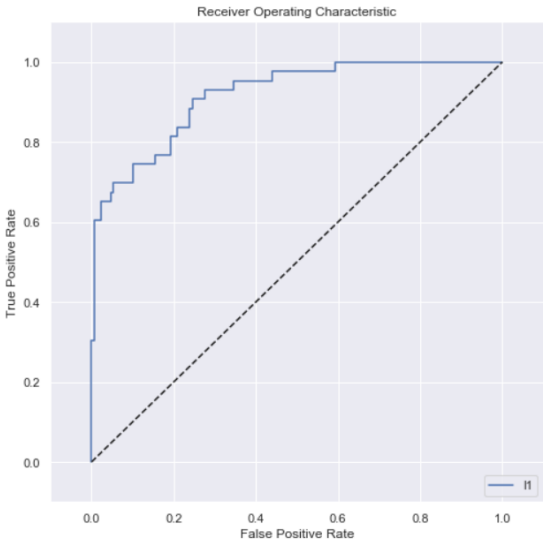

Fig 87 – Light GBM Algorithm

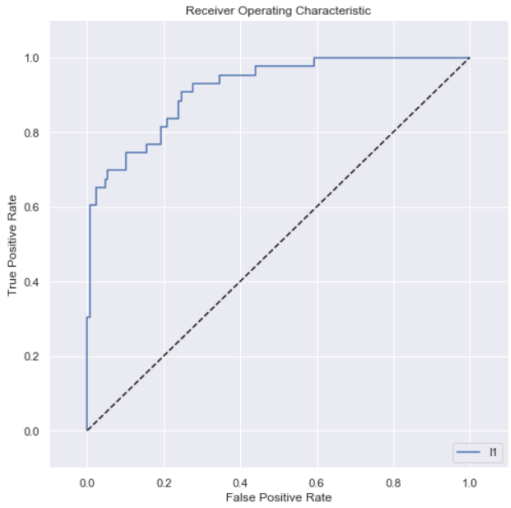

Fig 88– Stacking Estimator

Phase 2 – Web application

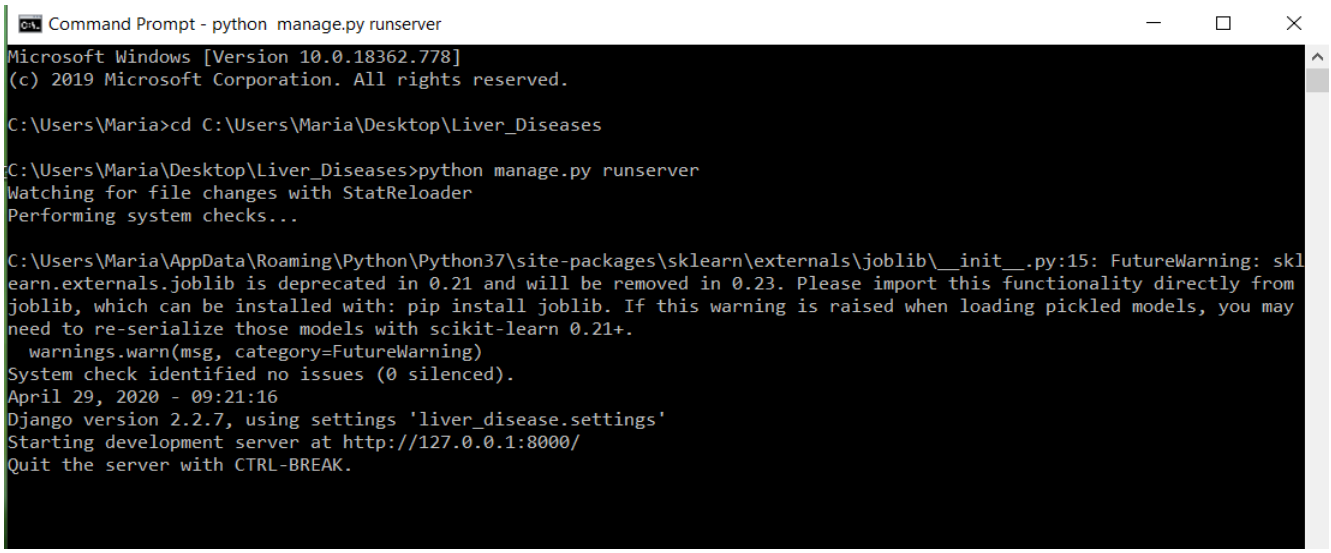

Fig 89 – Running the Django web application

Fig 90 - Screenshot of About Page

Fig 91 - Screenshot of Sign-Up Page

