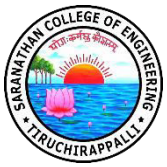


SARANATHAN COLLEGE OF ENGINEERING

VENKATESHWARA NAGAR, PANJAPPUR, TRICHY-12

NALAIYA THIRAN

PROJECT TITLE : STATISTICAL MACHINE LEARNING APPROACHES TO LIVER DISEASE PREDICTION

DOMAIN : APPLIED DATA SCIENCE

TEAM ID : PNT2022TMID32847

TEAM LEADER :

KALAIARASU T - 813819205025@SMARTINTERNZ.COM

TEAM MEMBERS :

ARUN KUMAR S - 813819205009@SMARTINTERNZ.COM

DHARUN UDHAYA K - 813819205014@SMARTINTERNZ.COM

DHIVAGAR P - 813819205015@SMARTINTERNZ.COM

GITHUB : PROJECT REPOSITORY

TABLE OF CONTENTS

TABLE OF CONTENTS

1. INTRODUCTION

With a growing trend of sedentary and lack of physical activities, diseases related to liver have become a common encounter nowadays. In rural areas the intensity is still manageable, but in urban areas, and especially metropolitan areas the liver disease is a very common sighting nowadays. Liver diseases cause millions of deaths every year. Viral hepatitis alone causes 1.34 million deaths every year. Problems with liver patients are not easily discovered in an early stage as it will be functioning normally even when it is partially damaged. An early diagnosis of liver problems will increase patients' survival rate. Liver failures are at high rate of risk among Indians. It is expected that by 2025 India may become the World Capital for Liver Diseases. The widespread occurrence of liver infection in India is contributed due to desk bound lifestyle, increased alcohol consumption and smoking. There are about 100 types of liver infections. With such alarming figures, it is necessary to have a concern towards tackling these diseases. After-all, we cannot expect a developed and prosperous nation, with unhealthy youths. In this project we have taken UCI ILPD Dataset which contains 10 variables that are age, gender, total Bilirubin, direct Bilirubin, total proteins, albumin, A/G ratio, SGPT, SGOT and Alkphos and contains 415 as liver disease patients and 167 as non-liver disease patients. As we got through the next parts of this paper, we will explain what process as taken place for the selection of best model and building necessary system for the prediction of liver disease.

1.1 PROJECT OVERVIEW

In India, delayed diagnosis of diseases is a fundamental problem due to a shortage of medical professionals. A typical scenario, prevalent mostly in rural and somewhat in urban areas is:

- A patient going to a doctor with certain symptoms.
- The doctor recommending certain tests like blood test, urine test etc. depending on symptoms.
- The patient taking the a fore mentioned tests in an analysis lab.
- The patient taking the reports back to the reports back to the hospital, where they are
- examined and the disease is identified.

The aim of this project is to somewhat reduce the time delay caused due to the unnecessary back and forth shuttling between the hospital and the pathology lab. Historically, work has been done in identifying the onset of diseases like heart disease, Parkinson's from various

1.2 PURPOSE

Early prediction of liver disease is very important to save human life and take proper steps to control the disease. Liver enlargement is usually an indicator of liver disease, although there are usually no symptoms associated with a slightly enlarged liver. Symptoms of a grossly enlarged liver include abdominal discomfort or "feeling full."

Over time, liver disease can cause cirrhosis (scaring). As more scar tissue replaces healthy liver tissue, the liver can no longer function properly. Left untreated, liver disease can lead to liver failure and liver

cancer. Patients with Liver disease have been continuously increasing because of excessive consumption of alcohol, inhale of harmful gases, intake of contaminated food, pickles, and drugs. This machine learning was used to evaluate prediction algorithms in an effort to reduce burden on doctors.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

S.NO	TITLE	AUTHORS - YEAR OF PUBLICTATIONS	PROPOSED WORK
1	Statistical Machine Learning Approaches to Liver Disease Prediction	Fahad Mostafa, Easin Hasan, Morgan Williamson and Hafiz Khan. 01/12/2021	This paper aims to extract significant predictors for liver disease from the medical analysis of 615 humans using ML algorithms. The study compared binary classifier machine learning algorithms (i.e., artificial neural network, random forest (RF), and support vector machine), which were utilized on a published liver disease data set to classify individuals with liver diseases, which will allow health professionals to make a better diagnosis.
2	A Comparative Study On Liver Disease Prediction Using Supervised Machine Learning algorithms	A.K.M Sazzadur Rahman, F. M. Javed Mehedi Shamrat, Zarrin Tasnim, Joy Roy, Syed Akhter Hossain. 11/11/2019	This paper evaluates the performance of different Machine Learning algorithms in order to reduce the high cost of chronic liver disease diagnosis by prediction. Six machine learning techniques have been applied including Logistic Regression, K Nearest Neighbors, Decision Tree, Support Vector Machine, Naïve Bayes, and Random Forest. The performance was evaluated on different measurement techniques such as accuracy, precision, recall, f-1 score, and specificity and the result were that LR achieved the and the highest accuracy

3	Liver Disease Prediction System using Machine Learning Techniques	Rakshith D B, Mrigank Srivastava, Ashwani Kumar, Gururaj S P. 06/06/2021	In this paper risk of liver disease for a person is predicted based on the blood test report results of the user. With the dataset used for this project, 100 % accuracy is obtained for SVM model. The programming language which was used is python and machine learning Sklearn was used to build the model using classification algorithm like KNN, SVM, Naive Bayes and ANN.

2.2 REFERENCE

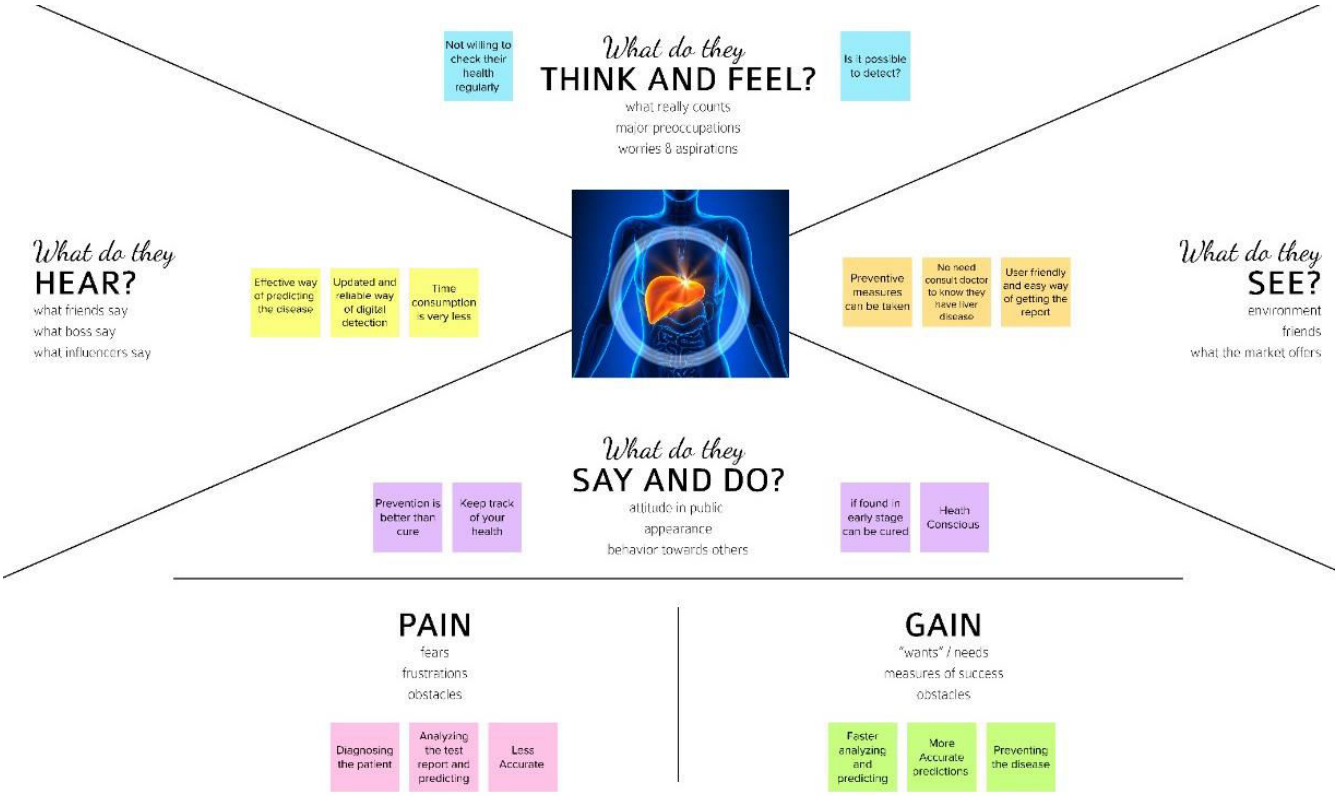
- Fahad Mostafa, Easin Hasan, Morgan Williamson and Hafiz Khan. 01/12/2021
- A.K.M Sazzadur Rahman, F. M. Javed Mehedi Shamrat, Zarrin Tasnim, Joy Roy, Syed Akhter Hossain. 11/11/2019
- Rakshith D B, Mrigank Srivastava, Ashwani Kumar, Gururaj S P.06/06/2021

