PROJECT DOCUMENTATION

Statistical Machine Learning Approaches to Liver Disease Prediction

TEAM ID: PNT2022TMID48226

Team Leader: P.M.Priyadharshini

Team Member: N.N.Deepika

Team Member: T.G.Yamini

Team Member: S.S.Megha

Project Objectives:

By the end of this project:

- You'll be able to understand the problem to classify if it is a regression or a classification kind of problem.
- You will be able to know how to pre-process / clean the data using different data preprocessing techniques.
- You will be able to analyze or get insights of data through visualization.
- Applying different algorithms according to dataset and based on visualization.
- You will be able to know how to find the accuracy of the model.
- You will be able to know how to build a web application using the Flask framework.

Project Workflow:

- User interacts with the UI (User Interface) to upload the input features.
- Uploaded features/input is analyzed by the model which is integrated.
- Once a model analyses the uploaded inputs, the prediction is showcased on the UI.

To accomplish this, we have to complete all the activities and tasks listed below

1.Dataset Collection

- Collect the Dataset or Create the Dataset

2.Data Pre-Processing

- Importing the Libraries
- Reading the Dataset
- Exploratory Data Analysis
- Checking for Null Values and Handling Null Values
- Data Visualization
- Splititng the Dataset into Dependent and Independent
- Split the Dependent and independent Features into Train Set and Test Set

3.Model Building

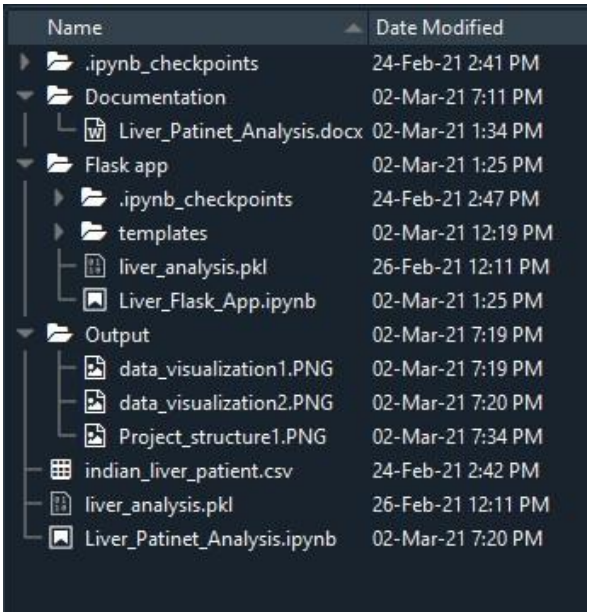
- Train and Test the Model Using Classification Algorithms
- Model Evaluation
- Save the Model

4. Application Building

- Create an HTML File
- Build Python Code
- Run the App

Project Structure:

Create a Project folder that contains files as shown below



We have three folders dataset, Flask and Training.

- Dataset has dataset indian_liver_patient.csv.
- A python file called Liver_Flask_App.py for server side scripting.
- We need the model which is saved and the saved model in this content is liver_analysis.pkl.
- Templates folder which contains home.html and upload.html files.
- Training folder has Liver_Patient_Analysis.ipynb where the model is created and saved.
- Static folder which contains css(styling), fontawe some(styling), img(images), js(Java script) folders to enhance the features of the web page.

Prerequisites:

In order to develop this project we need to install following software/packages .

Anaconda Navigator :

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupyter notebook and Spyder.

To build Machine learning models you must require the following packages:

- **Numpy:**
    - It is an open-source numerical Python library. It contains a multidimensional array and matrix data structures and can be used to perform mathematical operations.
- **Scikit-learn:**
    - It is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy.
- **Matplotlib and Seaborn:**
    - Matplotlib is mainly deployed for basic plotting. Visualization using Matplotlib generally consists of bars, pies, lines, scatter plots and so on. Seaborn: Seaborn, on the other hand, provides a variety of visualization patterns. It uses fewer syntax and has easily interesting default themes.
- **Pandas:**
    - It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
- **Pickle:**
    - The pickle module implements serialization protocol, which provides an ability to save and later load Python objects using special binary format.

If you are using **anaconda navigator**, follow below steps to download required packages:

- Open the anaconda prompt.
- Type “pip install jupyter notebook” and click enter.
- Type “pip install spyder” and click enter.
- Type “pip install numpy” and click enter. • Type “pip install pandas” and click enter.
- Type “pip install matplotlib” and click enter.
- Type “pip install seaborn” and click enter.
- Type “pip install sklearn” and click enter.
- Type “pip install Flask” and click enter.

If you are using Pycharm IDE, you can install the packages through the command prompt and follow the same syntax as above.

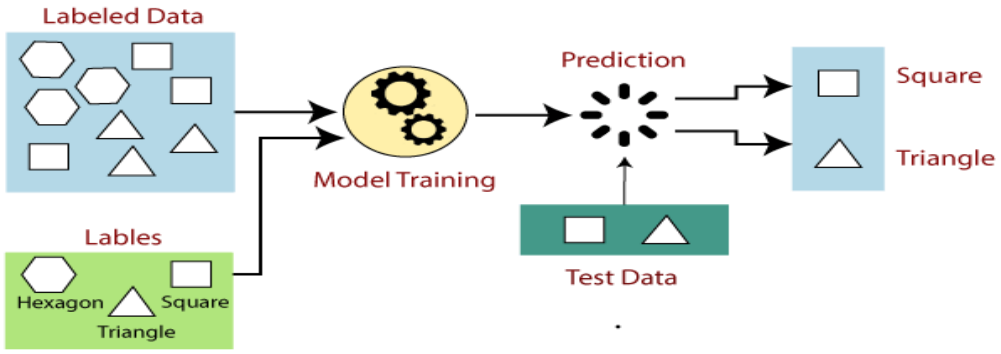
Prior Knowledge:

- Supervised learning
- Unsupervised learning

- Machine Learning in R – Classification
- Regression and Clustering Problems

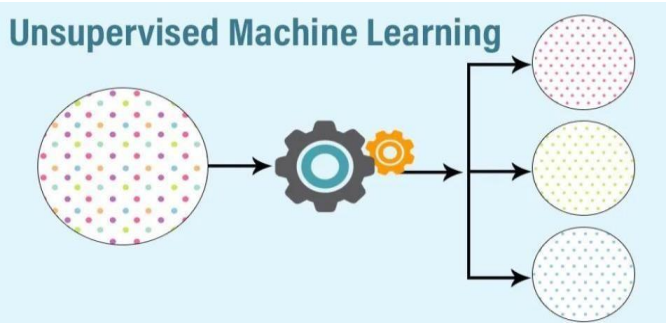
Supervised Learning:

Supervised learning is a machine learning approach that's defined by its use of labeled datasets. These datasets are designed to train or "supervise" algorithms into classifying data or predicting outcomes accurately.

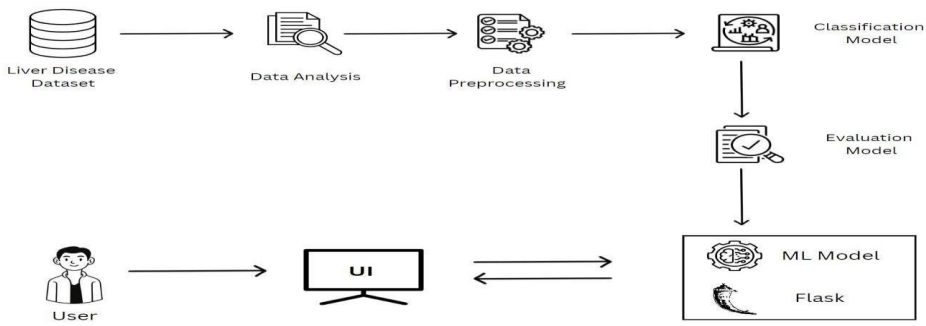


Unsupervised Learning:

Unsupervised learning uses machine learning algorithms to analyze and cluster unlabeled data sets. These algorithms discover hidden patterns in data without the need for human intervention (hence, they are "unsupervised").



Workflow Diagram:



Dataset Collection & Data Preprocessing

Tasks:

There are two tasks:

1. Dataset Collection
2. Data Preprocessing

1.Dataset Collection:

ML depends heavily on data, without data, a machine can't learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training data set. It is the actual data set used to train the model for performing various actions.

You can collect dataset from different open sources like kaggle.com, data.gov, UCI machine learning repository etc.

Dataset Link:

https://www.kaggle.com/datasets/uciml/indian-liver-patient-records

2.Data Preprocessing:

Data Pre-processing includes the following main tasks

1. Import the Libraries.
2. Reading the dataset.
3. Analyse the data.
4. Taking Care of Missing data.
5. Data Visualization.
6. Splitting the Dataset into Dependent and Independent variables.
7. Splitting Data into Train and Test

1.Importing The Libraries:

The first step is usually importing the libraries that will be needed in the program. The required libraries to be imported to Python script are:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
```

Numpy:

It is an open-source numerical Python library. It contains a multi-dimensional array and matrix data structures. It can be used to perform mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.

Example:

```
import numpy as np
a = np.array([0, 1, 2, 3])        # Create a rank 1 array
print(a)                          #print array a
print(type(a))                    #type of array a
print(a.ndim)                     #dimension of array a
print(a.shape)                    #shape(row,column) of array a
print(len(a))                     #length of array a
```

```
[0 1 2 3]
<class 'numpy.ndarray'>
1
(4,)
4
```

Pandas:

It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

Example:

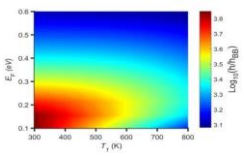
```
label = ['a', 'b', 'c']
my_data = [10, 20, 30]
pd.Series(data = my_data, index = label)
```

```
a    10
b    20
c    30
dtype: int64
```

Matplotlib:

Visualisation with python. It is a comprehensive library for creating static, animated, and interactive visualizations in Python.

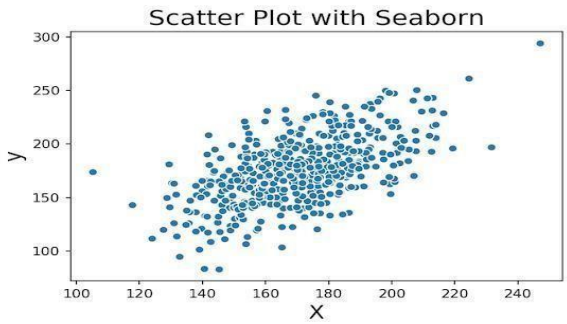
Example:



Seaborn:

Seaborn is a library for making statistical graphics in Python. Seaborn helps you explore and understand your data. Its plotting functions operate on dataFrames and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

Example:



Pickle:

The pickle module implements serialization protocol, which provides an ability to save and later load Python objects using special binary format

2.Reading the Dataset:

- You might have your data in .csv files, .excel files or .**tsv** files or something else. But the goal is the same in all cases. If you want to analyse that data using pandas, the first step will be to read it into a data structure that's compatible with pandas.

- Let's load a .csv data file into pandas. There is a function for it, called **read_csv().**We will need to locate the directory of the CSV file at first (it's more efficient to keep the dataset in the same directory as your program).

```
#import the dataset from specified location
data = pd.read_csv('E:/Datascience/Datasets/indian_liver_patient.csv')
```

Content:

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

Note: We have not started any data analysis yet, this is just to show you all the authenticity of the dataset.

3.Exploratory Data Analysis:

The exploratory data analysis (EDA) notebook is designed to assist you with discovering patterns in data, checking data sanity, and summarizing the relevant data for predictive models.

The EDA notebook example was optimized with web-based data in mind and consists of two parts. Part one starts with using Query Service to view trends and data snapshots. Next, with a goal in mind for exploratory data analysis, the data is aggregated at the profile and visitor level.

[4]:

	Year	Month	Count_days	First_date	Last_date	Count_hits
0	2020	1	1	31	31	117060
1	2020	2	29	1	29	3503948

head() :To check the first five rows of the dataset, we have a function called **head( ).**

```
# showing the data from top 5
data.head()
```

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferas
0	65	Female	0.7	0.1	187	16	18
1	62	Male	10.9	5.5	699	64	100
2	62	Male	7.3	4.1	490	60	68
3	58	Male	1.0	0.4	182	14	20
4	72	Male	3.9	2.0	195	27	59

tail(): To check the last five rows of the dataset, we have a function called **tail().**

```
data.tail()
```

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransfer
578	60	Male	0.5	0.1	500	20	34
579	40	Male	0.6	0.1	98	35	31
580	52	Male	0.8	0.2	245	48	49
581	31	Male	1.3	0.5	184	29	32
582	38	Male	1.0	0.3	216	21	24

Will see how our dataset is, by using the **info()** method.

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 583 entries, 0 to 582
Data columns (total 11 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Age                         583 non-null    int64
 1   Gender                      583 non-null    object
 2   Total_Bilirubin             583 non-null    float64
 3   Direct_Bilirubin            583 non-null    float64
 4   Alkaline_Phosphotase        583 non-null    int64
 5   Alamine_Aminotransferase    583 non-null    int64
 6   Aspartate_Aminotransferase  583 non-null    int64
 7   Total_Protiens              583 non-null    float64
 8   Albumin                     583 non-null    float64
 9   Albumin_and_Globulin_Ratio  579 non-null    float64
 10  Dataset                     583 non-null    int64
dtypes: float64(5), int64(5), object(1)
memory usage: 50.2+ KB
```

describe(): functions are used to compute values like count, mean, standard deviation and IQR(Inter Quantile Ranges) and give a summary of numeric type data.

```
data.describe()
```

```
data.describe()
```

	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransfera
count	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000
mean	44.746141	3.298799	1.486106	290.576329	80.713551	109.910806
std	16.189833	6.209522	2.808498	242.937989	182.620356	288.918529
min	4.000000	0.400000	0.100000	63.000000	10.000000	10.000000
25%	33.000000	0.800000	0.200000	175.500000	23.000000	25.000000
50%	45.000000	1.000000	0.300000	208.000000	35.000000	42.000000
75%	58.000000	2.600000	1.300000	298.000000	60.500000	87.000000
max	90.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000

4.Checking for null values and Handling Null Values:

This method commonly used to handle the null values. Here, we either delete a particular row if it has a null value for a particular feature and a particular column if it has more than 70-75% of missing values. This method is advised only when there are enough samples in the data set. Missing values can be handled by deleting the rows or columns having null values. If columns have more than half of the rows as null then the entire column can be dropped. The rows which are having one or more columns values as null can also be dropped.