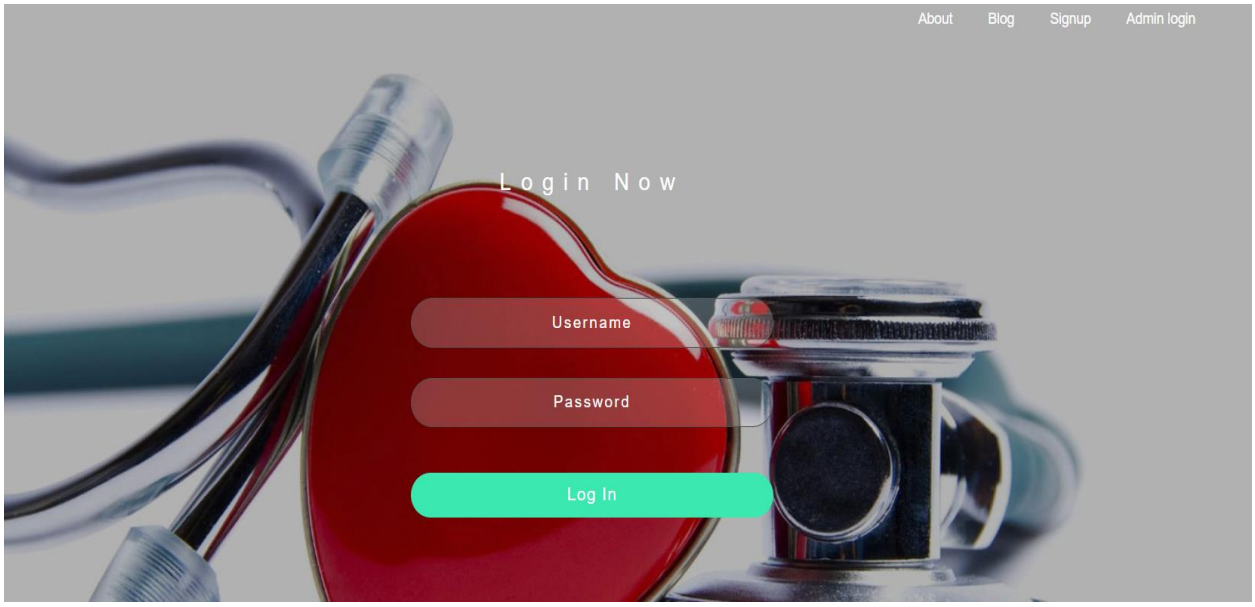

Fig 92 - Screenshot of Login Page

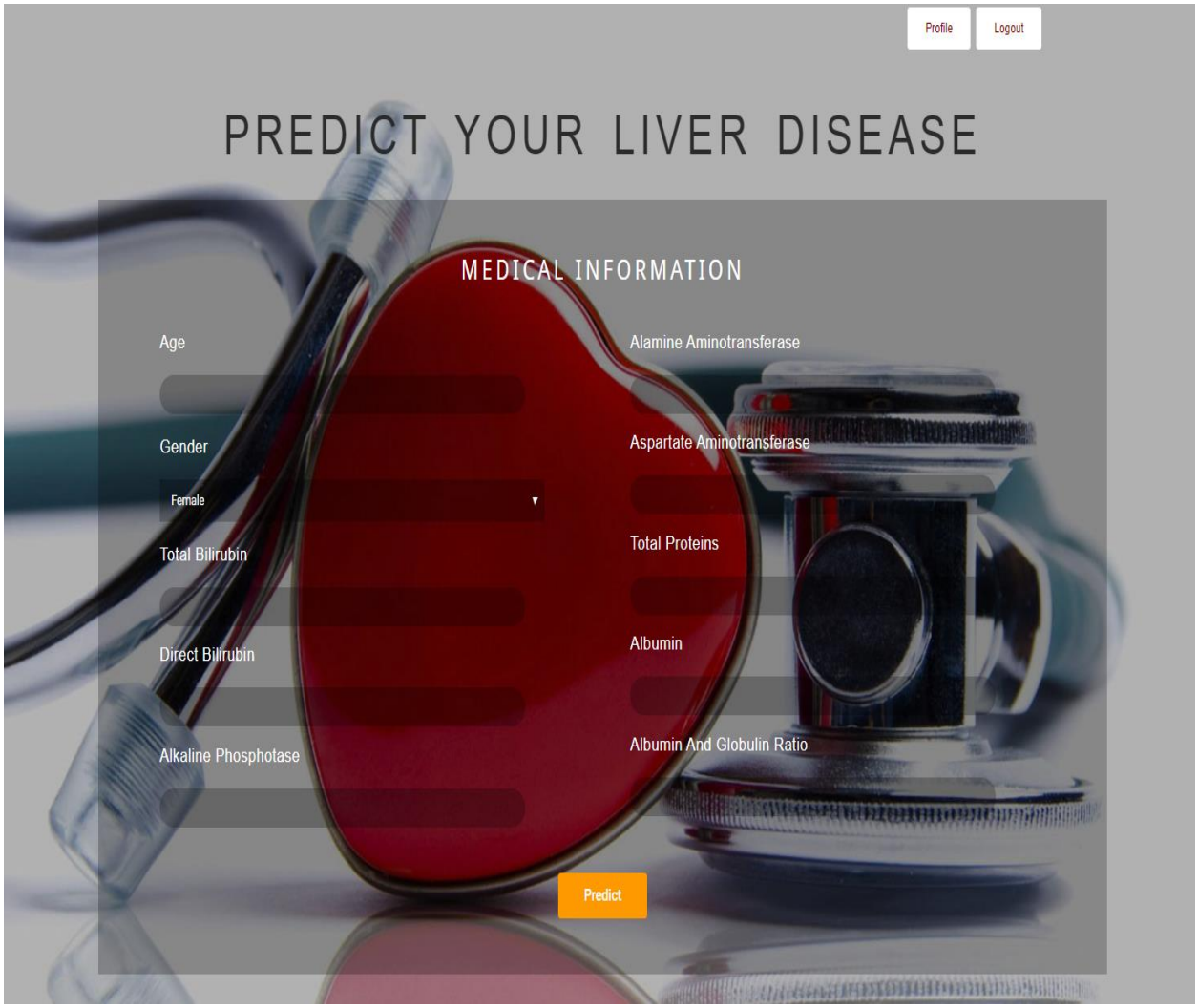

Fig 93 - Screenshot of Predict Disease

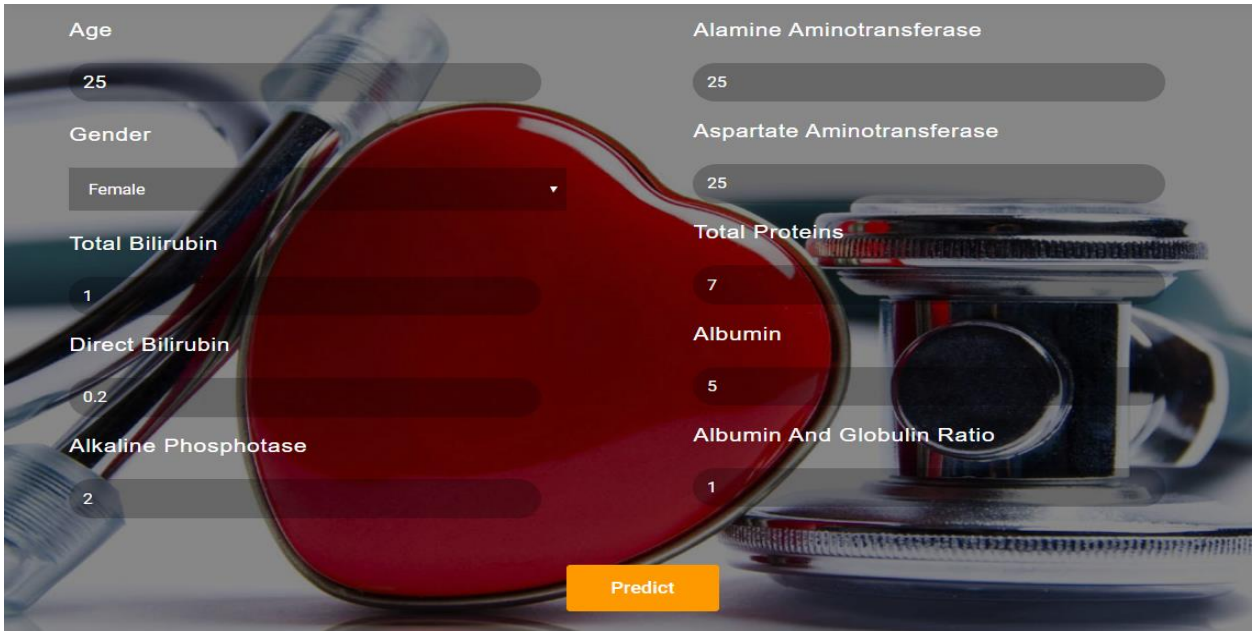

Fig 94 – Nominal (Normal) Range of values

Fig 95 - Output Predictions for Nominal (Normal) Values

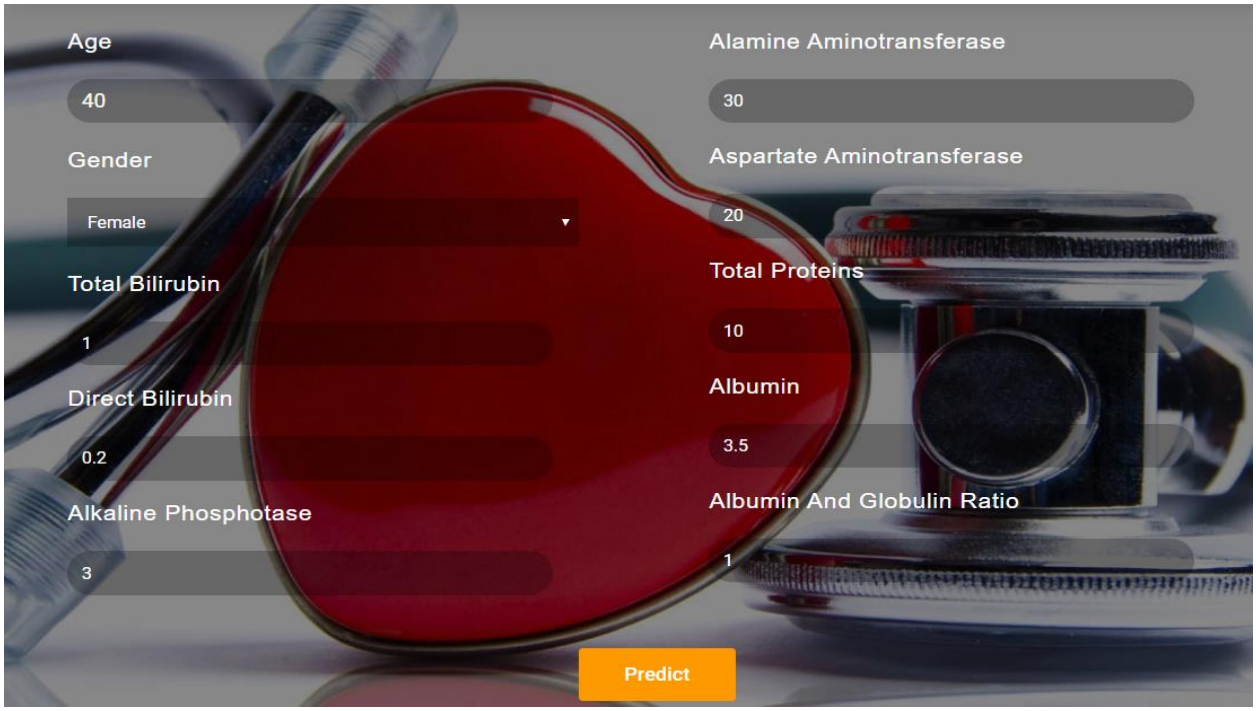

Fig 96 – Moderate Range of Values

You have a risk of liver disease !!!

Algorithm	Risk of Heart Disease
Logistic Regression	0.0
Naive Bayes	0.0
Decision Tree	0.0
Gradient Boosting Algorithm	0.0
Random Forest Tree	1.0
XGboost	0.0
LightGBM	1.0
AdaBoost Algorithm	1.0
MultiLayer Perceptron	1.0
KNN	0.0
Stacking Estimator 1	1.0
Stacking Estimator 2	1.0