2.3 PROBLEM STATEMENT DEFINITION

After researching and getting to know about the various approaches that have been devised for the liver disease prediction using machine learning we have decided to propose our problem statement as This Project examines data from liver patients concentrating on relationships between a key list of liver enzymes, proteins, age and gender using them to try and predict the likeliness of liver disease. Here we are building a model by applying various machine learning algorithms find the best accurate model. And integrate to flask-based web application. User can predict the disease by entering parameters in the web application

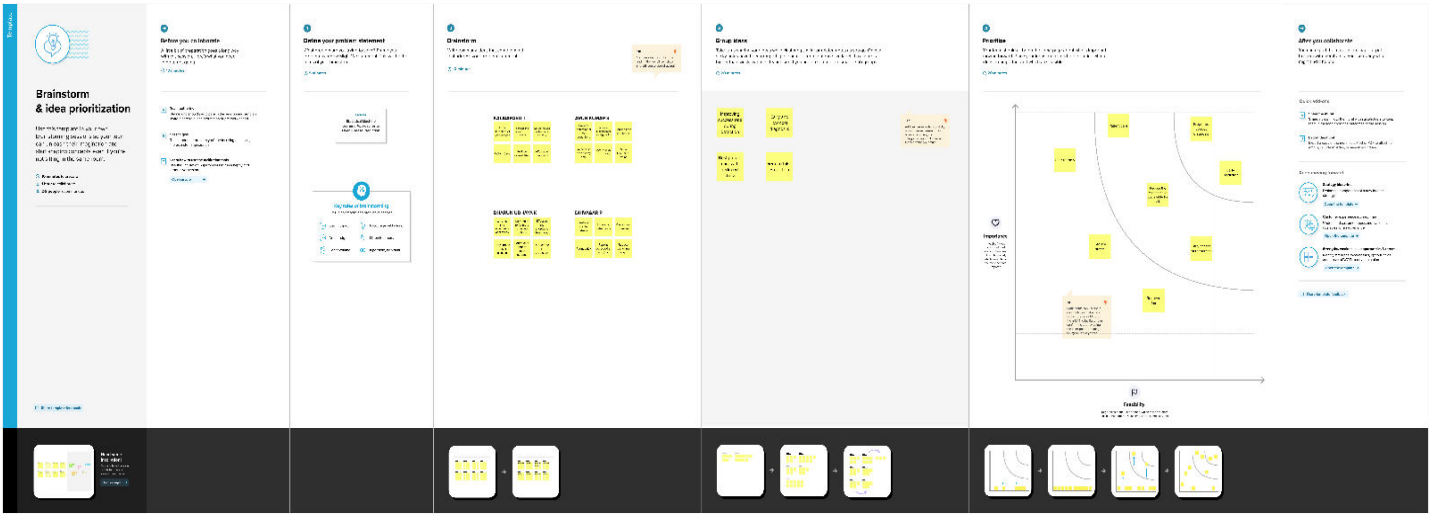
3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION & BRAINSTORMING

After researching and getting to know about the various approaches that have been devised for the liver disease prediction using machine learning we have decided to propose our problem statement as This Project examines data from liver patients concentrating on relationships between a key list of liver enzymes, proteins, age, and gender using them to try and predict the likeliness of liver disease. Here we are building a model by applying various machine learning algorithms find the best accurate model. And integrate to flask-based web application. User can predict the disease by entering parameters in the web application.



3.3 PROPOSED SOLUTION

S.NO	PARAMETER	DESCRIPTION
1	Problem Statement (Problem to be solved)	With a growing trend of sedentary and lack of physical activities, diseases related to liver have become a common encounter nowadays. In rural areas the intensity is still manageable, but in urban areas, and especially metropolitan areas the liver disease is a very common sighting nowadays. Liver diseases cause millions of deaths every year. Viral hepatitis alone causes 1.34 million deaths every year. Problems with liver patients are not easily discovered in an early stage as it will be functioning normally even when it is partially damaged.
2	Idea / Solution description	Healthcare system can benefit from various Machine Learning (ML) models to predict diseases in early stage. The aim of this study is to predict liver disease using different ML models applied on Indian Liver Patient Dataset (ILPD). The models used on this work are Support Vector Machine (SVM), K-Nearest Neighbour (KNN), Random Forest (RF), Artificial Neural Network (ANN) and various versions of Ensemble Learning (EL) to find the solution for this
3	Novelty / Uniqueness	In Human beings, Liver is the most primary part of the body that performs many functions including the production of Bile, excretion of bile and bilirubin, metabolism of proteins and carbohydrates, activation of Enzymes, Storing glycogen, vitamins, and minerals, plasma proteins synthesis and clotting factors. The liver easily gets affected due to intake of alcohol, pain killer tablets, food habits, and includes plenty of wired practices. Currently, the liver related diseases are identified ...
4	Social Impact / Customer Satisfaction	Morbidity and mortality of liver disease are increasing in frequency because alcoholism, adverse reactions from drug use and abuse, and viral hepatitis are more prevalent. As the nature of these factors suggests, the disadvantaged are particularly at risk.
5	Business Model (Revenue Model)	Optional
6	Scalability of the Solution	Early diagnosis and treating the patients are significant to reduce the risk. Healthcare system can benefit from various Machine Learning (ML) models to predict diseases in early stage. The aim of this study is to predict liver disease using different ML models applied on Indian Liver Patient Dataset (ILPD).

3.4 PROBLEM SOLUTION FIT

Define CS, fit into CC

1. CUSTOMER SEGMENT(S)

CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

Liver diseases avert the normal function of the liver. Mainly due to the large amount of alcohol consumption liver disease arises. Early prediction of liver disease using classification algorithms is an efficacious task that can help the doctors to diagnose the disease within a short duration of time.

6. CUSTOMER CONSTRAINTS

CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e., spending power, budget, no cash, network connection, available devices.