- We will be using isnull().any() method to see which column has missing values.
- This isnull().any() method return two values, False and True.
- False return that Column has No Null Values.
- True return that Column has Null values.

```
data.isnull().any()

Age                          False
Gender                       False
Total_Bilirubin              False
Direct_Bilirubin             False
Alkaline_Phosphotase         False
Alamine_Aminotransferase     False
Aspartate_Aminotransferase   False
Total_Protiens               False
Albumin                      False
Albumin_and_Globulin_Ratio    True
Dataset                      False
dtype: bool
```

Let us check how many numbers of null records present in the Closing Value column using sum() function.

```
#checking for missing data
data.isnull().sum()
```

```
Age                          0
Gender                       0
Total_Bilirubin              0
Direct_Bilirubin             0
Alkaline_Phosphotase         0
Alamine_Aminotransferase     0
Aspartate_Aminotransferase   0
Total_Protiens               0
Albumin                      0
Albumin_and_Globulin_Ratio   4
Dataset                      0
dtype: int64
```

We can notice that, there are 4 null values are there in the column Albumin_and_Globulin_Ratio. Now will handle or fill that null values with the help of fillna() method.

```
#checking for the missing data after cleaning data
data['Albumin_and_Globulin_Ratio'] = data.fillna(data['Albumin_and_Globulin_Ratio'].mode()[0])
data.isnull().sum()
```

```
Age                          0
Gender                       0
Total_Bilirubin              0
Direct_Bilirubin             0
Alkaline_Phosphotase         0
Alamine_Aminotransferase     0
Aspartate_Aminotransferase   0
Total_Protiens               0
Albumin                      0
Albumin_and_Globulin_Ratio   0
Dataset                      0
dtype: int64
```

In real world data, there are some instances where a particular element is absent because of various reasons, such as, corrupt data, failure to load the information, or incomplete extraction. Handling the missing values is one of the greatest challenges faced by analysts, because making the right decision on how to handle it generates robust data models. Let us look at different ways of imputing the missing values.

5.Data Visualization:

Data visualization is a field in data analysis that deals with visual representation of data. It graphically plots data and is an effective way to communicate inferences from data.

Using data visualization, we can get a visual summary of our data. With pictures, maps and graphs, the human mind has an easier time processing and understanding any given data. Data visualization plays a significant role in the representation of both small and large data sets, but it is especially useful when we have large data sets, in which it is impossible to see all of our data, let alone process and understand it manually.

Univariate Analysis:

Univariate analysis is the simplest form of data analysis where the data being analyzed contains only one variable.

Bivariate Analysis:

It involves the analysis of two variables (often denoted as X, Y), for the purpose of determining the empirical relationship between them.

1. Bar plot between Gender and Count:

2. Bar Plot which describe about the total number of disease:

6.Splitting the Data-set into Independent and Dependent:

Any predictive mathematical model tends to divide the observations (data) into dependent/ independent features in order to determine the causal effect. It should be noted that relationship between dependent and independent variables need not be linear, it can be polynomial. It is common practise while doing experiments to change one independent variable while keeping others constant to see the change caused on the dependent variable.

Splitting the Data-set into Independent and Dependent Features:

In machine learning, the concept of dependent and independent variables is important to understand. In the above dataset, if you look closely, the first four columns (Item_Category, Gender, Age, Salary) determine the outcome of the fifth, or last, column (Purchased). Intuitively, it means that the decision to buy a product of a given category (Fitness item, Food product, kitchen goods) is determined by the Gender (Male, Female), Age, and the Salary of

the individual. So, we can say that Purchased is the dependent variable, the value of which is determined by the other four variables.

Skill Tags:

Let’s split our dataset into independent and dependent variables.

1.The independent variable in the dataset would be considered as 'x'.

2.The dependent variable in the dataset would be considered as 'y'.

Now we will split the data of independent variables.

Splitting the Dataset into the Independent Feature Matrix:

```
X = df.iloc[:, :-1].values
print(X)
```

Output:

```
[['Fitness' 'Male' 20 30000]
['Fitness' 'Female' 50 70000]
['Food' 'Male' 35 50000]
['Kitchen' 'Male' 22 40000]
['Kitchen' 'Female' 30 35000]]
```

7.Split the Dependent and Independent Features into Train set and Test set:

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced.

After completing this tutorial, you will know:

- The train-test split procedure is appropriate when you have a very large dataset, a costly model to train, or require a good estimate of model performance quickly.
- How to use the scikit-learn machine learning library to perform the train-test split procedure.

- How to evaluate machine learning algorithms for classification and regression using the train-test split.

- When you are working on a model and you want to train it, you obviously have a

dataset. But after training, we have to test the model on some test dataset. For this, you will a dataset which is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.

- But the question is, how do you split the data? You can't possibly manually split the dataset into two sets. And you also have to make sure you split the data in a random manner. To help us with this task, the Scikit library provides a tool, called the Model Selection library. There is a class in the library which is,'train_test_split.' Using this we can easily split the dataset into the training and the testing datasets in various proportions.
- The train-test split is a technique for evaluating the performance of a machine learning algorithm.
- **Train Dataset:** Used to fit the machine learning model.
- **Test Dataset:** Used to evaluate the fit machine learning model.
- In general you can allocate 80% of the dataset to training set and the remaining 20% to test set.We will create 4 sets— x_train (training part of the matrix of features), x_test (test part of the matrix of features), y_train (training part of the dependent variables associated with the X train sets, and therefore also the same indices), y_test (test part of the dependent variables associated with the X test sets, and therefore also the same indices.
- There are a few other parameters that we need to understand before we use the class:
- **Test_size** — this parameter decides the size of the data that has to be split as the test dataset. This is given as a fraction. For example, if you pass 0.5 as the value, the dataset will be split 50% as the test dataset
- **Train_size** — you have to specify this parameter only if you're not specifying the test_size. This is the same as test_size, but instead you tell the class what percent of the dataset you want to split as the training set.
- **Random_state** — here you pass an integer, which will act as the seed for the random number generator during the split. Or, you can also pass an instance of the Random_state class, which will become the number generator. If you don't pass anything, the Random_state instance used by np.random will be used instead.
- Now split our dataset into train set and test using train_test_split class from scikit learn library.

Check the shape of both xtrain and xtest.

```
xtrain.shape

(466, 10)

xtest.shape

(117, 10)
```

```
...
 # split into train test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, ...)
```

MODEL BUILDING

Model Building:

Predictive modeling is a mathematical approach to create a statistical model to forecast future behavior based on input test data.

Model Prediction:

We make predictions by giving the input test data to the trained model. We measure the accuracy by using a cross-validation strategy or ROC curve which performs well to derive model output for test data.

Model building includes the following main tasks

1. Training and testing the model
2. Evaluation of Model
3. Save the model
4. Predicting the output using the model

1.Train And Test The Model Using Classification Algorithms:

There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values. The algorithms that you can choose according to the objective that you might have may be Classification algorithms are Regression algorithms.

Example:

1. Random Forest Classification.

2. Support Vector Machine

3. KNN Classification

Build the model

We're going to use x_train and y_train obtained above in train_test_split section to train our regression model. We're using the fit method and passing the parameters as shown below.

1. Import the Classification algorithms

```
# Importing the machine learning model
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

2.Initialize the model

```
# Initializing the machine learning models
svm=SVC()
RFmodel=RandomForestClassifier()
KNNmodel=KNeighborsClassifier()
```

3. Training model with our data.

- SVC Model

```
#Support Vector Machine Model
from sklearn.svm import SVC
svm=SVC()

# train the data with SVM model
svm.fit(xtrain, ytrain)

SVC()
```

Random Forest Model

```
#Random Forest Classifier Model
from sklearn.ensemble import RandomForestClassifier
RFmodel=RandomForestClassifier()

# train the data with Random Forest model
RFmodel.fit(xtrain, ytrain)

RandomForestClassifier()
```

2.**Model Evaluation:**

Finally, we need to check to see how well our model is performing on the test data.

Evaluation Metrics:

accuracy_score of SVM is

```
# Checking for accuracy score from actual data and predicted data
SVMaccuracy=accuracy_score(SVMpred, ytest)
SVMaccuracy

0.7606837606837606
```

accuarcy_score of Random forest classification is

```
#Random Forest Classifier Model
from sklearn.ensemble import RandomForestClassifier
RFmodel=RandomForestClassifier()

# train the data with Random Forest model
RFmodel.fit(xtrain, ytrain)

RandomForestClassifier()

RFpred=RFmodel.predict(xtest)

# Checking for accuracy score from actual data and predicted data
RFaccuracy=accuracy_score(RFpred, ytest)
RFaccuracy

0.7094017094017094

# showing the confusion matrix
RFcm=confusion_matrix(RFpred, ytest)
RFcm

array([[77, 22],
       [12,  6]], dtype=int64)
```

accuracy_score of KNN classification is

```
# K-Nearest Neighbors Model
from sklearn.neighbors import KNeighborsClassifier
KNN = KNeighborsClassifier()

# train the data with K-Nearest Neighbors Model
KNN.fit(xtrain, ytrain)

KNeighborsClassifier()

KNNpred=KNN.predict(xtest)

# Checking for accuracy score from actual data and predicted data
KNNaccuracy=accuracy_score(KNNpred, ytest)
KNNaccuracy

0.6495726495726496

# showing the confusion matrix
KNNcm=confusion_matrix(KNNpred, ytest)
KNNcm

array([[70, 22],
       [19,  6]], dtype=int64)
```

As we can see that the accuracy_score of the Support vector machine is higher compare to KNN and Random forest algorithms, we are proceeding with the support vector machine model.

```
# Model Evaluation

In [4]: model = ensemble.RandomForestClassifier()
        model.fit(X_train, y_train) #Put X_Train.values while running The app.py---
        y_pred = model.predict(X_test)
        print('Accuracy : {}'.format(accuracy_score(y_test, y_pred)))

        clf_report = classification_report(y_test, y_pred)
        print('Classification report')
        print("--------------------")
        print(clf_report)
        print("__________________")

        Accuracy : 1.0
        Classification report
        --------------------
                      precision    recall  f1-score   support

                   1       1.00      1.00      1.00      2038
                   2       1.00      1.00      1.00       836

            accuracy                           1.00      2874
           macro avg       1.00      1.00      1.00      2874
        weighted avg       1.00      1.00      1.00      2874

        __________________
```

3.Save The Model:

After building the model we have to save the model.

Pickle is used for serializing and de-serializing Python object structures, also called marshalling or flattening. Serialization refers to the process of converting an object in memory to a byte stream that can be stored on disk or sent over a network. Later on, this character stream can then be retrieved and de-serialized back to a Python object.

This is done by the below code

```
# saving the model
import pickle
pickle.dump(svm, open('liver_analysis.pkl','wb'))
```

Here, svm is our Support Vector Machine Classification class, saving as liver_analysis.pkl file. Wb is the write binary in bytes.

Steps to Save The Model:

- model. fit (X_train, Y_train) # save the model to disk.
- filename ='finalized_model.sav'
- pickle. dump (model, open (filename,'wb'))
- # load the model from disk.

○ loaded_model = pickle.load(open(filename, 'rb')) ○ result = loaded_model.score(X_test, Y_test)

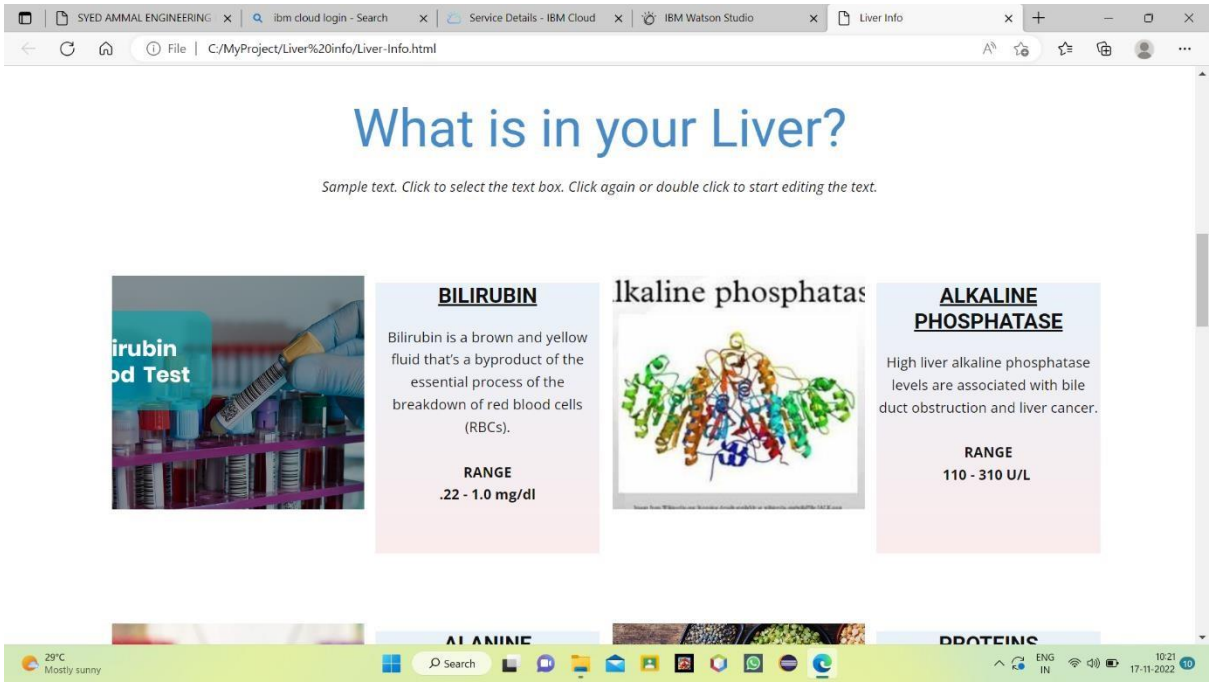
Application Building

Application Building involves following steps:

1. Create an HTML file
2. Build a Python Code
3. Run the app

1.Create An HTML File:

- We use HTML to create the front-end part of the web page.
- Here, we created 2 html pages- home.html, index.html.
- home.html displays the home page.
- index.html accepts the values from the user and displays the prediction.
- We also use JavaScript-main.js and CSS-main.css to enhance our functionality and view of HTML page



2.Build python code:

Python package builds are the product of coordination between a few different tools driven by a standardized process. One of the biggest choices you have as a package author is which set of tools to use. It can be difficult to assess the nuances of each, especially if you're new to packaging. Fortunately, tools are standardizing around the same core workflow, so once you learn it you've got the agility to switch between tools with minimal effort This article covers what you first need to learn about the pieces of the Python build system itself.

Importing Libraries:

```
from flask import Flask, render_template, request # Flask is a application
# used to run/serve our application
# request is used to access the file which is uploaded by the user in out application
# render_template is used for rendering the html pages
import pickle # pickle is used for serializing and de-serializing Python object structures
```

Libraries required for the app to run are to be imported.

Creating our flask app and loading the model

```
app=Flask(__name__) # our flask app
```

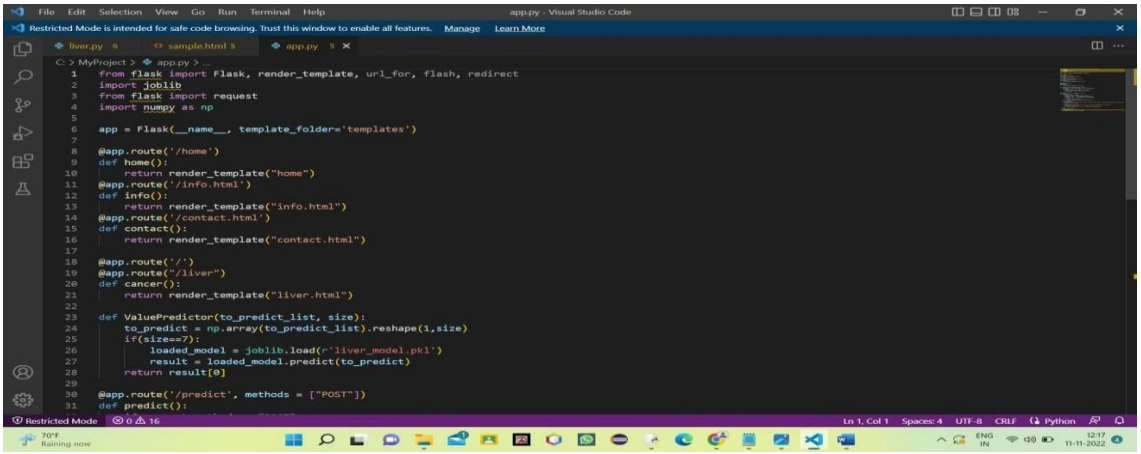
Now after all the libraries are imported, we will be creating our flask app. and the load our model into our flask app.

Routing to the html Page:

@app.route is used to route the application where it should route to. '/' URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page is rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method. Here, "home.html" is rendered when the home button is clicked on the UI and "index.html" is rendered when the predict button is clicked.