Fig 97 – Output Predictions for Moderate Range

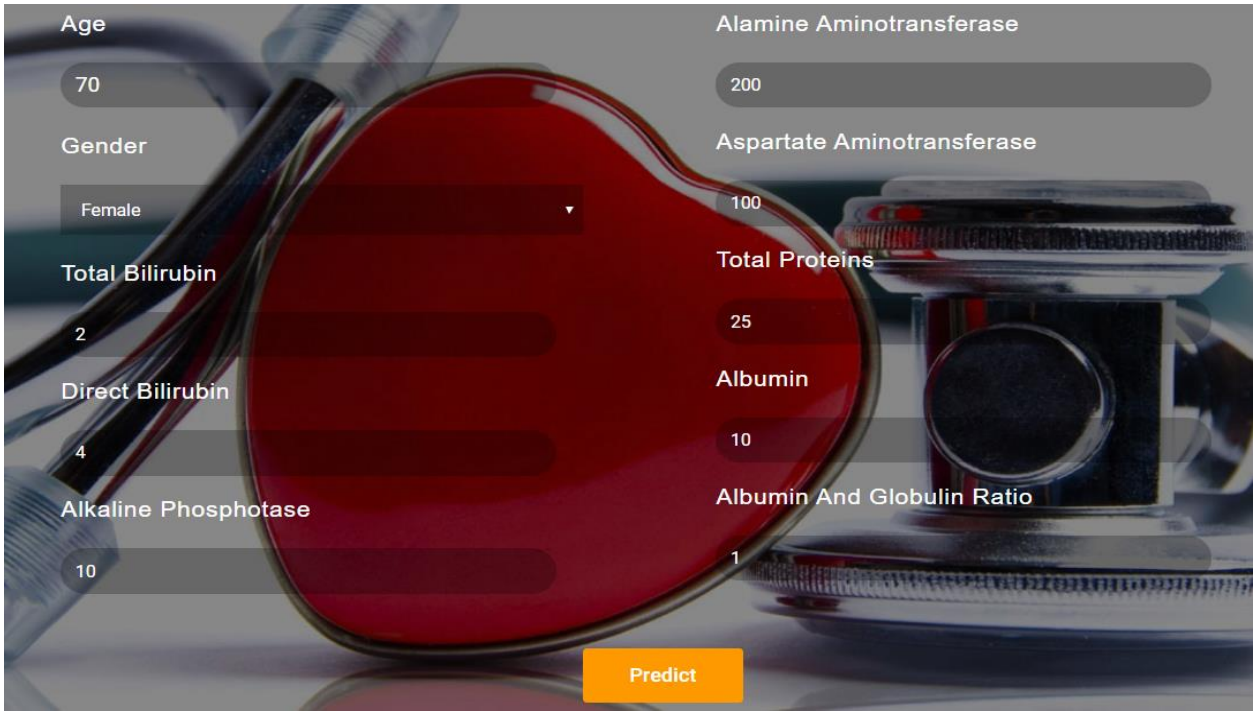

Fig 98 – Extreme Values

Algorithm	Risk of Heart Disease
Logistic Regression	1.0
Naive Bayes	1.0
Decision Tree	1.0
Gradient Boosting Algorithm	1.0
Random Forest Tree	1.0
XGboost	1.0
LightGBM	1.0
AdaBoost Algorithm	1.0
MultiLayer Perceptron	1.0
KNN	1.0
Stacking Estimator 1	1.0
Stacking Estimator 2	1.0