- Avoid risky behavior
- Keep your food safe
- Eat alternative medicine

5. AVAILABLE SOLUTIONS

AS

Which solutions are available to the customers when they face the problem
or need to get the job done? What have they tried in the past?
What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

Drink alcohol sparingly, if at all. Avoid red meat, trans fats, processed carbohydrates and foods with high-fructose corn syrup. Exercise 30 to 60 minutes around three to four times a week at a moderate intensity.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS

J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

- Eat large amounts of carbohydrate foods. ...
- Eat a moderate intake of fat, as prescribed by the provider. ...
- Have about 1.2 to 1.5 grams of protein per kilogram of body weight. ...
- Take vitamin supplements, especially B-complex vitamins.
- Many people with liver disease are deficient in vitamin D.

9. PROBLEM ROOT CAUSE

RC

What is the real reason that this problem exists? What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

Mainly due to the large amount of alcohol consumption liver disease arises. Early prediction of liver disease using classification algorithms is an efficacious task that can help the doctors to diagnose the disease within a short duration of time. Discovering the existence of liver disease at an early stage is a complex task for the doctors. The main objective of this project is to analyze the parameters of various classification algorithms and compare their predictive accuracies out the best classifier for determining the liver disease.

7. BEHAVIOUR

BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

- Reduce Your Drinking. According to the National Institute on Alcohol Abuse and Alcoholism, the biggest cause of liver damage – and death from liver disease – is chronic alcohol consumption
- Eat The Right Foods
- Cut Out Other Toxins
- Get Active
- Be Mindful Of Medications

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS

TR

What triggers customers to act?
i.e. seeing their neighbor installing solar panels, reading about a more efficient solution in the news.

- Heavy alcohol use.
- Type 2 diabetes.
- Tattoos or body piercings.
- Injecting drugs using shared needles.
- Blood transfusion before 1992.
- Exposure to other people's blood and body fluids.
- Unprotected sex.

4. EMOTIONS: BEFORE / AFTER

EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

- Dealing with angry customers.
- No crisis management or escalation protocol
- Not meeting customer expectations.
- Poor understanding Communication issue by client
- Task or event issue by manager or higher authority
- Time consume project issue
- Work Punctuality problem rises

10. YOUR SOLUTION

SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behavior

Yeah, but customer have rights to do with the public places we don't have rights or order them But in company if they working in organization means they must be followed strictly and uniformly for their position

8. CHANNELS of BEHAVIOUR

CH

8.1 ONLINE

What kind of actions do customers take online? Extract online channels from #7

- Customer if order something product will be delay for particular website to post unwanted comment
- Irresponsible behavior to cut the webpage access
- Poor network connectivity to down the domain network

8.2 OFFLINE

- What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development If you are shaking your head in disdain at the sheer naivety of the person who would give out his hard-earned money, you will do at the time compliant to police station. Uncomfortable Friends. The friends of the cheating partner usually know about it before you do.
- Inconsistent Expenses. ...
- False Accusations of Cheating.
- Unknown person issue to make call with you means just record the call and give it to vigilance department

Identify strong TR & EM

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

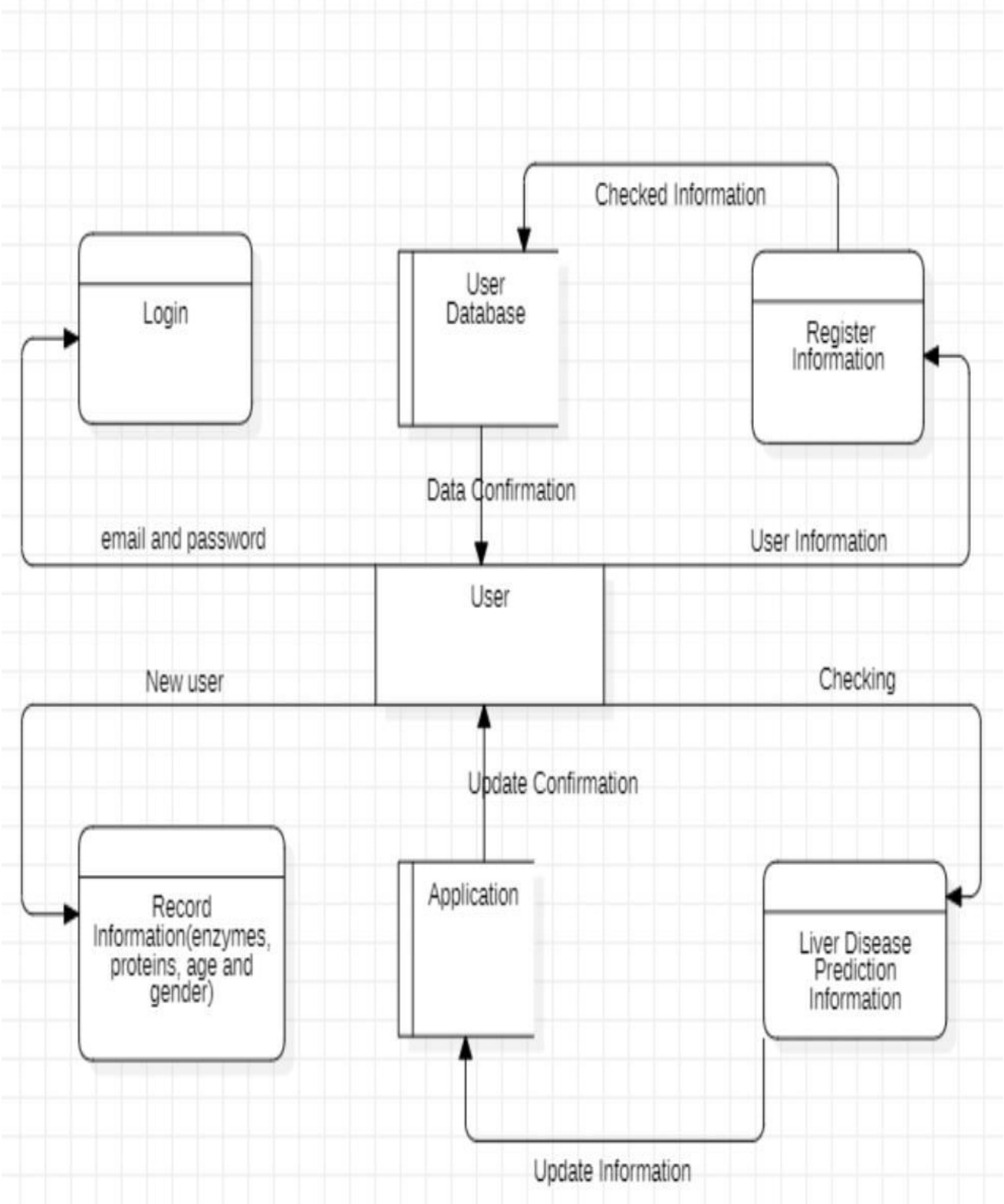
FR.NO	FUNCTIONAL REQUIREMENT(EPIC)	SUB REQUIREMENT(STORY/SUB-TASK)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIn
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Check-up details	Enter the body condition Provide the Solution
FR-4	Result of condition	Verify the Possibilities of life strength Result Confirmed