```
@app.route('/data_predict', methods=['POST']) # route for our prediction
def predict():
    age = request.form['age'] # requesting for age data
    gender = request.form['gender'] # requesting for gender data
    tb = request.form['tb'] # requesting for Total_Bilirubin data
    db = request.form['db'] # requesting for Direct_Bilirubin data
    ap = request.form['ap'] # requesting for Alkaline_Phosphotase data
    aa1 = request.form['aa1'] # requesting for Alamine_Aminotransferase data
    aa2 = request.form['aa2'] # requesting for Aspartate_Aminotransferase data
    tp = request.form['tp'] # requesting for Total_Protiens data
    a = request.form['a'] # requesting for Albumin data
    agr = request.form['agr'] # requesting for Albumin_and_Globulin_Ratio data

    # coverting data into float format
    data = [[float(age), float(gender), float(tb), float(db), float(ap), float(aa1), float(aa2), float(tp),

    # loading model which we saved
    model = pickle.load(open('liver_analysis.pkl', 'rb'))

    prediction= model.predict(data)[0]
    if (prediction == 1):
        return render_template('noChance.html', prediction='You have a liver desease problem, You must and
    else:
        return render_template('chance.html', prediction='You dont have a liver desease problem')

if __name__ == '__main__':
    app.run()
```

Here the route for prediction is given and necessary steps are performed in order to get the predicted output.Lastly, we run our app on the local host. Here we are running it on localhost:8087



3.Run the App:

Run the application from anaconda prompt:

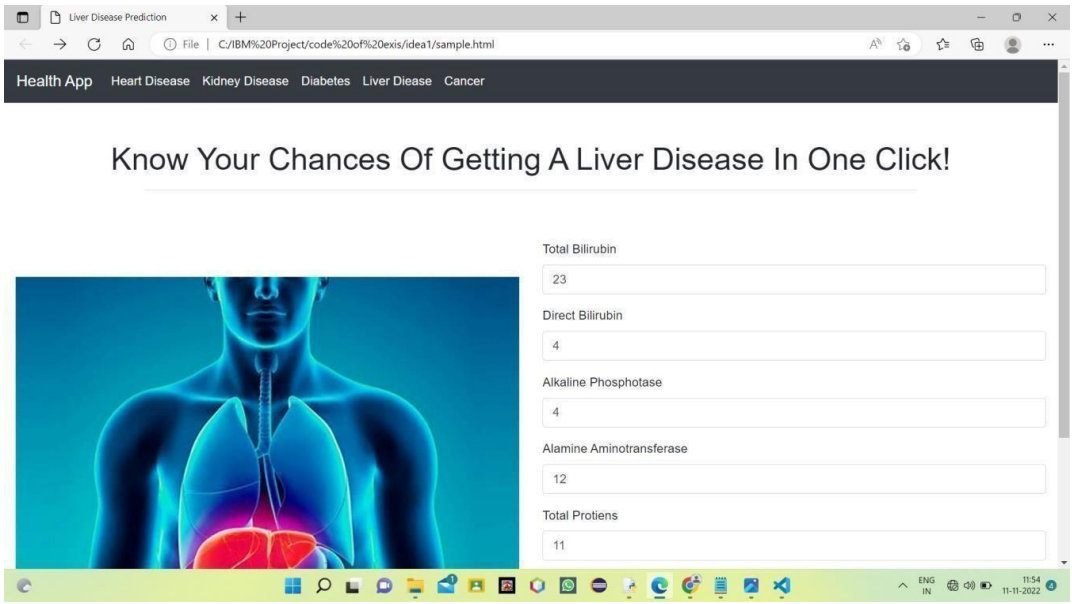
- Open new anaconda prompt from the start menu . Navigate to the folder where your python script is.
- Now type "python app.py" command

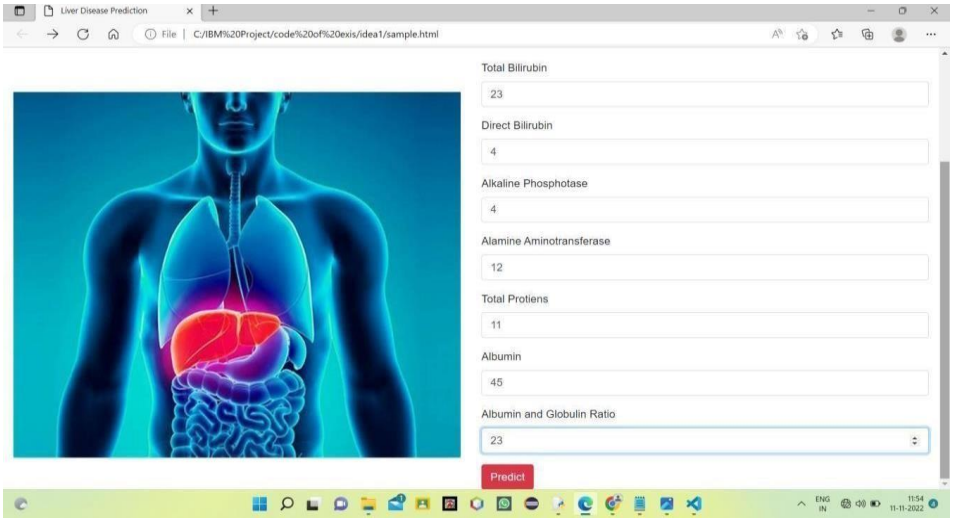
- It will show the local host where your app is running on **http://127.0.0.1.8000/**
- Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.
- Enter the values, click on the predict button and see the result/prediction on the web page.

```python
app=Flask(__name__) # our flask app

@app.route('/') # rendering the html template
def home():
    return render_template('home.html')
@app.route('/predict') # rendering the html template
def index() :
    return render_template("index.html")

@app.route('/data_predict', methods=['POST']) # route for our prediction
def predict():
    age = request.form['age'] # requesting for age data
    gender = request.form['gender'] # requesting for gender data
    tb = request.form['tb'] # requesting for Total_Bilirubin data
    db = request.form['db'] # requesting for Direct_Bilirubin data
    ap = request.form['ap'] # requesting for Alkaline_Phosphotase data
    aa1 = request.form['aa1'] # requesting for Alamine_Aminotransferase data
    aa2 = request.form['aa2'] # requesting for Aspartate_Aminotransferase data
    tp = request.form['tp'] # requesting for Total_Protiens data
    a = request.form['a'] # requesting for Albumin data
    agr = request.form['agr'] # requesting for Albumin_and_Globulin_Ratio data

    # coverting data into float format
    data = [[float(age), float(gender), float(tb), float(db), float(ap), float(aa1), float(aa2), float(tp),

    # loading model which we saved
    model = pickle.load(open('liver_analysis.pkl', 'rb'))

    prediction= model.predict(data)[0]
    if (prediction == 1):
        return render_template('noChance.html', prediction='You have a liver desease problem, You must and :
    else:
        return render_template('chance.html', prediction='You dont have a liver desease problem')

if __name__ == '__main__':
    app.run()
```

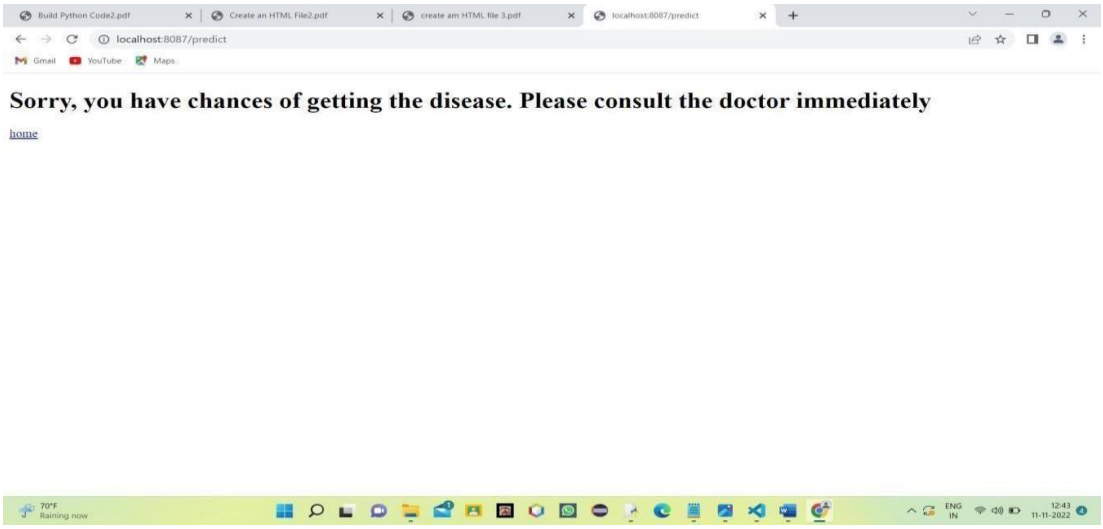
```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off

 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Home page is displayed when home button is clicked. Predict page is displayed when predict button is clicked. In predict page, enter input values to predict the liver disease or not. Finally, the prediction for the given input features is shown.

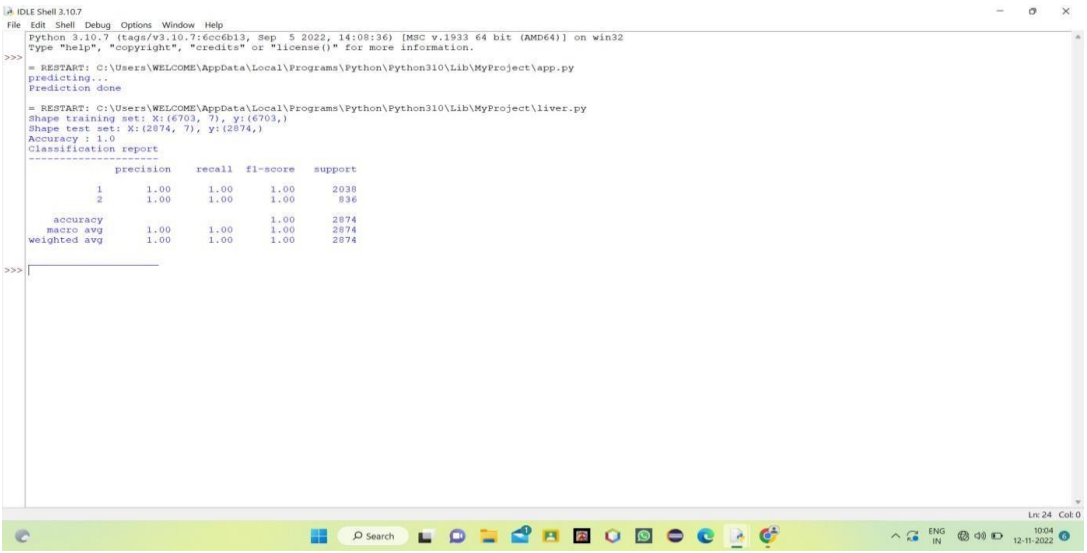




Output:



Run The App:

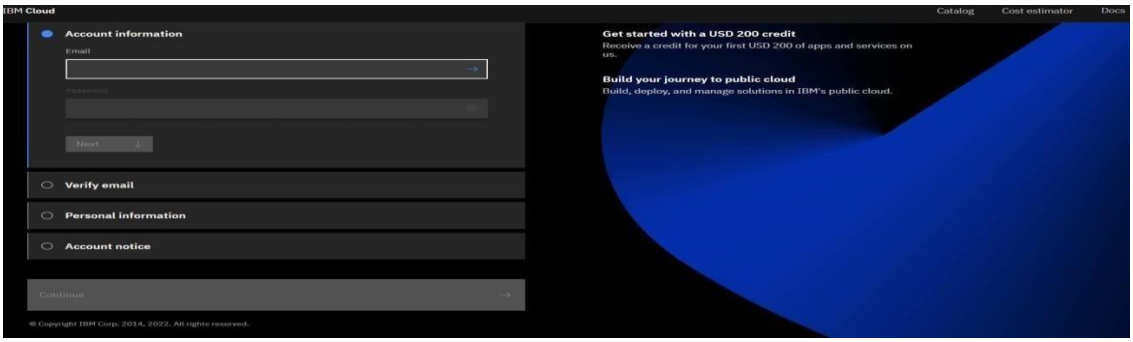


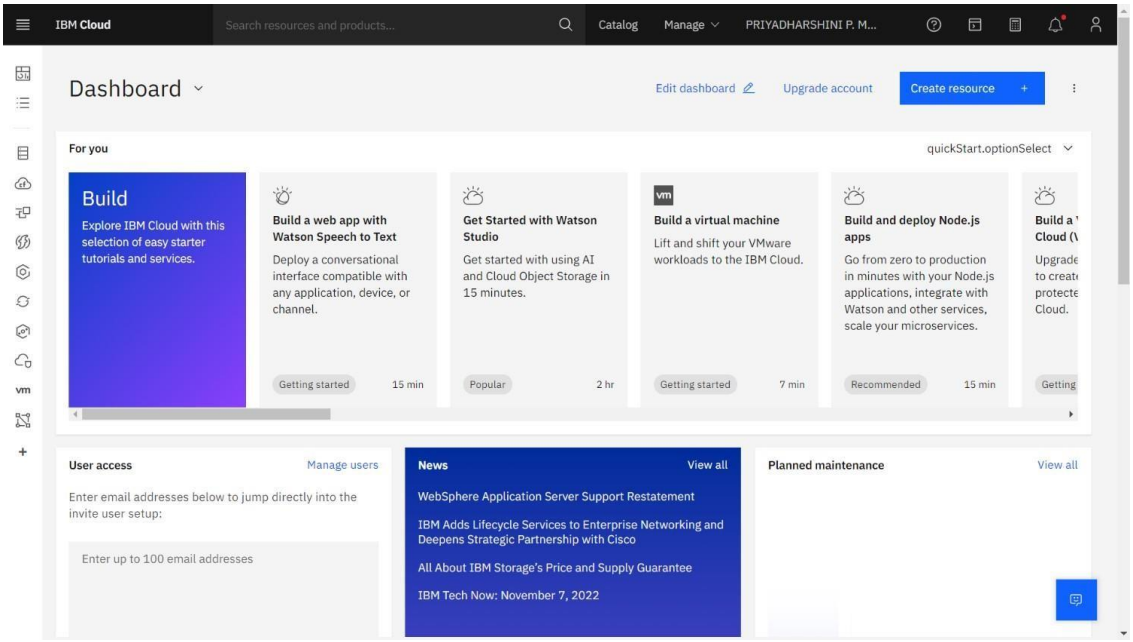
Train the Model on IBM

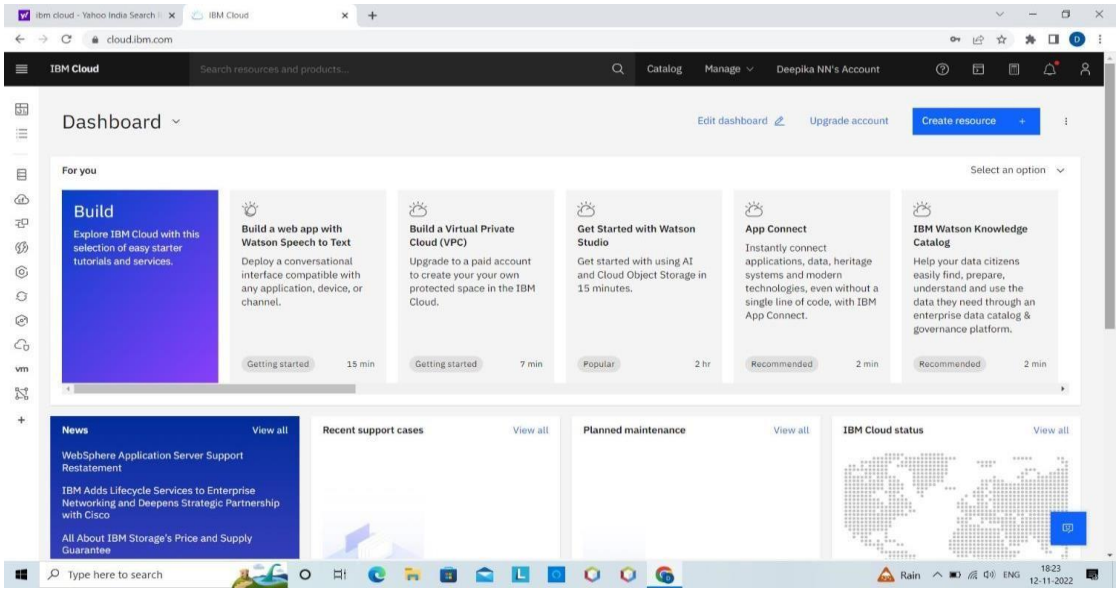
1.Register For IBM Cloud :

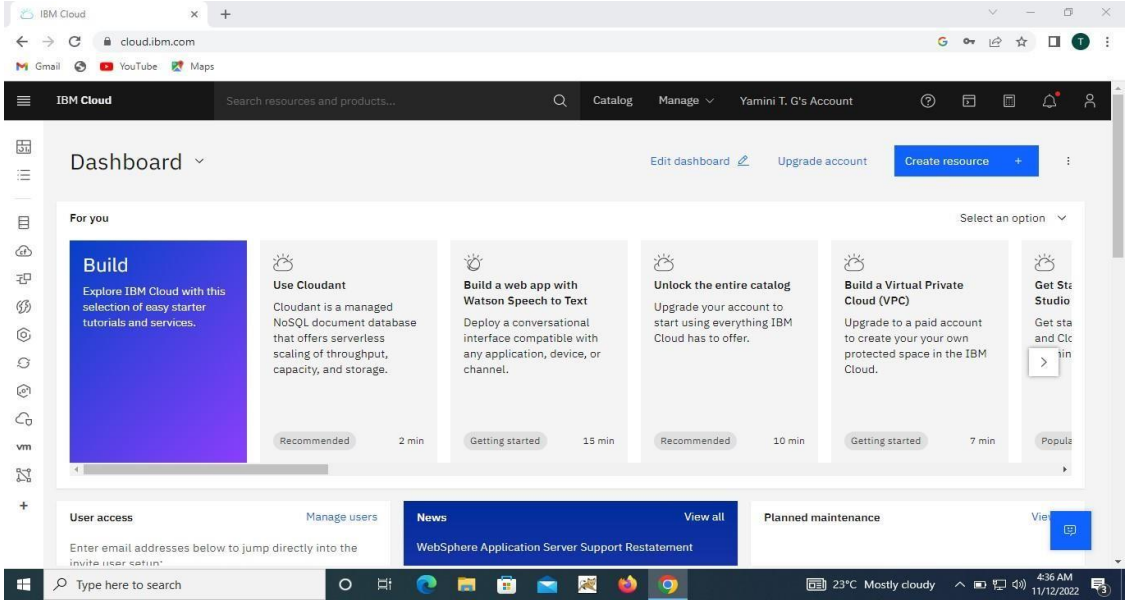
Using your IBMid:

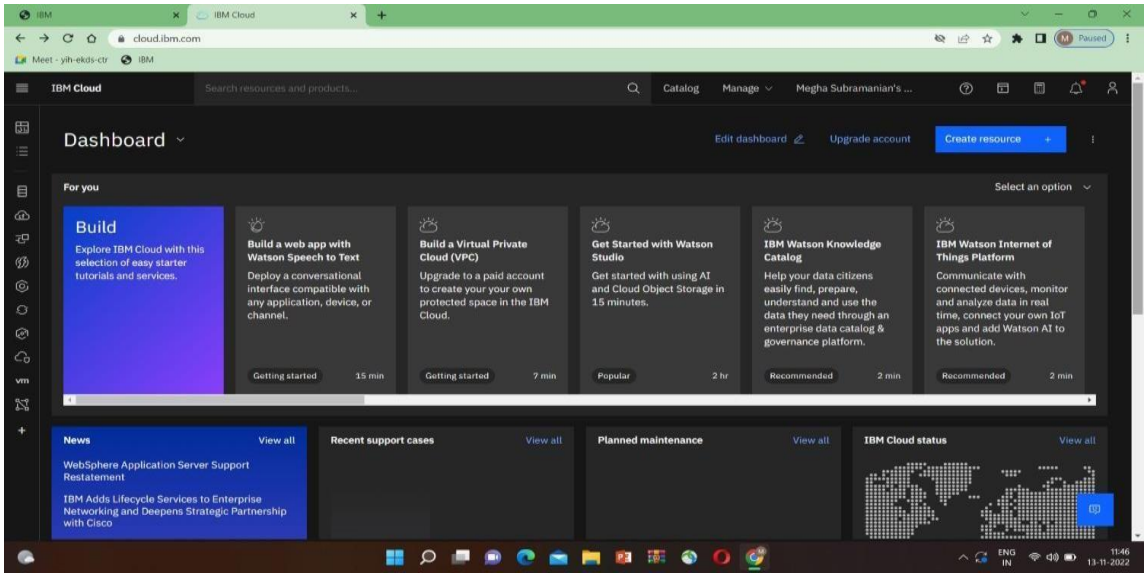
- Go to the IBM Cloud login page, and click Create an IBM Cloud account.
- Enter your IBMid email address. ...
- Complete the remaining fields with your information. ...
- Click Create account.
- Confirm your account by clicking the link in the confirmation email that's sent to your provided email address.






ibm cloud - Yahoo India Search
IBM Cloud
cloud.ibm.com
IBM Cloud
Search resources and products...
Catalog
Manage
Deepika NN's Account
Dashboard
Edit dashboard
Upgrade account
Create resource
For you
Select an option
Build
Explore IBM Cloud with this selection of easy starter tutorials and services.
Build a web app with Watson Speech to Text
Deploy a conversational interface compatible with any application, device, or channel.
Getting started
15 min
Build a Virtual Private Cloud (VPC)
Upgrade to a paid account to create your your own protected space in the IBM Cloud.
Getting started
7 min
Get Started with Watson Studio
Get started with using AI and Cloud Object Storage in 15 minutes.
Popular
2 hr
App Connect
Instantly connect applications, data, heritage systems and modern technologies, even without a single line of code, with IBM App Connect.
Recommended
2 min
IBM Watson Knowledge Catalog
Help your data citizens easily find, prepare, understand and use the data they need through an enterprise data catalog & governance platform.
Recommended
2 min
News
View all
WebSphere Application Server Support Restatement
IBM Adds Lifecycle Services to Enterprise Networking and Deepens Strategic Partnership with Cisco
All About IBM Storage's Price and Supply Guarantee
Recent support cases
View all
Planned maintenance
View all
IBM Cloud status
View all
Type here to search
Rain
ENG
18:23
12-11-2022

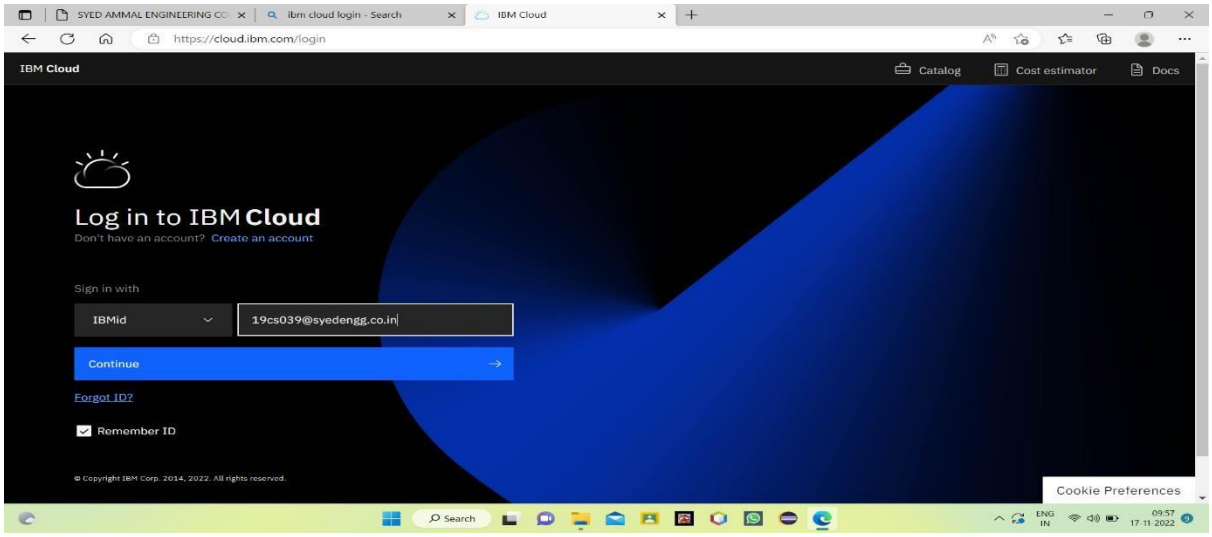

IBM Cloud
cloud.ibm.com
Gmail
YouTube
Maps
IBM Cloud
Search resources and products...
Catalog
Manage
Yamini T. G's Account
Dashboard
Edit dashboard
Upgrade account
Create resource
For you
Select an option
Build
Explore IBM Cloud with this selection of easy starter tutorials and services.
Use Cloudant
Cloudant is a managed NoSQL document database that offers serverless scaling of throughput, capacity, and storage.
Recommended
2 min
Build a web app with Watson Speech to Text
Deploy a conversational interface compatible with any application, device, or channel.
Getting started
15 min
Unlock the entire catalog
Upgrade your account to start using everything IBM Cloud has to offer.
Recommended
10 min
Build a Virtual Private Cloud (VPC)
Upgrade to a paid account to create your your own protected space in the IBM Cloud.
Getting started
7 min
User access
Manage users
Enter email addresses below to jump directly into the
News
View all
WebSphere Application Server Support Restatement
Planned maintenance
Type here to search
23°C Mostly cloudy
4:36 AM
11/12/2022



2.**Train Model on IBM:**

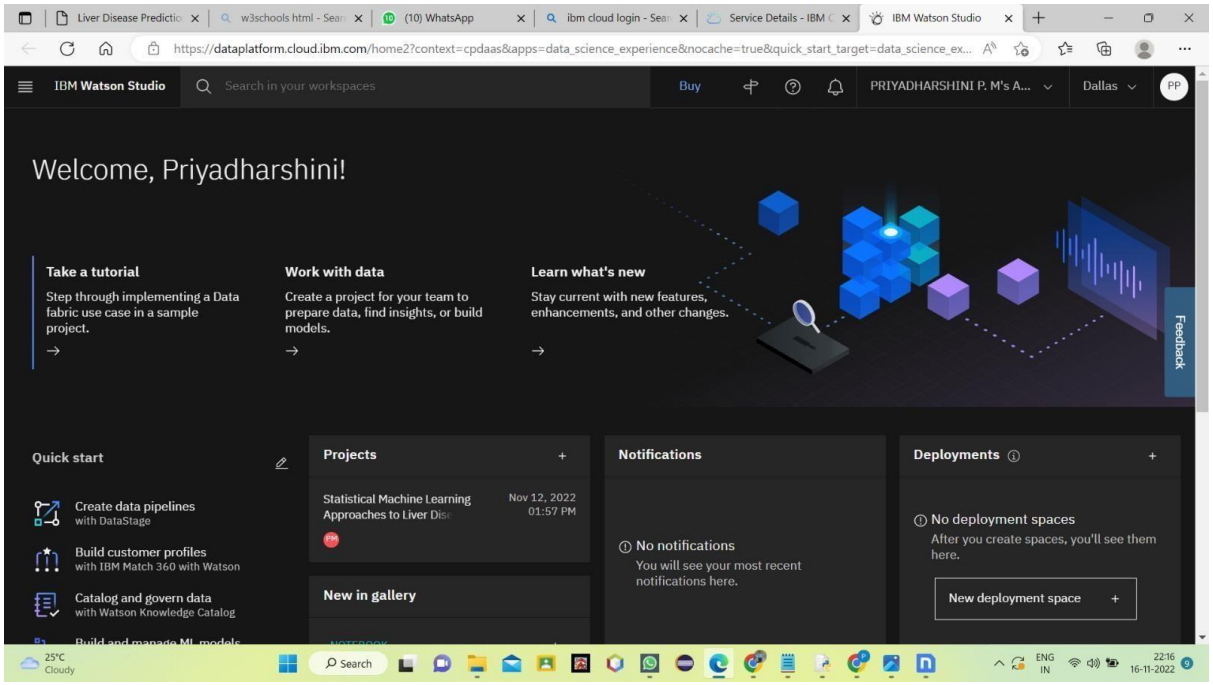
1.Register & Login IBM cloud:

From given link we can Register and afterwards login the IBM cloud using credentials.