Fig 99 – Output predictions for Extreme Values

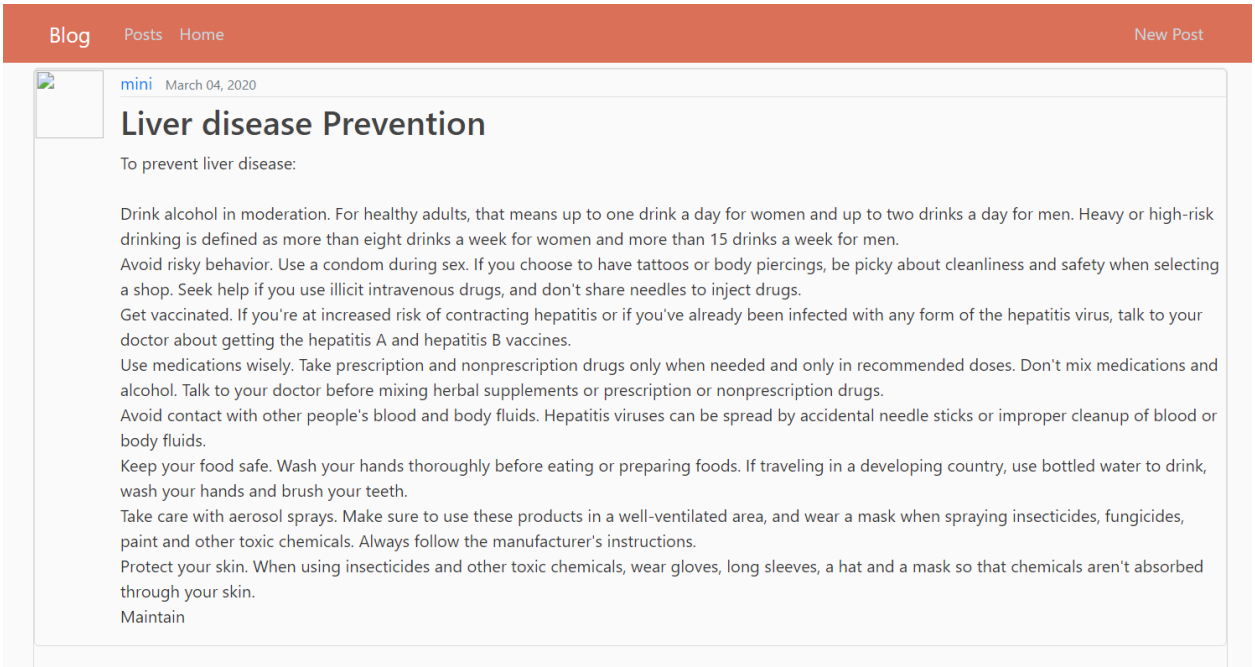

Fig 100 – Blog containing the posts

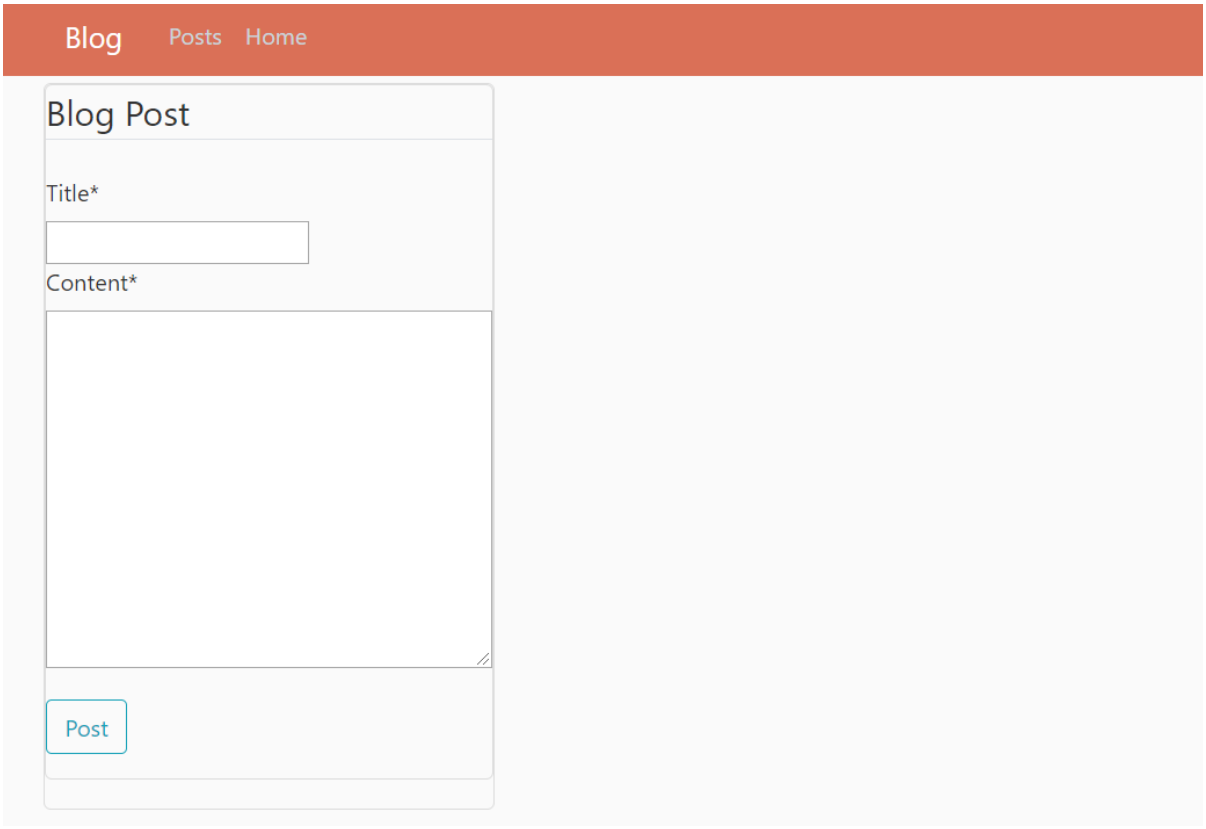

Fig 101– Blog for writing new post

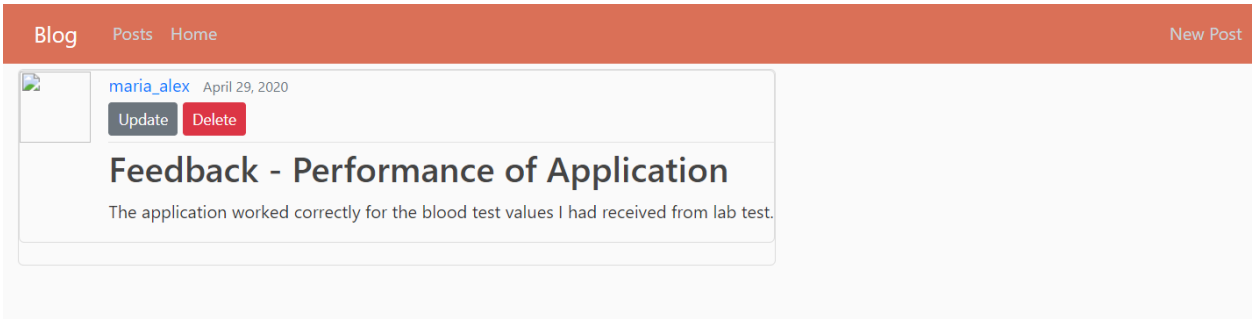

Fig 102 – Delete the Post

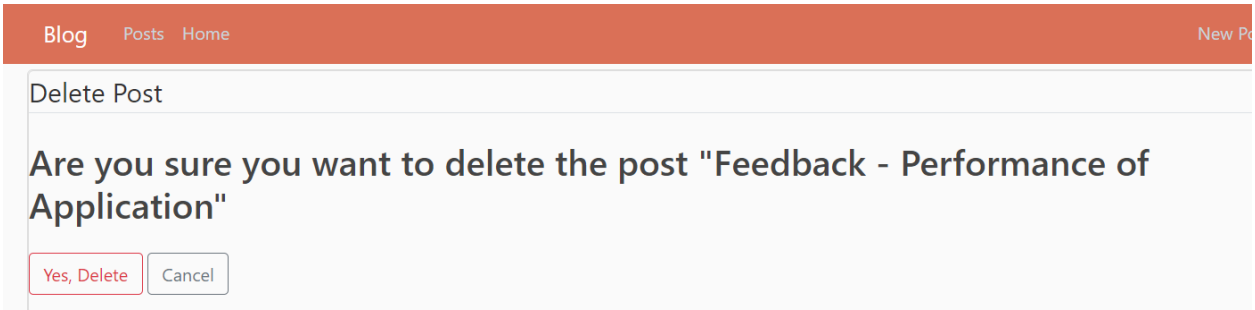

Fig 103 – Deleting the Post

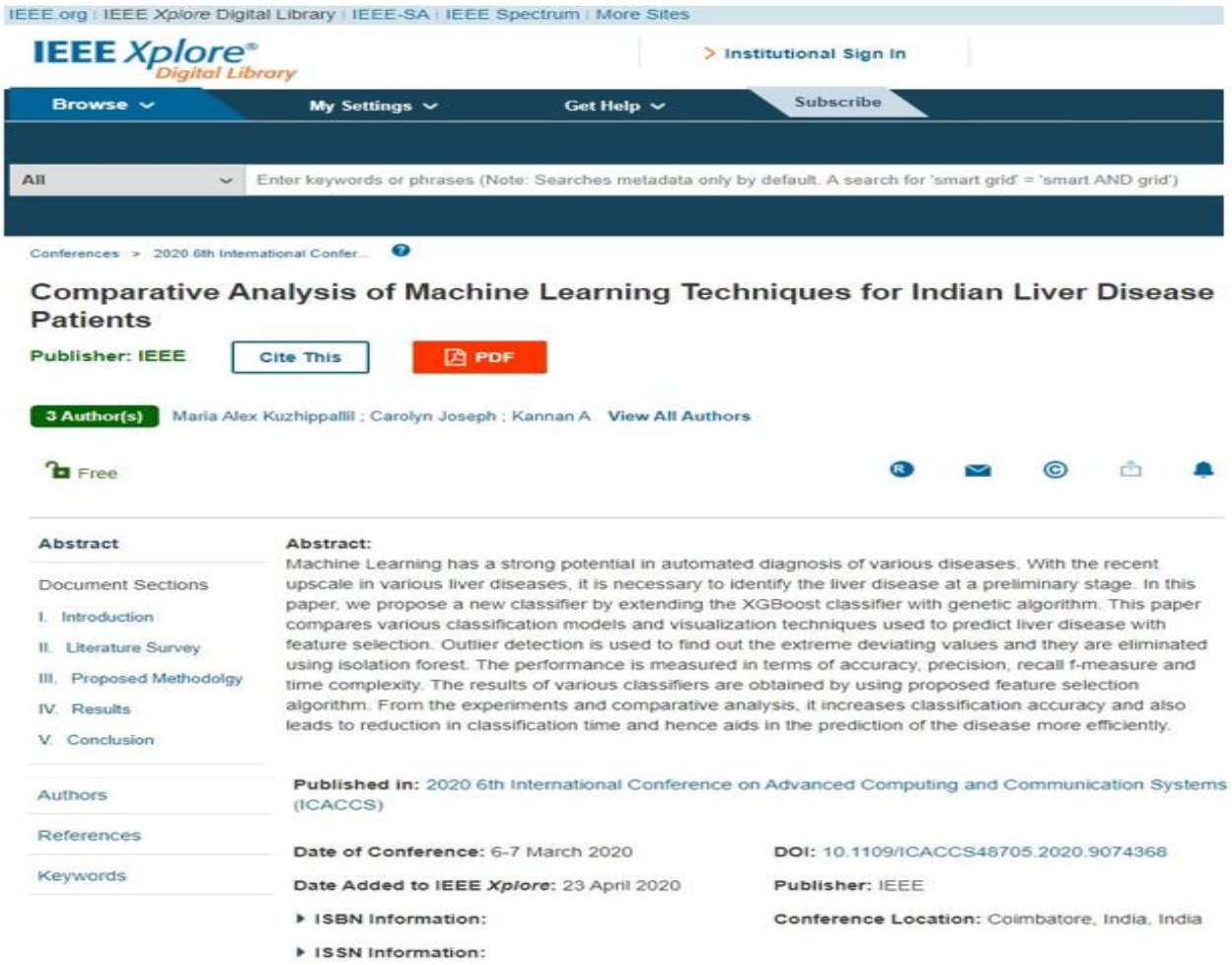

Fig 104 – Research Paper Published in IEEE Xplore Digital Library

7. RESULTS AND DISCUSSION

Comparitive analysis of performance before and after the outlier detection using Isolation forest and feature selection using Genetic Algorithm

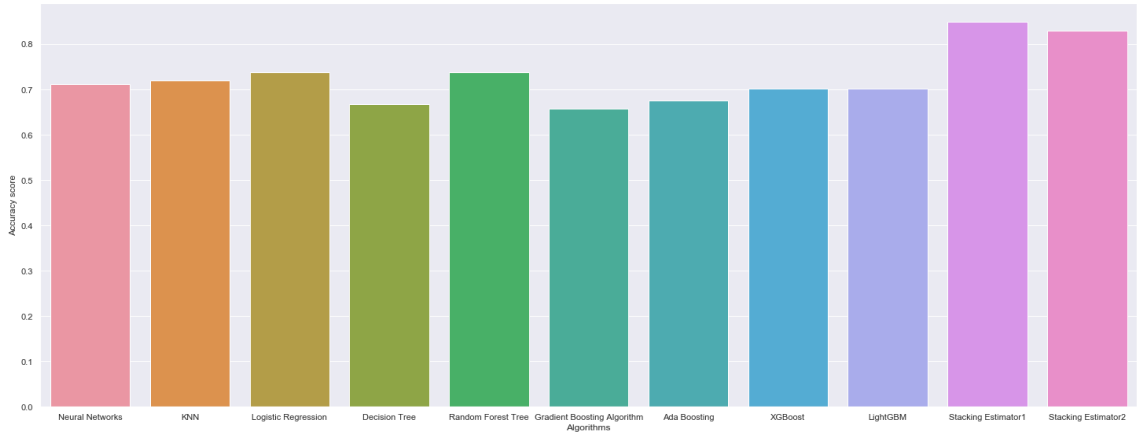

Fig 105 - Accuracy before outlier elimination and feature selection