4.2 NON-FUNCTIONAL REQUIREMENT

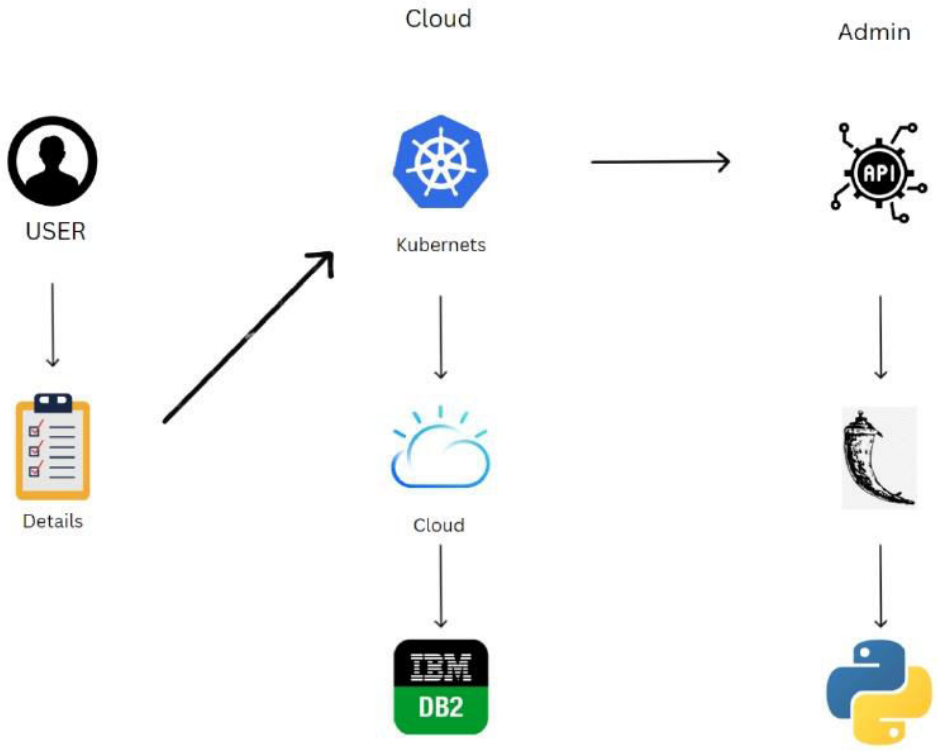
FR-NO	NON-FUNCTIONAL REQUIREMENT	DESCRIPTION
NFR-1	Usability	Requirement Negative to use
NFR-2	Security	As much of people self-protect with good habit
NFR-3	Reliability	Even when liver stiffness measurement is feasible, high BMI values negatively affect the diagnostic reliability. Improved performance of transient elastography could be obtained using specifically designed probes.
NFR-4	Performance	Data pre-processing Feature Extraction Prediction through body condition
NFR-5	Availability	Liver disease may result from Viral infections Hepatitis A, hepatitis B and hepatitis C are diseases caused by a viral infection. Problems with your immune system: When your immune system mistakenly attacks your liver, it can cause autoimmune liver diseases
NFR-6	Scalability	The Meld score ranges from 6 to 4 0, and is a measure of how severe a patient's liver disease is. MELD can fluctuate based on your current condition, with variations from a few points as lab values vary to a larger increase if you have an infection or an acute decompensation (worsening of your liver disease).

5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS



5.2 Solution & Technical Architecture



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the Liver Disease Prediction application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
	Register	USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
	Register	USN-4	As a user, I can register for the application through Gmail	I can register & access the dashboard with Gmail Login	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can access my account in Application	High	Sprint-1
Customer (Web user)	Login	USN-6	As a user, I can log into the website by entering username & password	I can access my account/ dashboard	High	Sprint-1
Customer Care Executive	Login	USN-7	AS a Customer Care Executive log into the website.	I can access user account/ dashboard	High	Sprint-2

6. PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING ESTIMATION

SPRINT	FUNCTIONAL REQUIREMENT (EPIC)	USER STORY NUMBER	USER STORY/TASK	STRORY POINTS	PRIORITY	TEAM MEMBERS
Sprint-1	Data Input	USN-1	As a user, I can enter the details that is asked to predict that I have Liver Disease.	2	Medium	Arun Kumar S
Sprint-2	Analyze	USN-2	I can analyse the dataset	1	High	Dharun Udhaya K
Sprint-3	Develop and train	USN-3	I can develop and train the model to predict the liver disease	2	High	Kalaiarasu T
Sprint-4	Application	USN-4	Shows the final Prediction	2	Medium	Dhivagar P

6.2 SPRINT DELIVERY SCHEDULE

SPRINT	TOTAL STORY POINTS	DURATION	SPRINT START DATE	SPRINT END DATE (PLANNED)	STORY POINTS COMPLETED (AS ON PLANNED END DATE)	SPRINT RELEASE DATE (ACTUAL)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 REPORTS FROM JIRA

7. CODING & SOLUTIONING

7.1 INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!--link rel="stylesheet" href="index.css"-->
    <title>Liver Disease Predictor</title>
    <style>
        *{
            background-color: #0e1538;
```

```
    color: whitesmoke;
    font-variant: small-caps;
    font-size: large;
    font-family: Georgia, 'Times New Roman', Times, serif;
    margin: 0;
    padding: 0;
}
body{
    min-width: 200px;
    min-height: 600px;
    align-items: center;
    justify-content: center;
    font-weight: bold;
}
h2{
    display: block;
    font-size: 1.5cm;
    margin: 0.7em;
    text-align: center;
}
table{
    margin-left: auto;
    margin-right: auto;
    vertical-align: middle;
}
td{
    margin: 1em;
    padding: 0.5em;
    text-align: justify;
}
input{
    border-radius: 1em;
    border: solid;
    border-color: whitesmoke;
    border-width: 2px;
    color: #00ccff;
    padding: 0.2em 0.7em;
```

```
    -moz-appearance: textfield; /*For Firefox*/
}
/*For Chrome and safari*/
input::-webkit-outer-spin-button,
input::-webkit-inner-spin-button {
-webkit-appearance: none;
margin: 0;
}
input::placeholder{
    color: #00ccff;
}
#btndiv{
    padding: 2px;
    width: max-content;
    border-radius: 1em;
    background: linear-gradient(45deg,#00ccff,#0e1538,#d400d4);
    display: inline;
}
#submit,#reset{
    /*border-radius: 1em;
    border-style: solid;
    border-color: whitesmoke;*/
    border: none;
    background: #0e1538;
    color: whitesmoke;
    padding: 0.2em 0.7em;
    border-radius: 1em;
}
#btn{
    display: flex;
}
#submit:hover{
    cursor: pointer;
}
#btndiv:hover{
    background: linear-gradient(45deg,#1aff22,#0e1538,#ff075b);
}
```

```
    #positive{
        background: linear-gradient(10deg,#1aff22,#0e1538,#0e1538,#0e1538,#1aff22);
        padding: 5px;
        margin: 1em;
        border-radius: 3rem;
    }
    #negative{
        background: linear-gradient(10deg,#ff075b,#0e1538,#0e1538,#0e1538,#ff075b);
        padding: 5px;
        margin: 1em;
        border-radius: 3rem;
    }
    #positive:hover{
        background: linear-gradient(170deg,#1aff22,#0e1538,#0e1538,#0e1538,#1aff22);
    }
    #negative:hover{
        background: linear-gradient(170deg,#ff075b,#0e1538,#0e1538,#0e1538,#ff075b);
    }
    h4{
        display: block;
        font-size: 0.8cm;
        margin: auto;
        padding: 1em;
        border-radius: 3rem;
        text-align: center;
        background: #0e1538;
    }
</style>
<script>
    function Predict()
    {
        document.getElementById('positive').hidden = true;
        document.getElementById('negative').hidden = true;
        var age = document.forms["ipdata"]["age"].value;
        var gender = document.forms["ipdata"]["gender"].value;
        var tb = document.forms["ipdata"]["tb"].value;
        var db = document.forms["ipdata"]["db"].value;
```