2.Opening IBM Watson Studio:

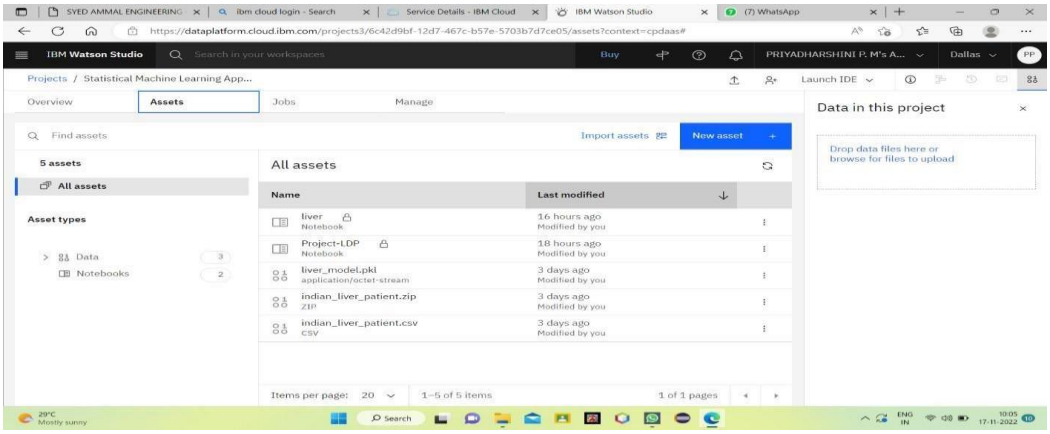
To Run our model we use IBM watson studio inside we will create our own new project.



3.Adding Assests:

Once we create a new project named Liver

Then we will Add some Assets like Python Notebook with extension of .ipynb

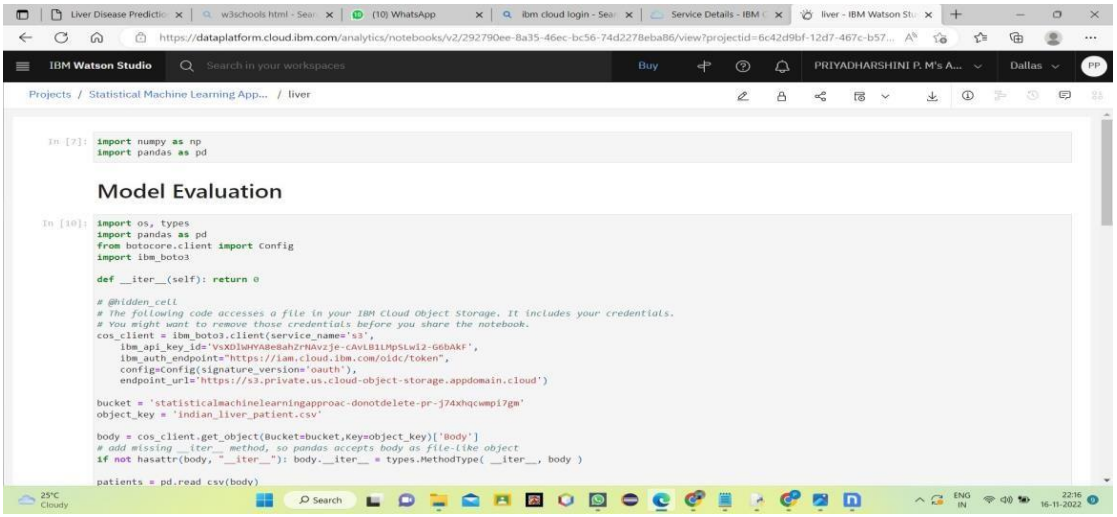


4.Opening notebook file :

At the end we will open the note book file and after we download the dataset once we download it into zip file or we will add the file into data assests.

5.Insert File code :

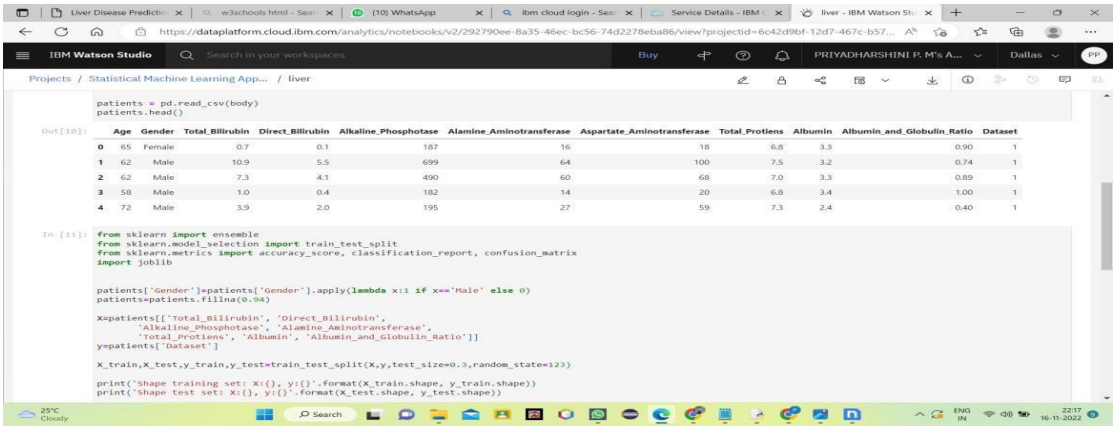
Then we will create a new cell and insert the code into it.



```
In [7]: import numpy as np
        import pandas as pd
```

Model Evaluation

```
In [10]: import os, types
         import pandas as pd
         from botocore.client import Config
         import ibm_boto3

         def __iter__(self): return 0

         # @hidden_cell
         # The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
         # You might want to remove those credentials before you share the notebook.
         cos_client = ibm_boto3.client(service_name='s3',
             ibm_api_key_id='VsXDlWHYA8e8ahZrNAvzje-cAvLB1LMpSLwi2-G6bAkF',
             ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
             config=Config(signature_version='oauth'),
             endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

         bucket = 'statisticalmachinelearningapproac-donotdelete-pr-j74xhqcwmpi7gm'
         object_key = 'indian_liver_patient.csv'

         body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
         # add missing __iter__ method, so pandas accepts body as file-like object
         if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

         patients = pd.read_csv(body)
```



```
In [10]: patients = pd.read_csv(body)
         patients.head()
```

Out[10]:

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

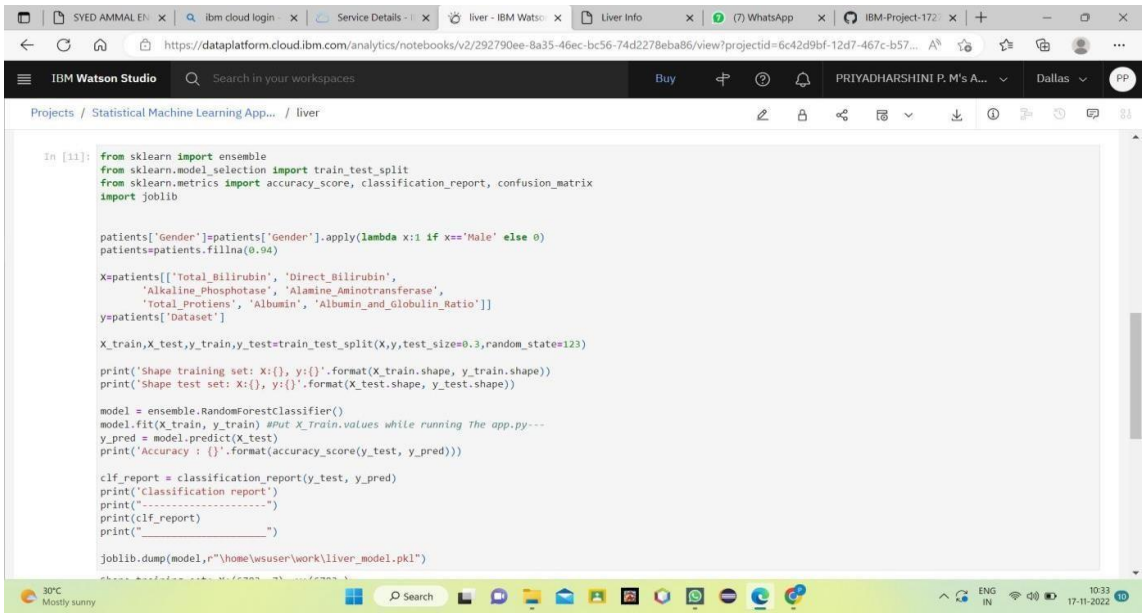
```
In [11]: from sklearn import ensemble
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
         import joblib


         patients['Gender']=patients['Gender'].apply(lambda x:1 if x=='Male' else 0)
         patients=patients.fillna(0.04)

         X=patients[['Total_Bilirubin', 'Direct_Bilirubin',
                 'Alkaline_Phosphotase', 'Alamine_Aminotransferase',
                 'Total_Protiens', 'Albumin', 'Albumin_and_Globulin_Ratio']]
         y=patients['Dataset']

         X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=123)

         print('Shape training set: X:{}, y:{}'.format(X_train.shape, y_train.shape))
         print('Shape test set: X:{}, y:{}'.format(X_test.shape, y_test.shape))
```

Know About the directory by using pwd command.

```
In [1]: pwd

Out[1]: '/home/wsuser/work'
```

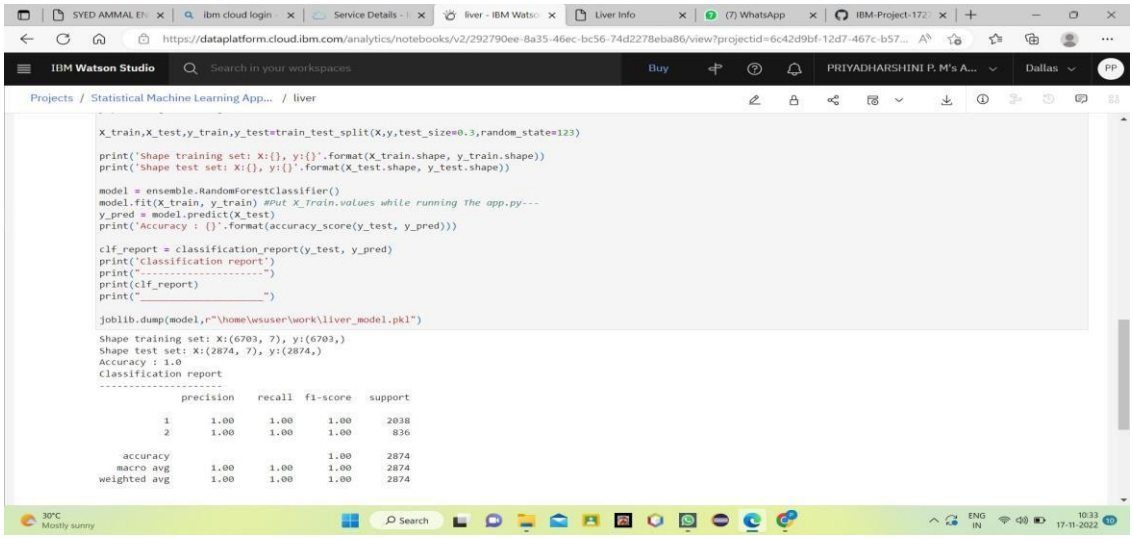
6.Train the Model in IBM Watson Studio:

After Completion of these stuffs we will move forward to start Train the model.



```
In [11]: from sklearn import ensemble
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
         import joblib


         patients['Gender']=patients['Gender'].apply(lambda x:1 if x=='Male' else 0)
         patients=patients.fillna(0.94)

         X=patients[['Total_Bilirubin', 'Direct_Bilirubin',
                'Alkaline_Phosphotase', 'Alamine_Aminotransferase',
                'Total_Protiens', 'Albumin', 'Albumin_and_Globulin_Ratio']]
         y=patients['Dataset']

         X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=123)

         print('Shape training set: X:{}, y:{}'.format(X_train.shape, y_train.shape))
         print('Shape test set: X:{}, y:{}'.format(X_test.shape, y_test.shape))

         model = ensemble.RandomForestClassifier()
         model.fit(X_train, y_train) #Put X_Train.values while running The app.py---
         y_pred = model.predict(X_test)
         print('Accuracy : {}'.format(accuracy_score(y_test, y_pred)))

         clf_report = classification_report(y_test, y_pred)
         print('Classification report')
         print("--------------------")
         print(clf_report)
         print("___________________")

         joblib.dump(model,r"\home\wsuser\work\liver_model.pkl")
```

Outcome of the Trained Model will be shown after compiling and summarizing it.