Before outlier elimination and feature selection, we can observe that the performance of majority of algorithms is fair and is between the range 70% to 80%. Among all the algorithms stacking estimator provides better results.

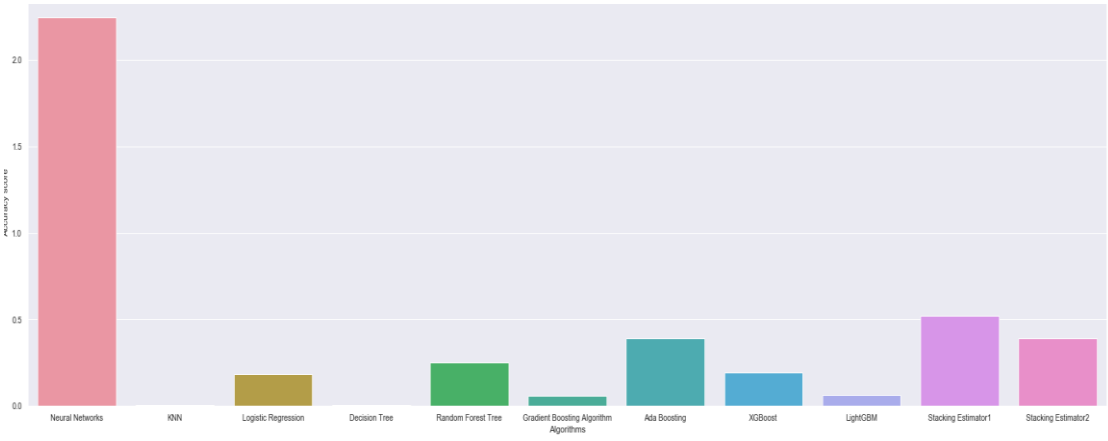

Fig 106 - Time before outlier elimination and feature selection

Time taken to execute the algorithms and prediction is extremely high for multilayer perceptron algorithm which is 2.243 seconds. Other algorithms such stacking estimator, adaboost, random forest tree, logistic algorithm and XGBoost also consume time. Considerable less time is taken by remaining algorithms. The range of time for algorithms is between 0 – 2.243seconds.