```
        var ap = document.forms["ipdata"]["ap"].value;
        var aa = document.forms["ipdata"]["aa"].value;
        var asa = document.forms["ipdata"]["asa"].value;
        var a = document.forms["ipdata"]["a"].value;
        var tp = document.forms["ipdata"]["tp"].value;
        var agr = document.forms["ipdata"]["agr"].value;
        document.getElementById('reset').click();

        if (gender == 'Male')
           gender = 0
        else
           gender = 1

        var url = 'http://127.0.0.1:5000/predict?age=' + age + '&gender=' + gender + '&tb=' + tb + '&db=' +
db + '&ap=' + ap + '&aa=' + aa + '&asa=' + asa + '&a=' + a + '&tp=' + tp + '&agr=' + agr
        fetch(url)
        .then( response => response.json() )
        .then( data => {
           console.log(data.result);
           if(parseInt(data.result)>50)
           {
              document.getElementById('neg_data').innerHTML = "Probability of Liver Failure is "+
data.result +"%<br/>\"There is a Possiblity that you are having Liver Disease.\"";
              document.getElementById('negative').hidden = false;
           }
           else
           {
              document.getElementById('pos_data').innerHTML = "Probability of Liver Failure is "+
data.result +"%<br/>\"There is a Very Less Possiblity that you are have Liver Disease. Stay Healthy\"";
              document.getElementById('positive').hidden = false;
           }

        })
        .catch( error => console.log(error) )
     }
  </script>
</head>
```

```
<body>
   <h2 id="h">Liver Disease Prediction</h2>
   <div id="positive" hidden><h4 id="pos_data">Positive</h4></div>
   <div id="negative" hidden><h4 id="neg_data">Negative</h4></div>
   <table>
      <form name="ipdata" onsubmit="event.preventDefault(); Predict();" on action="/" method="get">
      <tr>
         <td>Age</td>
         <td><input name="age" type="number" min="0" step="1" placeholder="Eg: 30" required></td>
      </tr>
      <tr>
         <td>Gender</td>
         <td><input name="gender" type="radio" value="Male" required> Male  <input
id="gender" name="gender" type="radio" value="Female"> Female</td>
      </tr>
      <tr>
         <td>Total Bilirubin</td>
         <td><input name="tb" type="number" min="0" step="0.01" placeholder="0.22 - 1.0 mg/dl"
required></td>
      </tr>
      <tr>
         <td>Direct Bilirubin</td>
         <td><input name="db" type="number" min="0" step="0.01" placeholder="0.0 - 0.2 mg/dl"
required></td>
      </tr>
      <tr>
         <td>Alkaline Phosphotase</td>
         <td><input name="ap" type="number" min="0" step="0.01" placeholder="110 - 310 U/L"
required></td>
      </tr>
      <tr>
         <td>Alamine Aminotransferase (SGPT)</td>
         <td><input name="aa" type="number" min="0" step="0.01" placeholder="5 - 45 U/L"
required></td>
      </tr>
      <tr>
         <td>Aspartate Aminotransferase (SGOT)</td>
```

```
        <td><input name="asa" type="number" min="0" step="0.01" placeholder="5 - 40 U/L"
required></td>
      </tr>
      <tr>
        <td>Albumin</td>
        <td><input name="a" type="number" min="0" step="0.01" placeholder="3.5 - 5 gm/dl"
required></td>
      </tr>
      <tr>
        <td>Total Proteins</td>
        <td><input name="tp" type="number" min="0" step="0.01" placeholder="7.2-8.0 gm/100ml"
required></td>
      </tr>
      <tr>
        <td>A/G Ratio</td>
        <td><input name="agr" type="number" min="0" step="0.01" placeholder="1.7-2.2" required></td>
      </tr>
      <tr>
        <td></td>
        <td id="btn"><div id="btndiv"><input id="submit" type="submit"
value="Predict"></div> <div id="btndiv"><input id="reset" type="reset"
value="Clear"></div></td>
      </tr>

      </form>
   </table>
</body>
</html>
```

7.2 MODEL_ANALYZE.IPYNB

```
# **Type of Machine Learning Problem**
It is a binary classification problem, where given the above set of features, we need to predict if a given patient has liver disease or not
## **Evaluation Metric (KPI)**
Since this is binary classification problem, we use the following metrics:
*   **Confusion matrix** - For getting a better clarity of the no of correct/incorrect predictions by the model
```

* **ROC-AUC** - It considers the rank of the output probabilities and intuitively measures the likelihood that model can distinguish between a positive point and a negative point. (Note: ROC-AUC is typically used for binary classification only). We will use AUC to select the best model.

```
# for numerical computing
import numpy as np

# for dataframes
import pandas as pd

# for easier visualization
import seaborn as sns

# for visualization and to display plots
from matplotlib import pyplot as plt
%matplotlib inline

# import color maps
from matplotlib.colors import ListedColormap

# Ignore Warnings
import warnings
warnings.filterwarnings("ignore")

from math import sqrt

# to split train and test set
from sklearn.model_selection import train_test_split

# to perform hyperparameter tuning
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import cross_val_score

# Machine Learning Models
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import roc_curve, auc, roc_auc_score, confusion_matrix

from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from matplotlib.colors import ListedColormap
from sklearn.metrics import accuracy_score

#import xgboost
import os
mingw_path = 'C:\\Program Files\\mingw-w64\\x86_64-7.2.0-posix-seh-rt_v5-rev0\\mingw64\\bin'
os.environ['PATH'] = mingw_path + ';' + os.environ['PATH']
from xgboost import XGBClassifier
from xgboost import plot_importance
from google.colab import files
files.upload()
df = pd.read_csv('Indian Liver Patient Dataset (ILPD).csv')
### Exploratory Data Analysis
df.shape
df.columns
df.head()
df.info()
## Distribution of Numerical Features
# Plot histogram grid
df.hist(figsize=(15,15), xrot=-45, bins=10) ## Display the labels rotated by 45 degress

# Clear the text "residue"
plt.show()
df.describe()
## if score==negative, mark 0 ;else 1
def partition(x):
    if x == 2:
        return 0
    return 1

df['Dataset'] = df['Dataset'].map(partition)
```

```
## Distribution of categorical data
df.describe(include=['object'])
## Bar plots for categorical Features
plt.figure(figsize=(5,5))
sns.countplot(y='Gender', data=df)
df[df['Gender'] == 'Male'][['Dataset', 'Gender']].head()
sns.factorplot (x="Age", y="Gender", hue="Dataset", data=df);
sns.countplot(data=df, x = 'Gender', label='Count')

M, F = df['Gender'].value_counts()
print('Number of patients that are male: ',M)
print('Number of patients that are female: ',F)
## if score==negative, mark 0 ;else 1
def partition(x):
    if x =='Male':
        return 0
    return 1

df['Gender'] = df['Gender'].map(partition)
sns.set_style('whitegrid')   ## Background Grid
sns.FacetGrid(df, hue = 'Dataset', size = 5).map(plt.scatter, 'Total_Bilirubin',
'Direct_Bilirubin').add_legend()
sns.set_style('whitegrid')   ## Background Grid
sns.FacetGrid(df, hue = 'Dataset', size = 5).map(plt.scatter, 'Total_Bilirubin', 'Albumin').add_legend()
sns.set_style('whitegrid')   ## Background Grid
sns.FacetGrid(df, hue = 'Dataset', size = 5).map(plt.scatter, 'Total_Protiens', 'AG_Ratio').add_legend()
## Correlations
df.corr()
plt.figure(figsize=(10,10))
sns.heatmap(df.corr())
mask=np.zeros_like(df.corr())
mask[np.triu_indices_from(mask)] = True
plt.figure(figsize=(10,10))
with sns.axes_style("white"):
    ax = sns.heatmap(df.corr()*100, mask=mask, fmt='.0f', annot=True, lw=1,
cmap=ListedColormap(['green', 'yellow', 'red','blue']))
## Data Cleaning
```