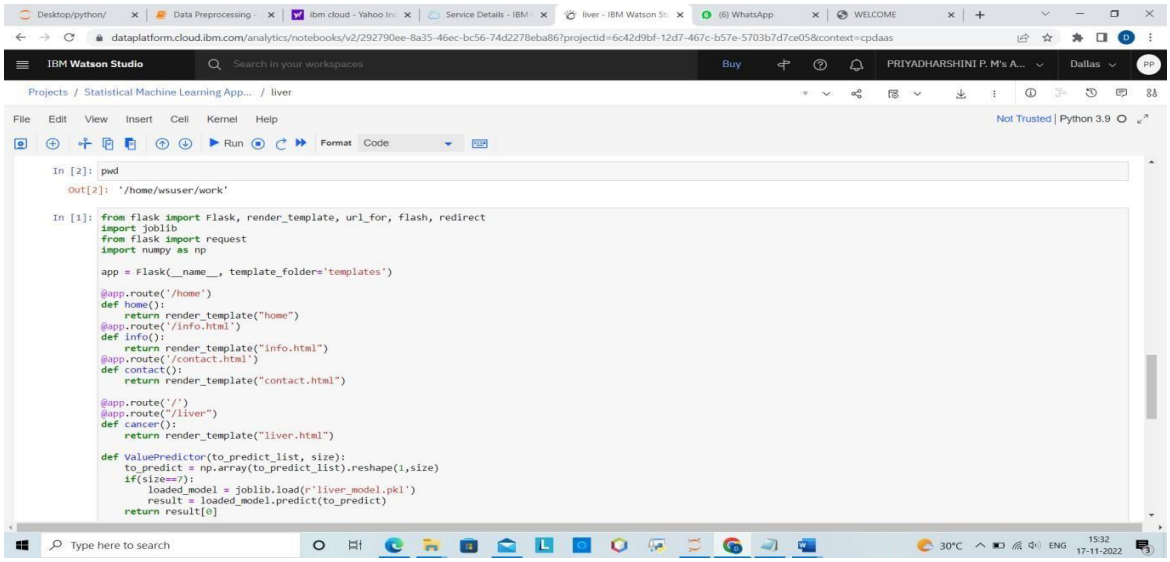


```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=123)

print('Shape training set: X:{}, y:{}'.format(X_train.shape, y_train.shape))
print('Shape test set: X:{}, y:{}'.format(X_test.shape, y_test.shape))

model = ensemble.RandomForestClassifier()
model.fit(X_train, y_train) #Put X_Train.values while running The app.py---
y_pred = model.predict(X_test)
print('Accuracy : {}'.format(accuracy_score(y_test, y_pred)))

clf_report = classification_report(y_test, y_pred)
print('Classification report')
print("--------------------")
print(clf_report)
print("___________________")

joblib.dump(model,r"\home\wsuser\work\liver_model.pkl")
Shape training set: X:(6703, 7), y:(6703,)
Shape test set: X:(2874, 7), y:(2874,)
Accuracy : 1.0
Classification report
--------------------
              precision    recall  f1-score   support

           1       1.00      1.00      1.00      2038
           2       1.00      1.00      1.00       836

    accuracy                           1.00      2874
   macro avg       1.00      1.00      1.00      2874
weighted avg       1.00      1.00      1.00      2874
```

This how the Model was Trained in IBM watson Studio.

7.Testing the Model:

At the end of the day we will try to test the model which can give the desired and precise prediction based on the trained model.



This How the Model was tested in IBM Watson Stud

3.Integrate Flask With Scoring End Point:

Abstract:

One of the most vital causes of death worldwide is liver disease. We, humans, have come a long way in the medical field and scientific advancements to treat diseases and it's evident that when these liver diseases are detected early, they can be treated easily. In order to be able to accurately predict if there's a chance of the liver disease it is imperative to identify the features/symptoms which play a significant role in causing the Liver Disease. In order to improve the performance of the prediction models, it is important to choose the right combination of features.

Keywords:

Machine Learning; Deep Learning; Neural networks; classification techniques

Methodologies:

There are several steps involved in the process of classification of liver disease. Since we are using machine learning models for the purpose of classification, the steps involved in the process:

1.Datacollection

2.Datapreparation

3.Chooseamodel

4.Trainthemodel

5.Evaluatethemodel
6.ParameterTuning

7.formulatepredictions

PROPOSEDMODEL:

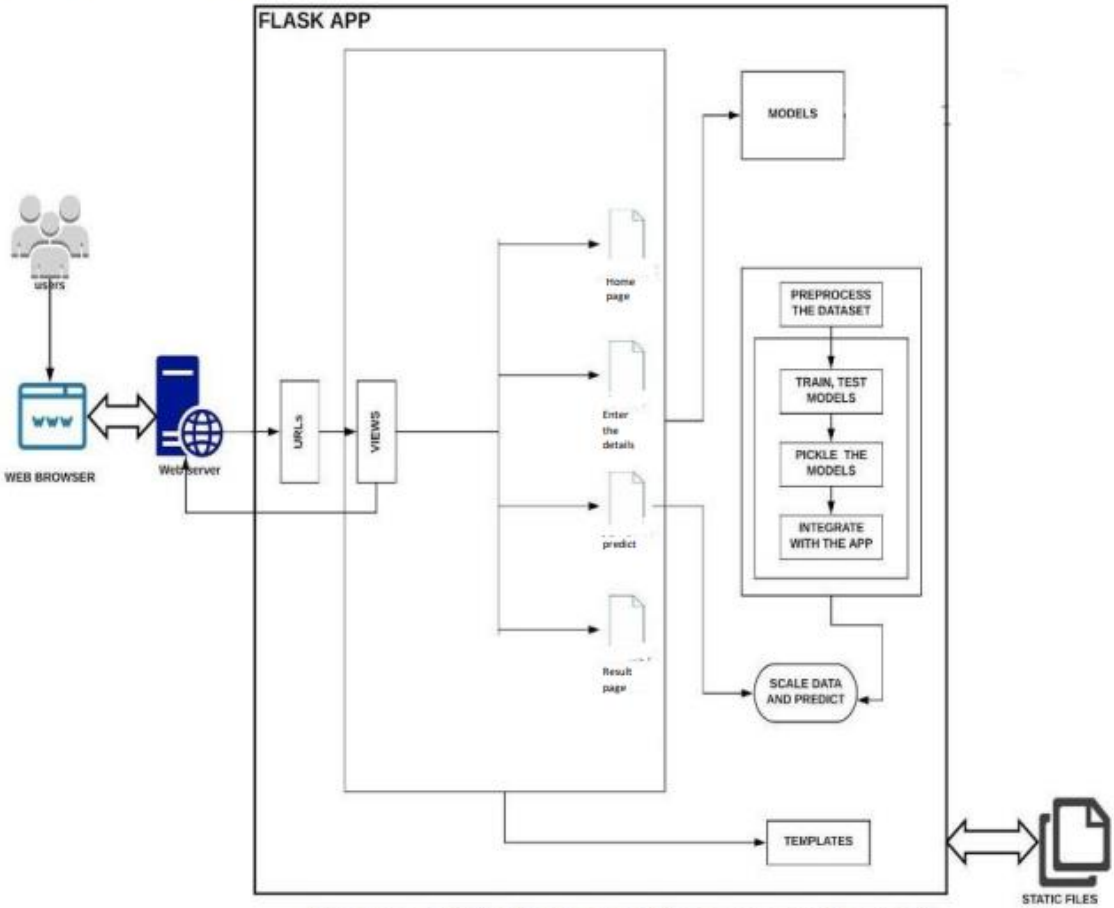


Figure-1: Work flow of the proposed model

PHASES

Ideation Phase:

In this milestone you are expected to get started with the Ideation process.

Literature Survey On The Selected Project & Information Gathering:

In this activity you are expected to gather/collect the relevant information on project usecase, refer the existing solutions, technical papers, research publications etc.

Prepare Empathy Map :

In this activity you are expected to prepare the empathy map canvas to capture the user Pains & Gains, Prepare list of problem statements.

Ideation:

In this activity you are expected to list the ideas (atleast 4 per each team member) by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.

Project Design Phase-I:

From this milestone you will be starting the project design phase. You are expected to cover the activities given.

Proposed Solution:

In this activity you are expected to prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.

Problem Solution Fit:

In this activity you are expected to prepare problem - solution fit document and submit for review.

Solution Architecture:

In this activity you are expected to prepare solution architecture document and submit for review.

Project Design Phase-II:

From this milestone you will be continue working on the project design phase. You are expected to cover the activities given.

Customer Journey:

Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).

Functional Requirement :

In this activity you are expected to prepare the functional requirement document.

Data Flow Diagrams:

In this activity you are expected to prepare the data flow diagrams and submit for review.

Technology Architecture:

In this activity you are expected to draw the technology architecture diagram.

Project Planning Phase:

In this milestone you are expected to prepare milestones & tasks, sprint schedules.

Prepare Milestone & Activity List :

In this activity you are expected to prepare the milestones & activity list of the project.

Sprint Delivery Plan:

In this activity you are expected to prepare the sprint delivery plan.

Project Development Phase:

In this milestone you will start the project development and expected to perform the coding & solutioning, acceptance testing, performance testing based as per the sprint and submit them.

Project Development - Delivery Of Sprint-1:

In this activity you are expected to develop & submit the developed code by testing it.

Project Development - Delivery Of Sprint-2:

In this activity you are expected to develop & submit the developed code by testing it.

Project Development - Delivery Of Sprint-3:

In this activity you are expected to develop & submit the developed code by testing it.

Project Development - Delivery Of Sprint-4:

In this activity you are expected to develop & submit the developed code by testing it.

Coding:

App.py:

```
from flask import Flask, render_template, url_for, flash, redirect
import joblib
from flask import request
import numpy as np
app = Flask(__name__, template_folder='templates')
@app.route('/home')
def home():
    return render_template("home")
@app.route('/info.html')
def info():
    return render_template("Liver-info.html")
@app.route('/contact.html')
def contact():
    return render_template("contact.html")
@app.route('/')
@app.route("/liver")
def cancer():
    return render_template("liver.html")
def ValuePredictor(to_predict_list, size):
    to_predict = np.array(to_predict_list).reshape(1,size)
    if(size==7):
        loaded_model = joblib.load(r'liver_model.pkl')
        result = loaded_model.predict(to_predict)
    return result[0]
@app.route('/predict', methods = ["POST"])
def predict():
    if request.method == "POST":
        to_predict_list = request.form.to_dict()
```

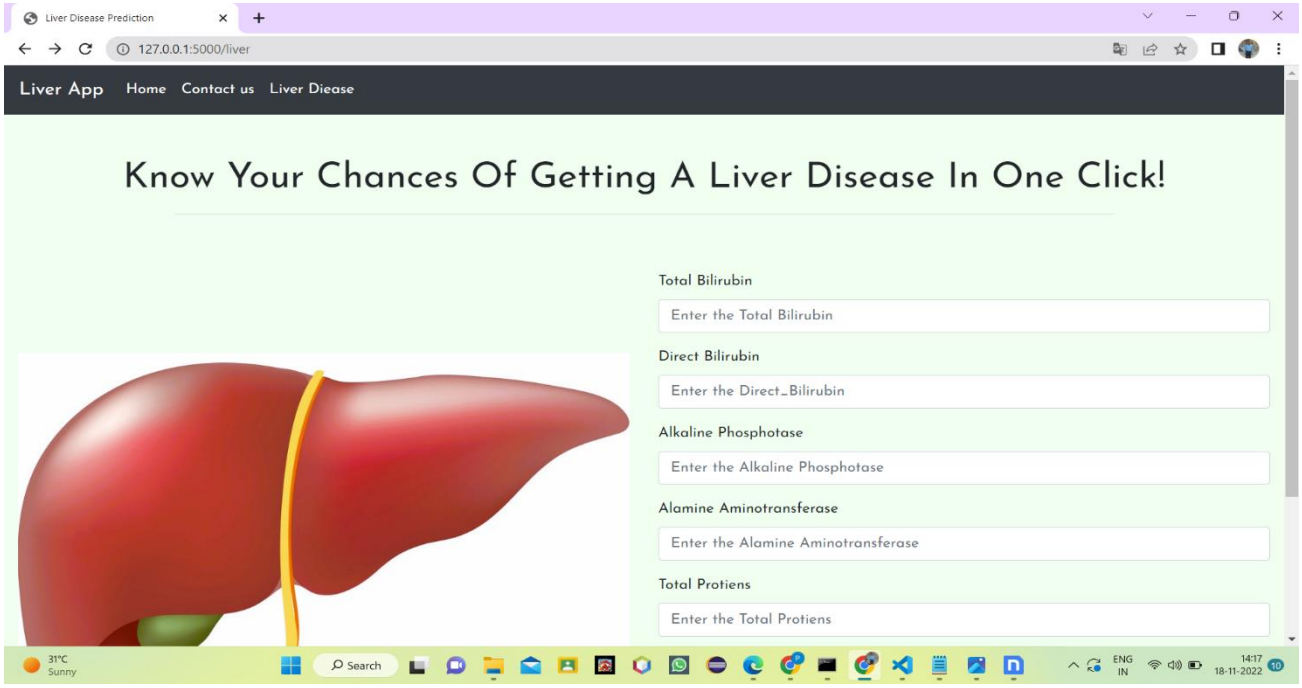
```
 to_predict_list = list(to_predict_list.values())
 to_predict_list = list(map(float, to_predict_list))
        #liver
       if(len(to_predict_list)==7):
           result = ValuePredictor(to_predict_list,7)
   print("predicting...")
   if(int(result)==1):
       prediction = "Sorry, you have chances of getting the disease. Please consult the doctor immediately"
       print("Prediction done")
   else:
       prediction = "No need to fear. You have no dangerous symptoms of the disease"
       print("Prediction done")
   return(render_template("result.html", prediction_text=prediction))
    if __name__ == "__main__":
   app.run(debug=True)
   from waitress import serve
   serve(app, host="0.0.0.0", port=8087)
```

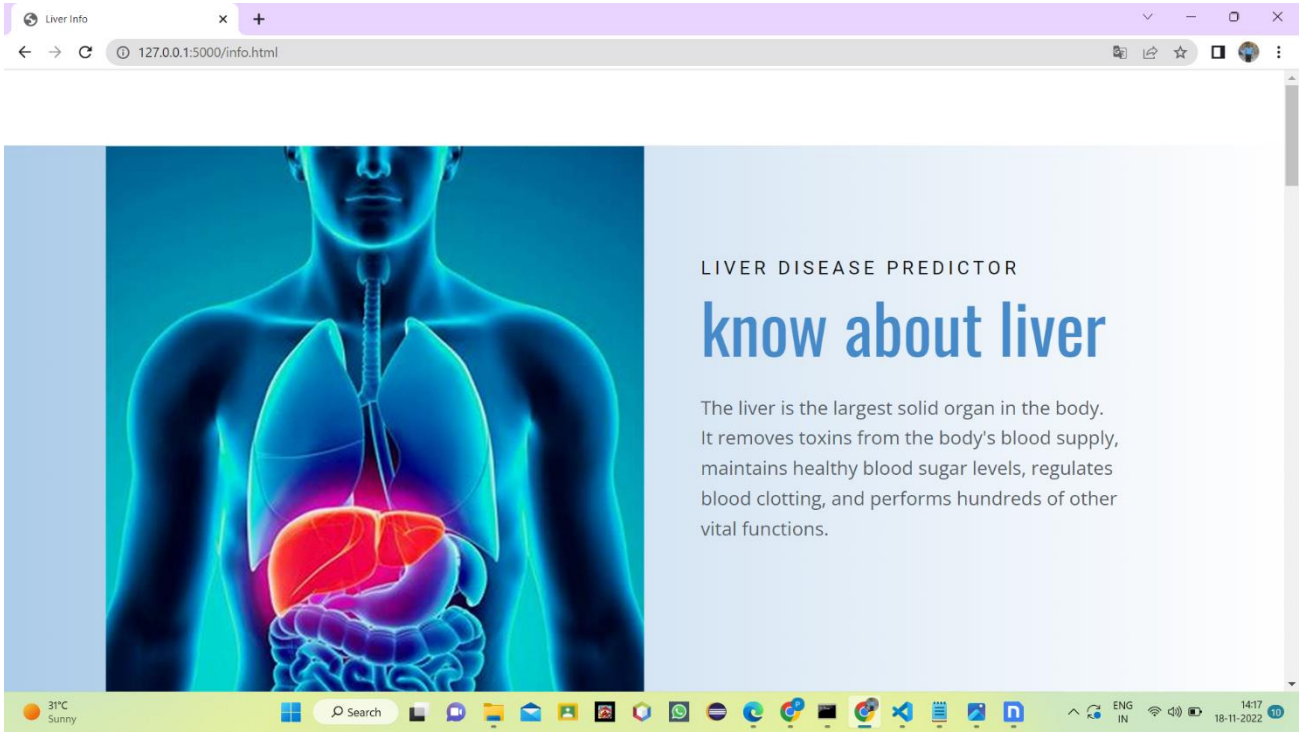
Liver.py:

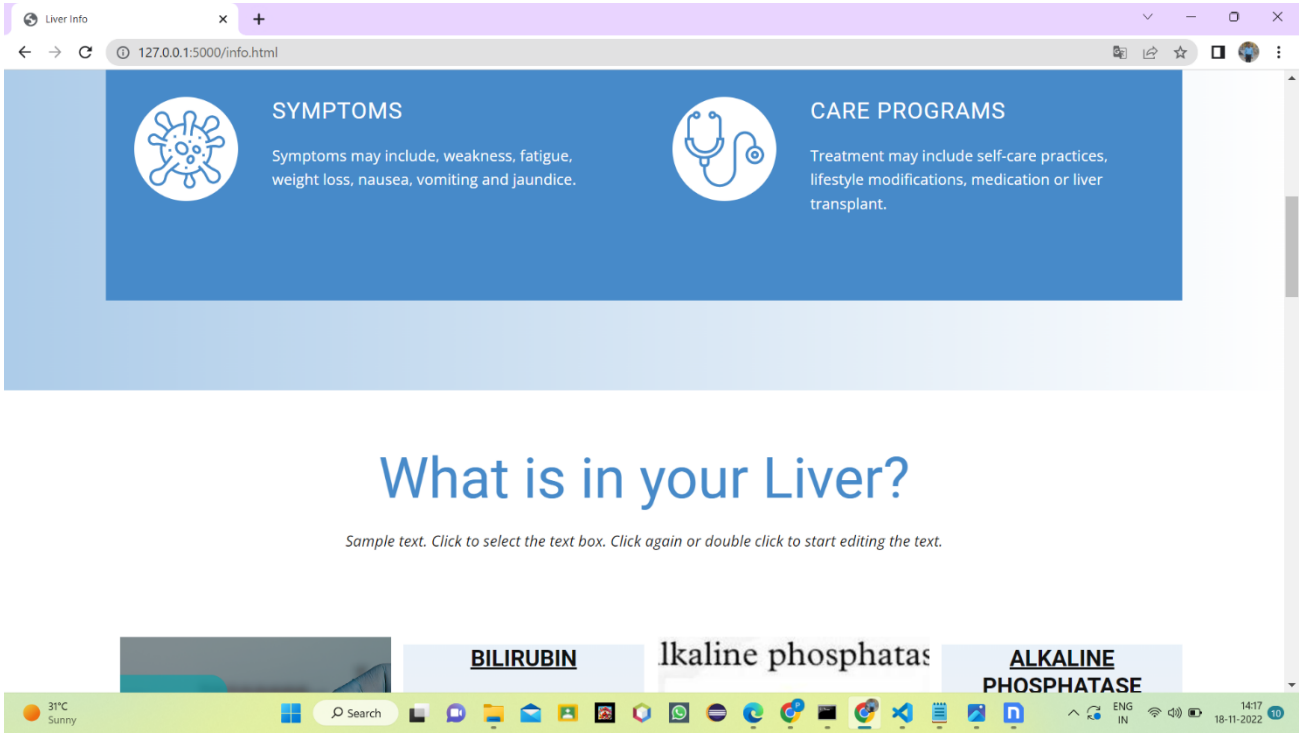
```
import numpy as np
import pandas as pd
from sklearn import ensemble
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import joblib
patients=pd.read_csv(r'E:\MyProject\indian_liver_patient.csv')
patients['Gender']=patients['Gender'].apply(lambda x:1 if x=='Male' else 0)
patients=patients.fillna(0.94)
X=patients[['Total_Bilirubin', 'Direct_Bilirubin',
       'Alkaline_Phosphotase', 'Alamine_Aminotransferase',
```

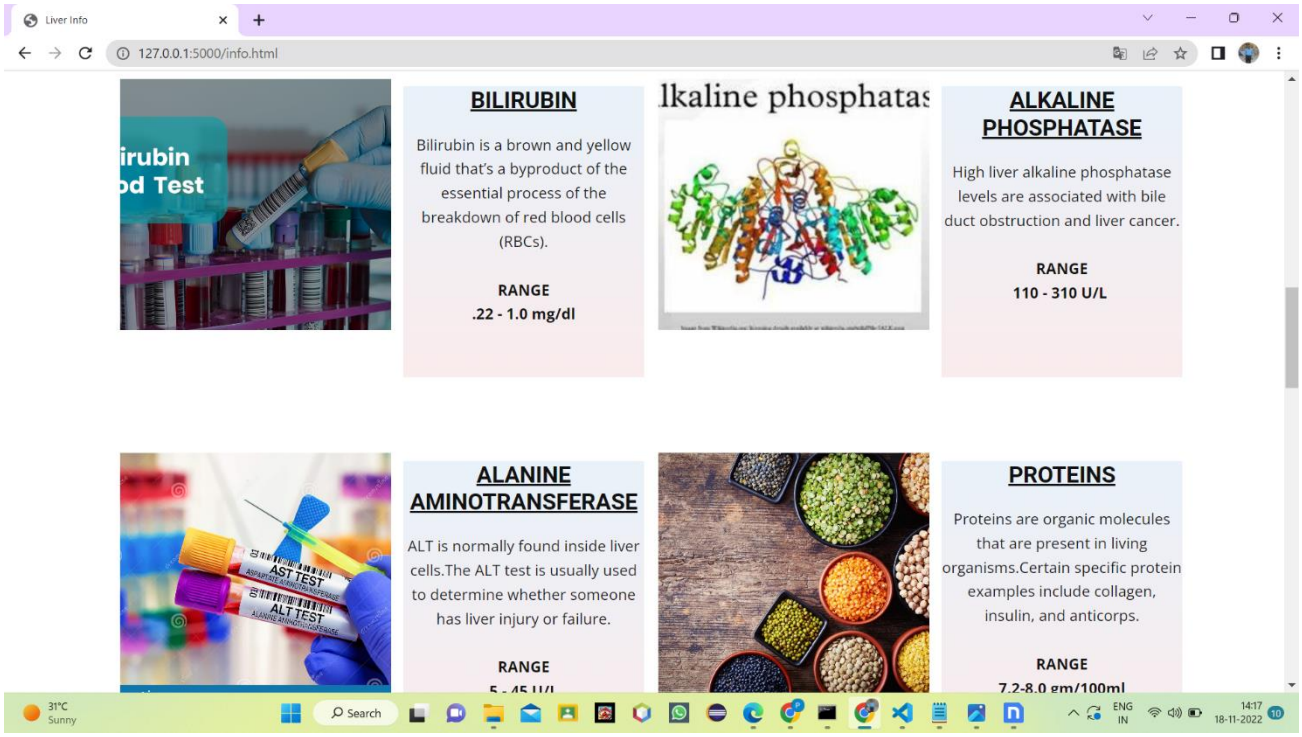
```
    'Total_Protiens', 'Albumin', 'Albumin_and_Globulin_Ratio']]
y=patients['Dataset']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=123)
print('Shape training set: X:{}, y:{}'.format(X_train.shape, y_train.shape))
print('Shape test set: X:{}, y:{}'.format(X_test.shape, y_test.shape))
model = ensemble.RandomForestClassifier()
model.fit(X_train.values, y_train.values)
y_pred = model.predict(X_test)
print('Accuracy : {}'.format(accuracy_score(y_test, y_pred)))
clf_report = classification_report(y_test, y_pred)
print('Classification report')
print("--------------------")
print(clf_report)
print("_____________________")
joblib.dump(model,r"E:\MyProject\liver-disease-prediction-main\liver_model.pkl")
```

Outputs Screenshots:

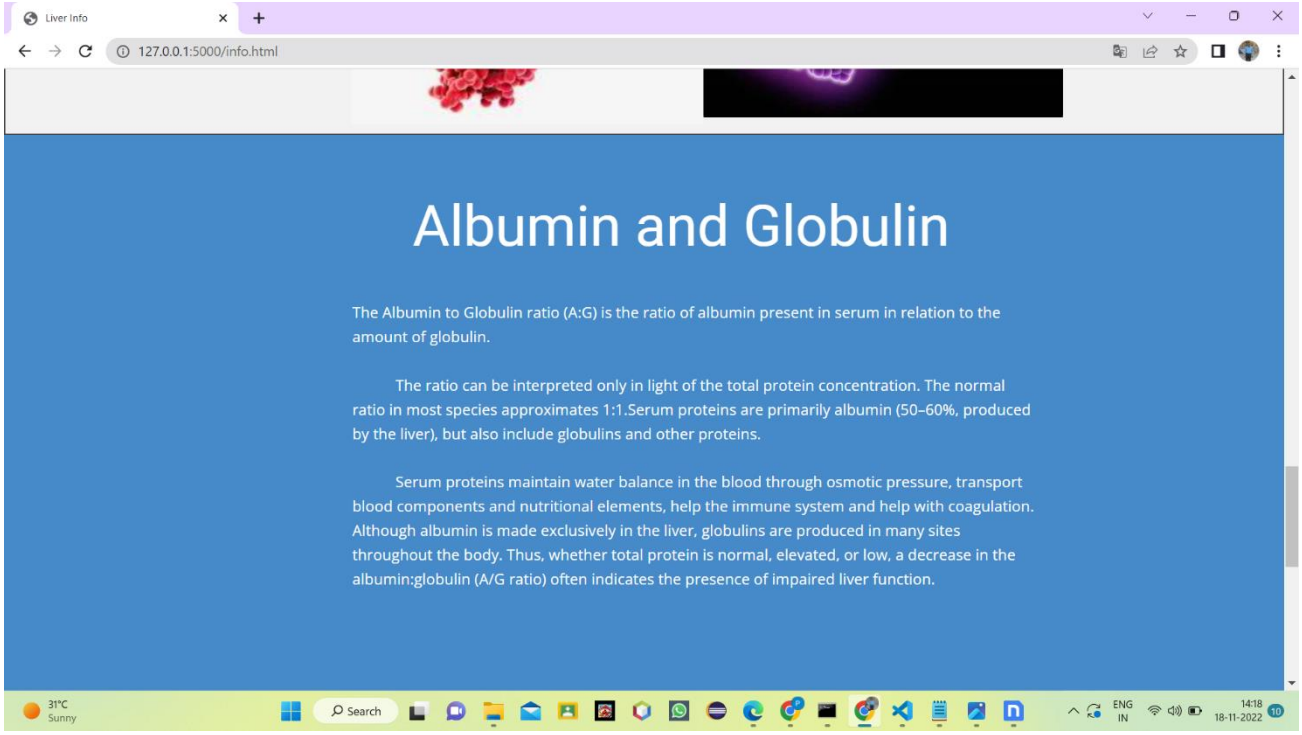


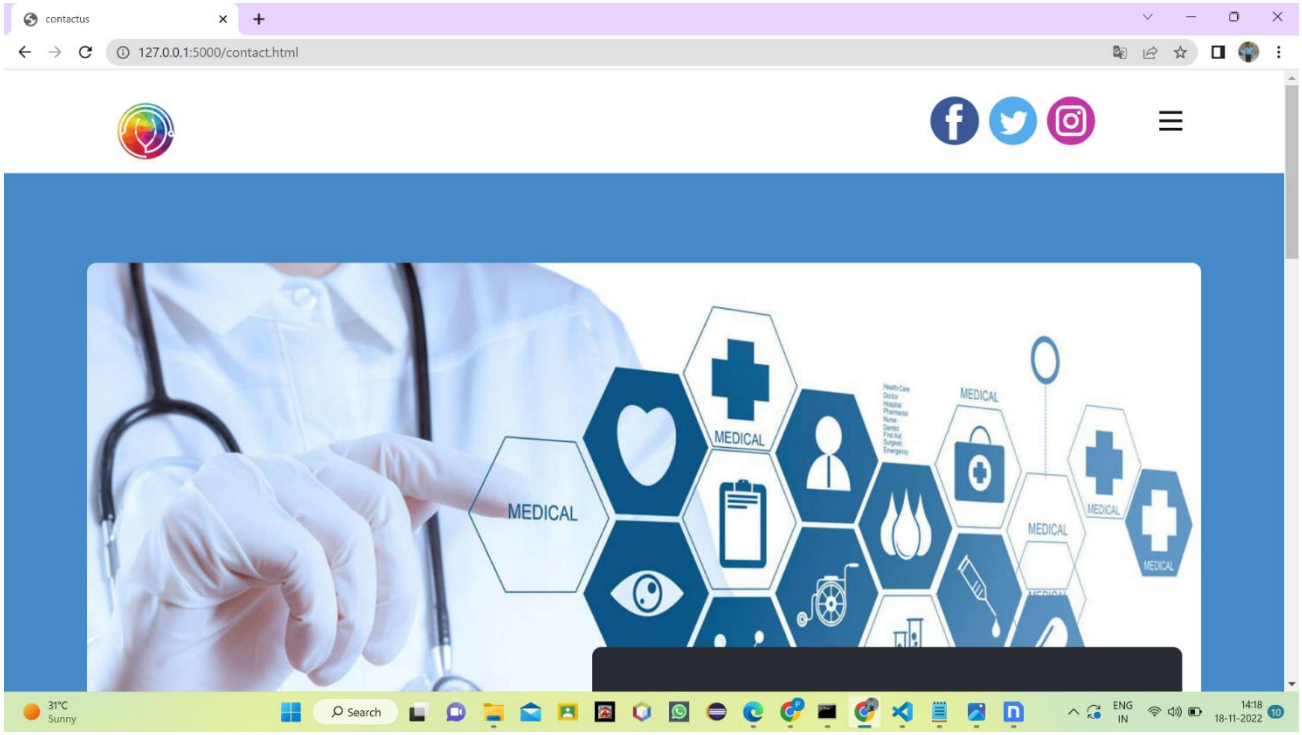

Liver Info
127.0.0.1:5000/info.html
LIVER DISEASE PREDICTOR
know about liver
The liver is the largest solid organ in the body. It removes toxins from the body's blood supply, maintains healthy blood sugar levels, regulates blood clotting, and performs hundreds of other vital functions.