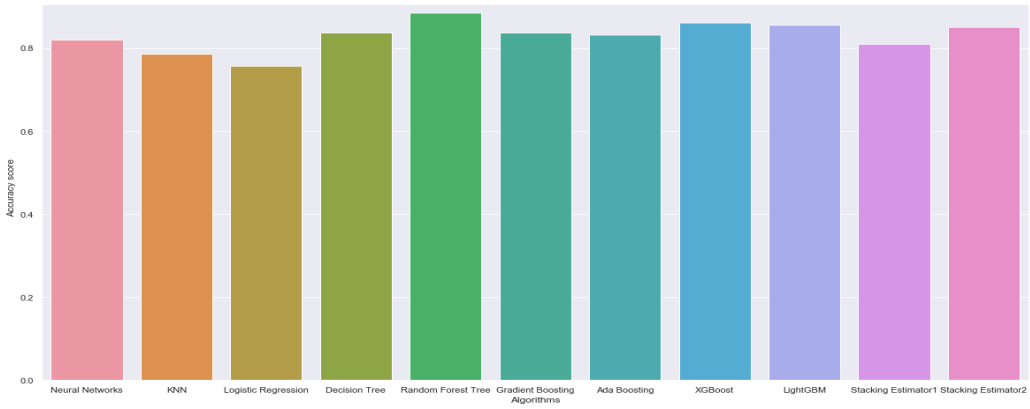

Fig 107 - Accuracy after outlier elimination and feature selection

The accuracy from 70% to 80% has increased for most of the algorithms and the current range is 79% to 86%. The performance of all algorithms has increased irrespective of nature of algorithm. All algorithms are able to classify liver patients correctly. Among all the best performance is obtained from XGBoost, LightGBM, Random forest tree and stacking estimator which inculcates ensemble machine learning concepts.

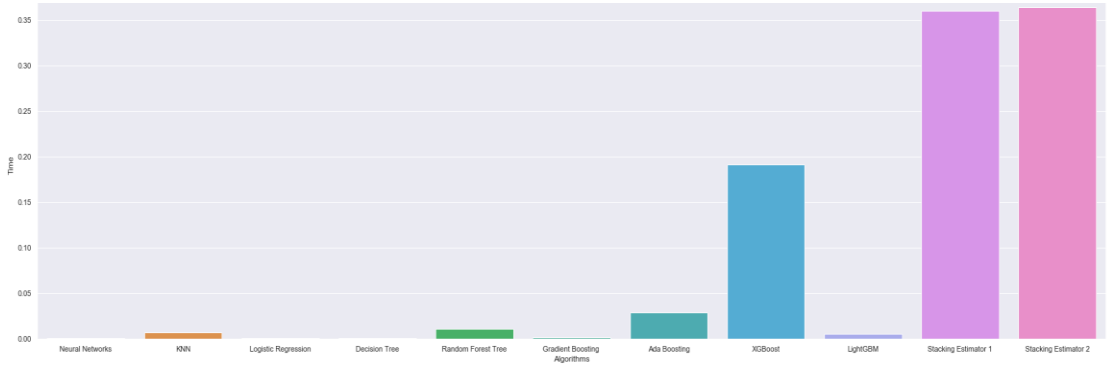

Fig 108 - Time after outlier elimination and feature selection

Time has reduced drastically, previously the range from 0 - 2.243 has reduced to 0 – 0.35. The maximum time is taken by Stacking estimator that is 0.35.

The performance of various machine learning algorithms have been evaluated for liver data. The data is obtained from UCI machine learning repository and over sampling is performed using a variant of SMOTE using Genetic algorithm.

Table 5
Before feature selection and outlier elimination

Algorithm	Accuracy	Precision	Recall	F-measure	Time Complexity
Multilayer Perceptron	0.71	0.50	0.71	0.59	2.243
KNN	0.72	0.69	0.72	0.69	0.0079
Logistic Regression	0.74	0.71	0.74	0.70	0.184
Decision Tree	0.67	0.68	0.67	0.67	0.0079
Random Forest Tree	0.74	0.72	0.74	0.72	0.2503
Gradient Boosting	0.66	0.67	0.66	0.66	0.0568
AdaBoost	0.68	0.64	0.68	0.65	0.390
XGBoost	0.70	0.68	0.70	0.69	0.191
Light GBM	0.70	0.70	0.70	0.70	0.063
Stacking Estimator	0.83	0.83	0.83	0.83	0.388

Table 6
After feature selection and outlier elimination

Algorithm	Accuracy	Precision	Recall	F-measure	Time Complexity
Multilayer Perceptron	0.82	0.81	0.82	0.80	0.00099
KNN	0.79	0.77	0.79	0.74	0.0069
Logistic Regression	0.76	0.72	0.76	0.72	0.00099
Decision Tree	0.84	0.84	0.84	0.84	0.00099
Random Forest Tree	0.88	0.88	0.88	0.88	0.0109
Gradient Boosting	0.84	0.84	0.84	0.84	0.0019
AdaBoost	0.83	0.83	0.83	0.83	0.0289
XGBoost	0.86	0.86	0.86	0.86	0.191
Light GBM	0.86	0.85	0.86	0.85	0.0059
Stacking Estimator	0.85	0.85	0.85	0.85	0.364

From the above two tables it's clear that the performance in terms of accuracy, precision, recall and f-measure has increased after feature selection using genetic algorithm and eliminating anomaly values using isolation forest algorithm. The best performance in terms

of these performance measures is produced by Boosting algorithms like XGBoost and Light GBM.