```
df = df.drop_duplicates()
print( df.shape )
## Removing Outliers
sns.boxplot(df.Aspartate_Aminotransferase)
df.Aspartate_Aminotransferase.sort_values(ascending=False).head()
df = df[df.Aspartate_Aminotransferase <=3000 ]
df.shape
sns.boxplot(df.Aspartate_Aminotransferase)
df.Aspartate_Aminotransferase.sort_values(ascending=False).head()
df = df[df.Aspartate_Aminotransferase <=2500 ]
df.shape
df.isnull().values.any()
df=df.dropna(how='any')
df.shape
df.head()
## Machine Learning Models
### Data Preparation
# Create separate object for target variable
y = df.Dataset

# Create separate object for input features
X = df.drop('Dataset', axis=1)
# Split X and y into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1234,
stratify=df.Dataset)

# Print number of observations in X_train, X_test, y_train, and y_test
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
### Data standardization
train_mean = X_train.mean()
train_std = X_train.std()
## Standardize the train data set
X_train = (X_train - train_mean) / train_std
## Check for mean and std dev.
X_train.describe()
## Note: We use train_mean and train_std_dev to standardize test data set
X_test = (X_test - train_mean) / train_std
```

```
## Check for mean and std dev. - not exactly 0 and 1
X_test.describe()
##  Model-1 Logistic Regression
tuned_params = {'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000], 'penalty': ['l1', 'l2']}
LR_model = GridSearchCV(LogisticRegression(), tuned_params, scoring = 'roc_auc', n_jobs=-1)
LR_model.fit(X_train, y_train)
LR_model.best_estimator_
## Predict Train set results
y_train_pred = LR_model.predict(X_train)
## Predict Test set results
y_pred = LR_model.predict(X_test)
# Get just the prediction for the positive class (1)
y_pred_proba = LR_model.predict_proba(X_test)[:,1]
# Display first 10 predictions
y_pred_proba[:10]
i=28  ## Change the value of i to get the details of any point (56, 213, etc.)
print('For test point {}, actual class = {}, precited class = {}, predicted probability = {}'.format(i,
y_test.iloc[i], y_pred[i], y_pred_proba[i]))
confusion_matrix(y_test, y_pred).T
# Calculate ROC curve from y_test and pred
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
# Plot the ROC curve
fig = plt.figure(figsize=(8,8))
plt.title('Receiver Operating Characteristic')

# Plot ROC curve
plt.plot(fpr, tpr, label='l1')
plt.legend(loc='lower right')

# Diagonal 45 degree line
plt.plot([0,1],[0,1],'k--')

# Axes limits and labels
plt.xlim([-0.1,1.1])
plt.ylim([-0.1,1.1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
```

```
plt.show()
# Calculate AUC for Train set
print(roc_auc_score(y_train, y_train_pred))
# Calculate AUC for Test set
print(auc(fpr, tpr))
#### Feature Importance
# Building the model again with the best hyperparameters
LR_model = LogisticRegression(C=1, penalty = 'l2')
LR_model.fit(X_train, y_train)
indices = np.argsort(-abs(LR_model.coef_[0,:]))
print("The features in order of importance are:")
print(50*'-')
for feature in X.columns[indices]:
    print(feature)

## Model-2 Random Forest
tuned_params = {'n_estimators': [100, 200, 300, 400, 500], 'min_samples_split': [2, 5, 10],
'min_samples_leaf': [1, 2, 4]}
RF_model = RandomizedSearchCV(RandomForestClassifier(), tuned_params, n_iter=15, scoring =
'roc_auc', n_jobs=-1)
RF_model.fit(X_train, y_train)
RF_model.best_estimator_
y_train_pred = RF_model.predict(X_train)
y_pred = RF_model.predict(X_test)
# Get just the prediction for the positive class (1)
y_pred_proba = RF_model.predict_proba(X_test)[:,1]
# Display first 10 predictions
y_pred_proba[:10]
confusion_matrix(y_test, y_pred).T
# Calculate ROC curve from y_test and pred
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
# Plot the ROC curve
fig = plt.figure(figsize=(8,8))
plt.title('Receiver Operating Characteristic')

# Plot ROC curve
plt.plot(fpr, tpr, label='l1')
```

```
plt.legend(loc='lower right')

# Diagonal 45 degree line
plt.plot([0,1],[0,1],'k--')

# Axes limits and labels
plt.xlim([-0.1,1.1])
plt.ylim([-0.1,1.1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
# Calculate AUC for Train set
roc_auc_score(y_train, y_train_pred)
# Calculate AUC for Test set
print(auc(fpr, tpr))
#### Feature Importance
## Building the model again with the best hyperparameters
RF_model = RandomForestClassifier(n_estimators=500, min_samples_split=2, min_samples_leaf=4)
RF_model.fit(X_train, y_train)
indices = np.argsort(-RF_model.feature_importances_)
print("The features in order of importance are:")
print(50*'-')
for feature in X.columns[indices]:
    print(feature)
## Model-3 XGBoost
tuned_params = {'max_depth': [1, 2, 3, 4, 5], 'learning_rate': [0.01, 0.05, 0.1], 'n_estimators': [100, 200, 300,
400, 500], 'reg_lambda': [0.001, 0.1, 1.0, 10.0, 100.0]}
XGB_model = RandomizedSearchCV(XGBClassifier(), tuned_params, n_iter=15, scoring = 'roc_auc',
n_jobs=-1)
XGB_model.fit(X_train, y_train)
XGB_model.best_estimator_
y_train_pred = XGB_model.predict(X_train)
y_pred = XGB_model.predict(X_test)
# Get just the prediction for the positive class (1)
y_pred_proba = XGB_model.predict_proba(X_test)[:,1]
# Display first 10 predictions
y_pred_proba[:10]
```

```
confusion_matrix(y_test, y_pred).T
# Calculate ROC curve from y_test and pred
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
# Plot the ROC curve
fig = plt.figure(figsize=(8,8))
plt.title('Receiver Operating Characteristic')

# Plot ROC curve
plt.plot(fpr, tpr, label='l1')
plt.legend(loc='lower right')

# Diagonal 45 degree line
plt.plot([0,1],[0,1],'k--')

# Axes limits and labels
plt.xlim([-0.1,1.1])
plt.ylim([-0.1,1.1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
# Calculate AUC for Train
roc_auc_score(y_train, y_train_pred)
# Calculate AUC for Test
print(auc(fpr, tpr))
#### Feature Importance
XGB_model = XGBClassifier(max_depth=1,learning_rate=0.05,n_estimators=500, reg_lambda=1)
XGB_model.fit(X_train, y_train)
def my_plot_importance(booster, figsize, **kwargs):
    from matplotlib import pyplot as plt
    from xgboost import plot_importance
    fig, ax = plt.subplots(1,1,figsize=figsize)
    return plot_importance(booster=booster, ax=ax, **kwargs)
my_plot_importance(XGB_model, (10,10))
## Model-4 KNN
# creating odd list of K for KNN
neighbors = list(range(1,20,2))
# empty list that will hold cv scores
```

```
cv_scores = []

#  10-fold cross validation , 9 datapoints will be considered for training and 1 for cross validation (turn by turn) to determine value of k
for k in neighbors:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_train, y_train, cv=5, scoring='accuracy')
    cv_scores.append(scores.mean())