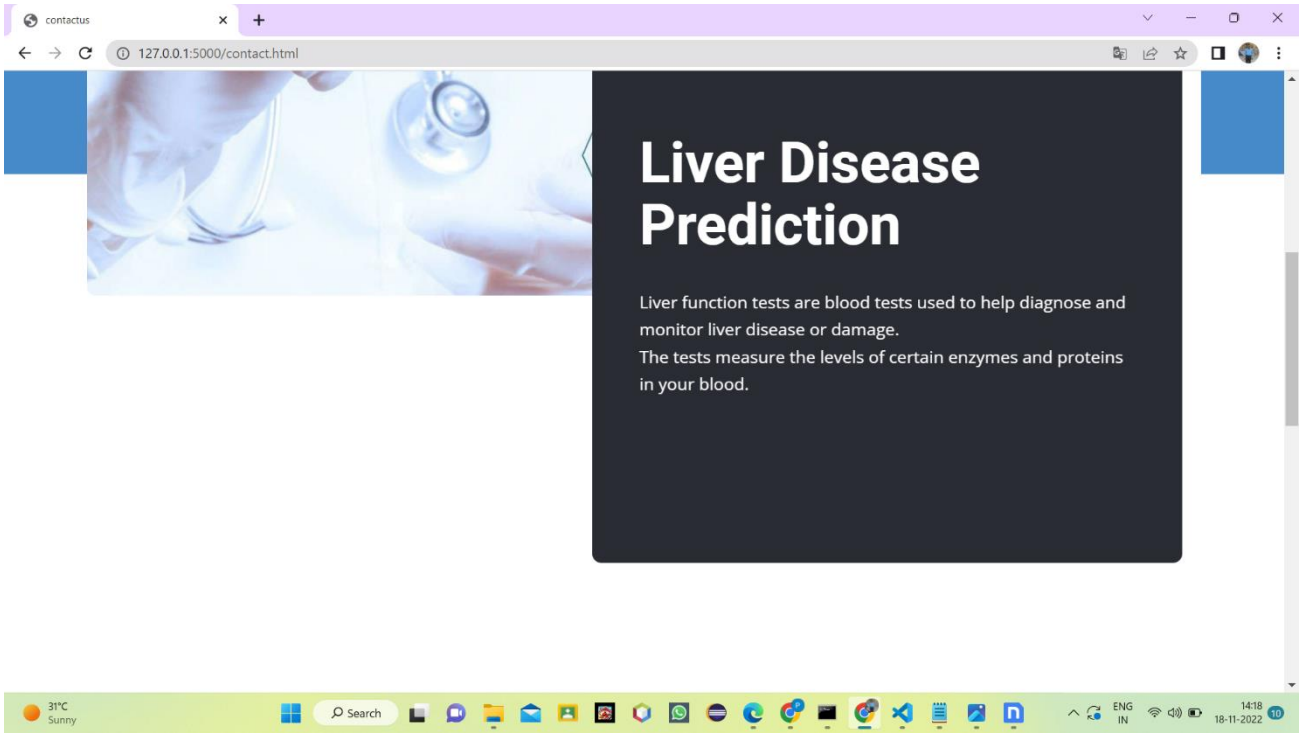

Liver Info
127.0.0.1:5000/info.html
SYMPTOMS
Symptoms may include, weakness, fatigue, weight loss, nausea, vomiting and jaundice.
CARE PROGRAMS
Treatment may include self-care practices, lifestyle modifications, medication or liver transplant.
What is in your Liver?
Sample text. Click to select the text box. Click again or double click to start editing the text.
BILIRUBIN
lkaline phosphatas
ALKALINE PHOSPHATASE

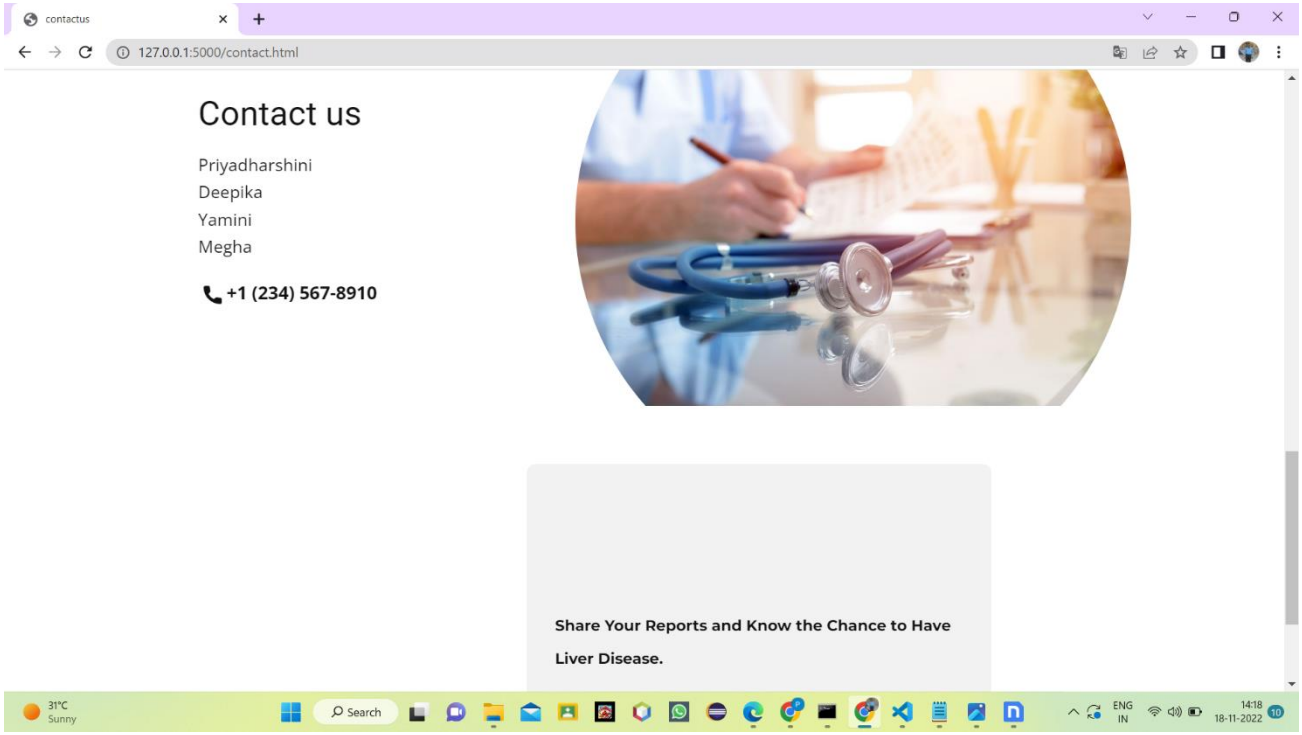

Liver Info
127.0.0.1:5000/info.html
irubin
od Test
BILIRUBIN
Bilirubin is a brown and yellow fluid that's a byproduct of the essential process of the breakdown of red blood cells (RBCs).
RANGE
.22 - 1.0 mg/dl
lkaline phosphatas
ALKALINE PHOSPHATASE
High liver alkaline phosphatase levels are associated with bile duct obstruction and liver cancer.
RANGE
110 - 310 U/L
AST TEST
ALT TEST
ALANINE AMINOTRANSFERASE
ALT is normally found inside liver cells.The ALT test is usually used to determine whether someone has liver injury or failure.
RANGE
PROTEINS
Proteins are organic molecules that are present in living organisms.Certain specific protein examples include collagen, insulin, and anticorps.
RANGE
7.2-8.0 gm/100ml
31°C
Sunny
Search
ENG
IN
14:17
18-11-2022

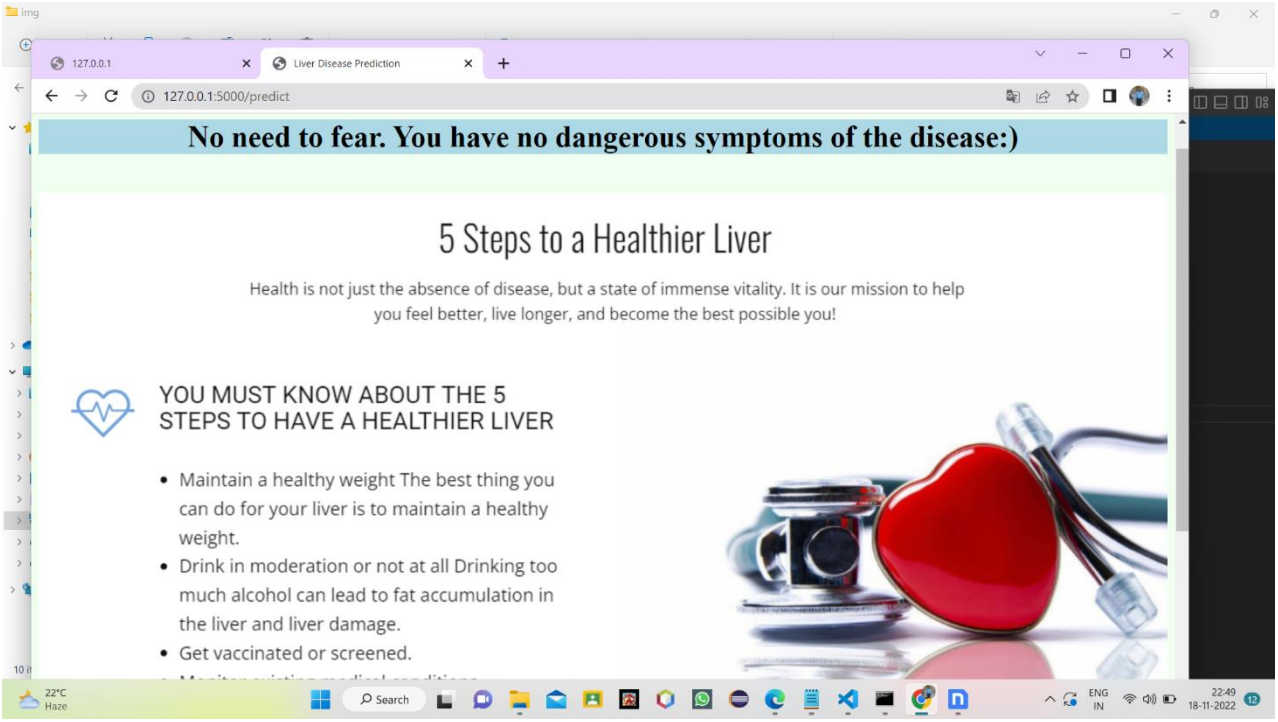

Liver Info
127.0.0.1:5000/info.html
Albumin and Globulin
The Albumin to Globulin ratio (A:G) is the ratio of albumin present in serum in relation to the
31°C
Sunny
Search
ENG
IN
14:17
18-11-2022

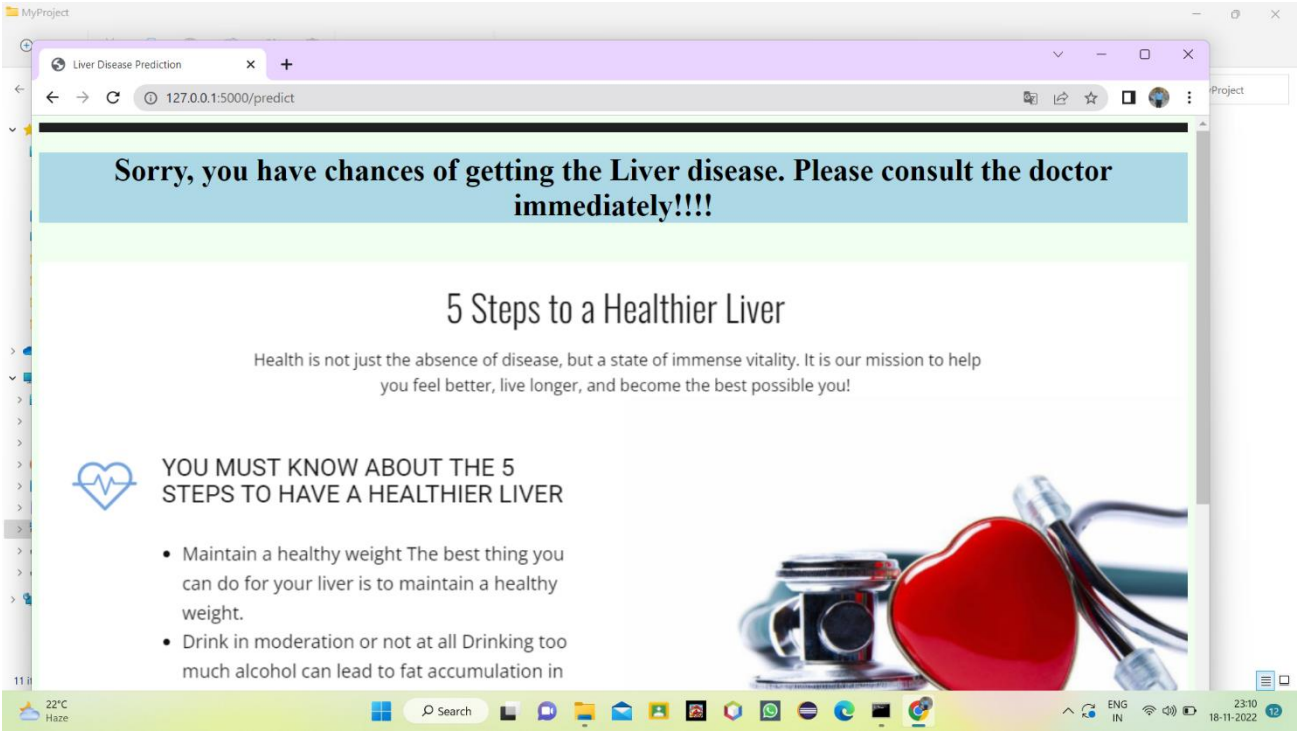

Liver Info
127.0.0.1:5000/info.html
Albumin and Globulin
The Albumin to Globulin ratio (A:G) is the ratio of albumin present in serum in relation to the amount of globulin.
The ratio can be interpreted only in light of the total protein concentration. The normal ratio in most species approximates 1:1.Serum proteins are primarily albumin (50–60%, produced by the liver), but also include globulins and other proteins.
Serum proteins maintain water balance in the blood through osmotic pressure, transport blood components and nutritional elements, help the immune system and help with coagulation. Although albumin is made exclusively in the liver, globulins are produced in many sites throughout the body. Thus, whether total protein is normal, elevated, or low, a decrease in the albumin:globulin (A/G ratio) often indicates the presence of impaired liver function.
31°C
Sunny
Search
ENG
IN
14:18
18-11-2022


contactus
127.0.0.1:5000/contact.html
MEDICAL
31°C
Sunny
Search
ENG
IN
14:18
18-11-2022


contactus
127.0.0.1:5000/contact.html
Liver Disease Prediction
Liver function tests are blood tests used to help diagnose and monitor liver disease or damage.
The tests measure the levels of certain enzymes and proteins in your blood.


contactus
127.0.0.1:5000/contact.html
Contact us
Priyadharshini
Deepika
Yamini
Megha
+1 (234) 567-8910
Share Your Reports and Know the Chance to Have Liver Disease.





Done by

Priyadharshini & team