Stacking estimator which combines multiple algorithm also provides quite good performance.

There is a tremendous decrease of time in all algorithms increasing the efficiency and suggesting the importance of isolation forest and feature selection genetic algorithm.

Before feature selection and using isolation forest, the F-measure value is low because of data skewness which leads to tradeoff between precision and recall. In order to improve data skewness is reduced using interquartile range values and since the data used is highly imbalanced Genetic SMOTE algorithm is used to over sample the data.

Apart from these techniques such data cleaning using label encoding, imputation of missing values, duplicate value elimination has aided in providing better results.

8. SUMMARY

With the rising advancements of data mining and analytics makes it necessary to extend the applications to healthcare dataset. The current researches used in the healthcare data classification have low accuracy ranging in 70s. Hybrid models have proved to improve the classification models for text classification datasets. With the rise in patients suffering from various diseased, it is important to use accurate classification algorithms. Different classification algorithms were used to predict the presence or absence of liver disease. Performance metrics such as accuracy, precision, recall, f-measure and time complexity are effectively utilized to analyse the performance of various classification algorithms. And the proposed model of Genetic algorithm combined with XGBoost is used to fetch the best attributes required for prediction of liver disease.

In this project, liver disease prediction has been studied and analyzed. The data is cleaned by performing various techniques such as imputation of missing values with median, label encoding to convert categorical into numerical data for easy analysis, duplicate value elimination and outliers are eliminated using Isolation forest in order to improve the performance. Genetic algorithm is used to fetch the best attributes required for prediction of liver disease. Different classification algorithms are used to predict the presence or absence of liver disease. Performance metrics such as accuracy, precision, recall, f-measure and time complexity is effectively utilized to analyses the performance of various classification algorithms.

The project also provides a platform for users to predict the presence or absence of liver disease using machine learning techniques helps identify the disease at a preliminary stage. As a future work, other diseases can be incorporated. Data set for other diseases such as diabetes, heart disease etc are available in UCI machine learning repository

A blog was also created which would predict the chances of having a liver disease, where the patients could enter the values of the various entities from the blood test and then get the result. The blog would also allow users on the latest finding in the medical field regarding the liver disease and also preventive measures.

References

[1] Saravanan, R. & Sujatha, Pothula. (2018). A State of Art Techniques on Machine Learning Algorithms: A Perspective of Supervised Learning Approaches in Data Classification. 945-949. 10.1109/ICCONS.2018.8663155.

[2] Kumar, P. Nagendra Siva and Ramjeevan Singh Thakur, "Diagnosis of Liver Disorder Using Fuzzy Adaptive and Neighbor Weighted K-NN Method for LFT Imbalanced Data." 2019 International Conference on Smart Structures and Systems (ICSSS) (2019): 1-5.

[3] S. Bahramirad, A. Mustapha and M. Eshraghi, "Classification of liver disease diagnosis: A comparative study," 2013 Second International Conference on Informatics & Applications (ICIA), Lodz, 2013, pp. 42-46, doi: 10.1109/ICoIA.2013.6650227.

[4] Dua, D. and Graff, C. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science (2019).

[5] Arshad, Insha & Dutta, Chiranjit & Choudhury, Tanupriya & Thakral, Abha. (2018). Liver Disease Detection Due to Excessive Alcoholism Using Data Mining Techniques. 163-168. 10.1109/ICACCE.2018.8441721.

[6] Srikanth, Panigrahi and Dharmaiah Deverapalli. "A Critical Study of Classification Algorithms Using Diabetes Diagnosis." 2016 IEEE 6th International Conference on Advanced Computing (IACC) (2016): 245-249.

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

[8] Baiju, B. V. and D. John Aravindhar. "Disease Influence Measure Based Diabetic Prediction with Medical Data Set Using Data Mining." 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT) (2019): 1-6.

[9] Noria, Bidi & Elberrichi, Zakaria. (2016). Feature selection for text classification using genetic algorithms. 806-810. 10.1109/ICMIC.2016.7804223.

[10] A. Ahlawat and B. Suri, "Improving classification in data mining using hybrid algorithm," 2016 1st India International Conference on Information Processing (IICIP), Delhi, 2016, pp. 1-4, doi: 10.1109/IICIP.2016.7975380.

[11] Shi, Haobin and Meng Xu. "A Data Classification Method Using Genetic Algorithm and K-Means Algorithm with Optimizing Initial Cluster Center." 2018 IEEE International Conference on Computer and Communication Engineering Technology (CCET) (2018): 224-228.

[12] Ramkumar, N. & Prakash, S. & Kumar, S. & Sangeetha, K. (2017). Prediction of liver cancer using Conditional probability Bayes theorem. 1-5. 10.1109/ICCCI.2017.8117752.

[13] Yan, Ke & You, Xiaoming & Ji, Xiaobo & Yin, Guangqiang & Yang, Fan. (2016). A Hybrid Outlier Detection Method for Health Care Big Data. 157-162. 10.1109/BDCloud-SocialCom-SustainCom.2016.34.

APPENDIX

Personal Contribution (Maria Alex Kuzhippallil – 16BCE2190)

- Developed the entire data analysis of Liver Disease in Jupyter Notebook.
- For research paper –

  -Performed part of literature survey

  -Proposed methodology

  -Results

  -Conclusion

  -Formatting in IEEE format
- Skype Presentation for the research paper in Set Conference.
- Developed the entire web application in Django

  -Disease prediction

  -Creating the blog.
- For final project report –

  - Introduction (Objectives)

  - Project Description and Goals

  - Technical Specifications

  - Design Approach and Details

  - Schedule,Tasks and Milestones

  - Project Demonstration

  - Cost Analysis/ Result & Discussion.
- Video Recording –

  -Explaining aim

  -Objective

  -Motivation

  -Demonstration

  -Future scope

  -Conclusion.

Link for Published Paper – https://ieeexplore.ieee.org/document/9074368

Link for Video Description – https://drive.google.com/file/d/1edX8biIF5aKfmkxaYcwr--mSPlGVydoO/view?usp=sharing