# changing to misclassification error
MSE = [1 - x for x in cv_scores]

# determining best k
optimal_k = neighbors[MSE.index(min(MSE))]
print('\nThe optimal number of neighbors is %d.' % optimal_k)
MSE.index(min(MSE))
# plot misclassification error vs k
plt.plot(neighbors, MSE)
plt.xlabel('Number of Neighbors K')
plt.ylabel('Misclassification Error')
plt.show()
classifier = KNeighborsClassifier(n_neighbors = optimal_k)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

y_train_pred = classifier.predict(X_train)
acc = accuracy_score(y_test, y_pred, normalize=True) * float(100)  ## get the accuracy on testing data
acc
cnf=confusion_matrix(y_test,y_pred).T
cnf
# Get just the prediction for the positive class (1)
y_pred_proba = classifier.predict_proba(X_test)[:,1]
# Display first 10 predictions
y_pred_proba[:10]
# Calculate ROC curve from y_test and pred
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
# Plot the ROC curve
```

```
fig = plt.figure(figsize=(8,8))
plt.title('Receiver Operating Characteristic')

# Plot ROC curve
plt.plot(fpr, tpr, label='l1')
plt.legend(loc='lower right')

# Diagonal 45 degree line
plt.plot([0,1],[0,1],'k--')

# Axes limits and labels
plt.xlim([-0.1,1.1])
plt.ylim([-0.1,1.1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
# Calculate AUC for Train
roc_auc_score(y_train, y_train_pred)
# Calculate AUC for Test
print(auc(fpr, tpr))
## Model-5 Descision Trees
tuned_params = {'min_samples_split': [2, 3, 4, 5, 7], 'min_samples_leaf': [1, 2, 3, 4, 6], 'max_depth': [2, 3, 4, 5, 6, 7]}
DT_model = RandomizedSearchCV(DecisionTreeClassifier(), tuned_params, n_iter=15, scoring = 'roc_auc', n_jobs=-1)
DT_model.fit(X_train, y_train)
DT_model.best_estimator_
y_train_pred = DT_model.predict(X_train)
y_pred = DT_model.predict(X_test)
y_pred_proba = DT_model.predict_proba(X_test)[:,1]
y_pred_proba[:10]
confusion_matrix(y_test, y_pred).T
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
# Plot the ROC curve
fig = plt.figure(figsize=(8,8))
plt.title('Receiver Operating Characteristic')
```

```
# Plot ROC curve
plt.plot(fpr, tpr, label='l1')
plt.legend(loc='lower right')

# Diagonal 45 degree line
plt.plot([0,1],[0,1],'k--')

# Axes limits and labels
plt.xlim([-0.1,1.1])
plt.ylim([-0.1,1.1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
# Calculate AUC for Train
roc_auc_score(y_train, y_train_pred)
print(auc(fpr, tpr))
#### Feature Importance
DT_model = DecisionTreeClassifier(min_samples_split=2, min_samples_leaf=6, max_depth=4)
DT_model.fit(X_train, y_train)
indices = np.argsort(-DT_model.feature_importances_)
print("The features in order of importance are:")
print(50*'-')
for feature in X.columns[indices]:
    print(feature)
## Model-6 SVC
from sklearn import svm
def svc_param_selection(X, y, nfolds):
    Cs = [0.001, 0.01, 0.1, 1, 10]
    gammas = [0.001, 0.01, 0.1, 1]
    param_grid = {'C': Cs, 'gamma' : gammas}
    grid_search = GridSearchCV(svm.SVC(kernel='rbf'), param_grid, cv=nfolds)
    grid_search.fit(X_train, y_train)
    grid_search.best_params_
    return grid_search.best_params_
svClassifier=SVC(kernel='rbf',probability=True)
svClassifier.fit(X_train,y_train)
svc_param_selection(X_train,y_train,5)
```

```
###### Building the model again with the best hyperparameters
SVC_model = SVC(C=1, gamma=1,probability=True)
SVC_model.fit(X_train, y_train)
## Predict Train results
y_train_pred = SVC_model.predict(X_train)
## Predict Test results
y_pred = SVC_model.predict(X_test)
confusion_matrix(y_test, y_pred).T
y_pred_proba = SVC_model.predict_proba(X_test)[:,1]
# Calculate ROC curve from y_test and pred
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
# Plot the ROC curve
fig = plt.figure(figsize=(8,8))
plt.title('Receiver Operating Characteristic')

# Plot ROC curve
plt.plot(fpr, tpr, label='l1')
plt.legend(loc='lower right')

# Diagonal 45 degree line
plt.plot([0,1],[0,1],'k--')

# Axes limits and labels
plt.xlim([-0.1,1.1])
plt.ylim([-0.1,1.1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
# Calculate AUC for Train
roc_auc_score(y_train, y_train_pred)
print(auc(fpr, tpr))
## Model-7 Gradient Boosting
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.ensemble import AdaBoostClassifier, BaggingClassifier
from sklearn.linear_model import Perceptron
```

```
from sklearn.linear_model import SGDClassifier
from sklearn.neural_network import MLPClassifier
#Import Library
from sklearn.ensemble import GradientBoostingClassifier
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor) of test_dataset
# Create Gradient Boosting Classifier object
gbclass = GradientBoostingClassifier(
                random_state = 1000,
                verbose = 0,
                n_estimators = 10,
                learning_rate = 0.9,
                loss = 'deviance',
                max_depth = 3
               )
#gbclass = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=1,
random_state=0)
# Train the model using the training sets and check score
gbclass.fit(X_train, y_train)
#Predict Output
predicted= gbclass.predict(X_test)

gbclass_score = round(gbclass.score(X_train, y_train) * 100, 2)
gbclass_score_test = round(gbclass.score(X_test, y_test) * 100, 2)
print('Score: \n', gbclass_score)
print('Test Score: \n', gbclass_score_test)
print('Accuracy: \n', accuracy_score(y_test,predicted))
print(confusion_matrix(predicted,y_test))
print(classification_report(y_test,predicted))
## Predict Train results
y_train_pred = gbclass.predict(X_train)
## Predict Test results
y_pred = gbclass.predict(X_test)
y_pred_proba = gbclass.predict_proba(X_test)[:,1]
# Calculate ROC curve from y_test and pred
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
# Plot the ROC curve
fig = plt.figure(figsize=(8,8))
```

```
plt.title('Receiver Operating Characteristic')

# Plot ROC curve
plt.plot(fpr, tpr, label='l1')
plt.legend(loc='lower right')

# Diagonal 45 degree line
plt.plot([0,1],[0,1],'k--')

# Axes limits and labels
plt.xlim([-0.1,1.1])
plt.ylim([-0.1,1.1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
roc_auc_score(y_train,y_train_pred )
# Calculate AUC for Test
print(auc(fpr, tpr))
```

7.3 MODEL.PY

```
import numpy as np

# for dataframes
import pandas as pd

# for easier visualization
import seaborn as sns

# for visualization and to display plots
from matplotlib import pyplot as plt
# %matplotlib inline

# import color maps
from matplotlib.colors import ListedColormap

# Ignore Warnings
import warnings
```

```
warnings.filterwarnings("ignore")

from math import sqrt

# to split train and test set
from sklearn.model_selection import train_test_split

# to perform hyperparameter tuning
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import cross_val_score

# Machine Learning Models
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import roc_curve, auc, roc_auc_score, confusion_matrix

from matplotlib.colors import ListedColormap
from sklearn.metrics import accuracy_score

class RFT_Model:
    model = None
    def __init__(self) -> None:
        df = pd.read_csv("Indian Liver Patient Dataset (ILPD).csv")

        ## if score==negative, mark 0 ;else 1
        def partition(x):
            if x == 2:
                return 0
            return 1
        df['Dataset'] = df['Dataset'].map(partition)

        """## Distribution of categorical data"""
        df.describe(include=['object'])

        df[df['Gender'] == 'Male'][['Dataset', 'Gender']].head()
        M, F = df['Gender'].value_counts()
```

```
## if score==negative, mark 0 ;else 1
def partition(x):
    if x =='Male':
        return 0
    return 1
df['Gender'] = df['Gender'].map(partition)

## Correlations
df.corr()

## Data Cleaning
df = df.drop_duplicates()

## Removing Outliers
df.Aspartate_Aminotransferase.sort_values(ascending=False).head()
df = df[df.Aspartate_Aminotransferase <=3000 ]
df.Aspartate_Aminotransferase.sort_values(ascending=False).head()
df = df[df.Aspartate_Aminotransferase <=2500 ]
df.isnull().values.any()
df=df.dropna(how='any')

### Data Preparation
# Create separate object for target variable
y = df.Dataset

# Create separate object for input features
X = df.drop('Dataset', axis=1)

# Split X and y into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1234,
stratify=df.Dataset)

# Print number of observations in X_train, X_test, y_train, and y_test
#print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

### Data standardization
```

```
        self.train_mean = X_train.mean()
        self.train_std = X_train.std()

        ## Standardize the train data set
        X_train = (X_train - self.train_mean) / self.train_std

        ## Check for mean and std dev.
        X_train.describe()

        ## Note: We use train_mean and train_std_dev to standardize test data set
        X_test = (X_test - self.train_mean) / self.train_std

        ## Check for mean and std dev. - not exactly 0 and 1
        X_test.describe()

        #Random Forest Tree Model
        tuned_params = {'n_estimators': [100, 200, 300, 400, 500], 'min_samples_split': [2, 5, 10],
'min_samples_leaf': [1, 2, 4]}
        self.model = RandomizedSearchCV(RandomForestClassifier(), tuned_params, n_iter=15, scoring =
'roc_auc', n_jobs=-1)
        self.model.fit(X_train, y_train)

        #self.model.best_estimator_

        y_train_pred = self.model.predict(X_train)

        y_pred = self.model.predict(X_test)

        # Get just the prediction for the positive class (1)
        y_pred_proba = self.model.predict_proba(X_test)[:,1]

        #### Feature Importance
        ## Building the model again with the best hyperparameters
        self.model = RandomForestClassifier(n_estimators=500, min_samples_split=2, min_samples_leaf=4)
        self.model.fit(X_train, y_train)

    def predict(self, test_data):
```

```
        test_data = (test_data - self.train_mean) / self.train_std
        pred_proba = self.model.predict_proba(test_data)[:,1]
        return pred_proba[0]
```

7.4 APP.PY

```
from flask import Flask, request, render_template, jsonify
import flask_cors
from model import RFT_Model
from model_copy import SVC_Model
import pandas as pd

app = Flask(__name__)
flask_cors.CORS(app)
ML_model = RFT_Model()

@app.route('/')
@flask_cors.cross_origin()
def home():
    return render_template('index.html')

@app.route('/predict')
@flask_cors.cross_origin()
def predict():
    global ML_model
    age = int(request.args.get('age'))
    gender = int(request.args.get('gender'))
    tb = float(request.args.get('tb'))
    db = float(request.args.get('db'))
    ap = float(request.args.get('ap'))
    aa = float(request.args.get('aa'))
    asa = float(request.args.get('asa'))
    a = float(request.args.get('a'))
    tp = float(request.args.get('tp'))
    agr = float(request.args.get('agr'))

    count = 0
    if( 0.22<=tb and tb<=1 ):
```

```
        count += 1
    if( 0<=db and db<=.2 ):
        count += 1
    if( 110<=ap and ap<=310 ):
        count += 1
    if( 5<=aa and aa<=45 ):
        count += 1
    if( 5<=asa and asa<=40 ):
        count += 1
    if( 3.5<=a and a<=5 ):
        count += 1
    if( 7.2<=tp and tp<=8 ):
        count += 1
    if( 1.7<=agr and agr<=2.2 ):
        count += 1
    if( 0.5<= count/8):
        count = True
    else:
        count = False

    data = pd.DataFrame({'Age':[age],'Gender':[gender],'Total_Bilirubin':[tb],
'Direct_Bilirubin':[db],'Alkaline_Phosphotase':[ap],'Alamine_Aminotransferase':[aa],
'Aspartate_Aminotransferase':[asa],'Albumin':[a],'Total_Protiens':[tp],'AG_Ratio':[agr]})
    res = ML_model.predict(data)
    res = int(res*100)
    if count:
        res = 100 - res
    response = jsonify({'result': res})
    return response

if __name__ == '__main__':
    app.run(host="127.0.0.1",port="5000",debug=True)
```

8. TESTING

8.1 TEST CASES

Age	30	40	50	60
Gender	Male	Female	Male	Female
Total Bilirubin	0.5	0.2	2	0.9
Direct Bilirubin	0.1	0.3	0.3	0.19
Alkaline Phosphotase	200	120	400	150
Alamine Aminotransferase (SGPT)	30	40	50	44
Aspartate Aminotransferase (SGOT)	25	45	48	10
Albumin	4	3	5.2	4.86
Total Proteins	7.6	8.1	8.9	7.89
A/G Ratio	2	2	2.5	2.05
Expected Output	0	1	1	0
Actual Output	0.32	0.75	0.89	0.22
Status	Pass	Pass	Pass	Pass

9. RESULTS

9.1 PERFORMANCE METRICS

Confusion matrix - For getting a better clarity of the no of correct/incorrect predictions by the model.

	ACTUAL	
PREDICTED	5	13
	27	68

Area Under the Curve (AUC) - is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

AUC = 0.719

10. ADVANTAGES & DISADVANTAGES

- Our study has successfully developed and vali- dated the first risk prediction model and subse- quent user-friendly scoring tool, the Algorithm for Liver Function Investigations, for liver condi- tion diagnosis in patients with no obvious liver condition at the time of incident liver function testing in primary care.
- This model can be used to facilitate general prac- titioner decision-making about whom to refer to secondary care.

- The observational data lacked some potential predictors of liver disease, for example, alcohol intake and body mass index. However, other available predictors such as liver function tests and deprivation may act as surrogate markers for such factors.

11. CONCLUSION

The principal part of this work is to make an effective diagnosis system for chorionic liver infection patients utilizing six distinctive supervised machine learning classifiers. We researched all classifiers execution on patient's information parameters and the LR classifier gives the most elevated order exactness 75% dependent on F1 measure to predict the liver disease and NB gives the least precision 53%. From now on, the outperform classification procedure will give for the decision support system and diagnosis of chronic disease. The application will have the option to predict liver infection prior and advise the wellbeing condition. This application can be surprisingly gainful in low salary nations where our absence of medicinal foundations and just as particular specialists. In our study, there are a few bearings for future work in this field. We just explored some popular supervised machine learning algorithms; more algorithms can be picked to assemble an increasingly precise model of liver disease prediction and performance can be progressively improved. Additionally, this work likewise ready to assume a significant role in health care research and just as restorative focuses to anticipate liver infection.

12. FUTURE SCOPE

With a single test value, we can’t predict whether the user have liver disease or not. Therefore in future we are planning to store the data given by each user in there login and make a prediction based on the past test reports by making this model real we can make more accurate prediction.

13. APPENDIX

- Statistical Machine Learning Approaches to Liver Disease Prediction - Fahad Mostafa, Easin Hasan, Morgan Williamson and Hafiz Khan. Dt: 01/12/2021.
- A Comparative Study on Liver Disease Prediction Using Supervised Machine Learning Algorithms - A.K.M Sazzadur Rahman, F. M. Javed Mehedi Shamrat, Zarrin Tasnim, Joy Roy, Syed Akhter Hossain. Dt: 11/11/2019.
- Liver Disease Prediction System using Machine Learning Techniques - Rakshith D B, Mrigank Srivastava, Ashwani Kumar, Gururaj S P. Dt: 06/06/2021.