Liver Patient Analysis

Using IBM Machine

—

Your Name
Sunkar Sai Revanth

Pre Requisites:

1. In order to develop this project we need to install the 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 very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using a Jupyter notebook and Spyder To install the Anaconda navigator and to know how to use Jupyter Notebook & Spyder using Anaconda watch the video https://youtu.be/5mDYijMfSzs Python packages: NumPy: NumPy is a Python package that stands for 'Numerical Python. It is the core library for scientific computing, which contains a powerful n-dimensional array of objects. Pandas: pandas is a fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation tool

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.

1 . INTRODUCTION

1.1 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:

1. A patient going to a doctor with certain symptoms.

2. The doctor recommending certain tests like blood test, urine test etc depending on the symptoms.

3. The patient taking the aforementioned tests in an analysis lab.

4. 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#39;s from various

features a machine learning algorithm that will be trained to predict liver disease in patients.

1.2 Purpose

The goal of this project is to reduce some of the delays caused by unnecessary detours between the hospital and the pathology laboratory. Historically, work has been done to detect the onset of heart disease, such as Parkinson’s, and machine learning algorithms have been developed to predict liver disease in patients based on a variety of characteristics.

2 . LITERATURE SURVEY

2.1 Existing Problem

The problem statement is formally defined as:

‘Given a dataset containing various attributes of 584 Indian patients, use the features available in the dataset and define a supervised classification algorithm which can identify whether a person is suffering from liver disease or not. This data set contains 416 liver patient records and 167 non- liver patient records.The data set was collected from north east of Andhra Pradesh, India. This data set contains 441 male patient records and 142 female patient records. Any patient whose age exceeded 89 is listed as being of age "90".

Strategy

This seems to be a classic example of supervised learning. We have been provided with a fixed number of features for each data point, and our aim will be to train a variety of Supervised Learning algorithms on this data, so that , when a new data point arises, our best performing classifier can be used to categorize the data point as a positive example or negative. Exact details of the number and types of algorithms used for training is included in the 'Algorithms and Techniques#39; sub-section of the 'Analysis#39; part.

2.2 Proposed Solution

Project Workflow

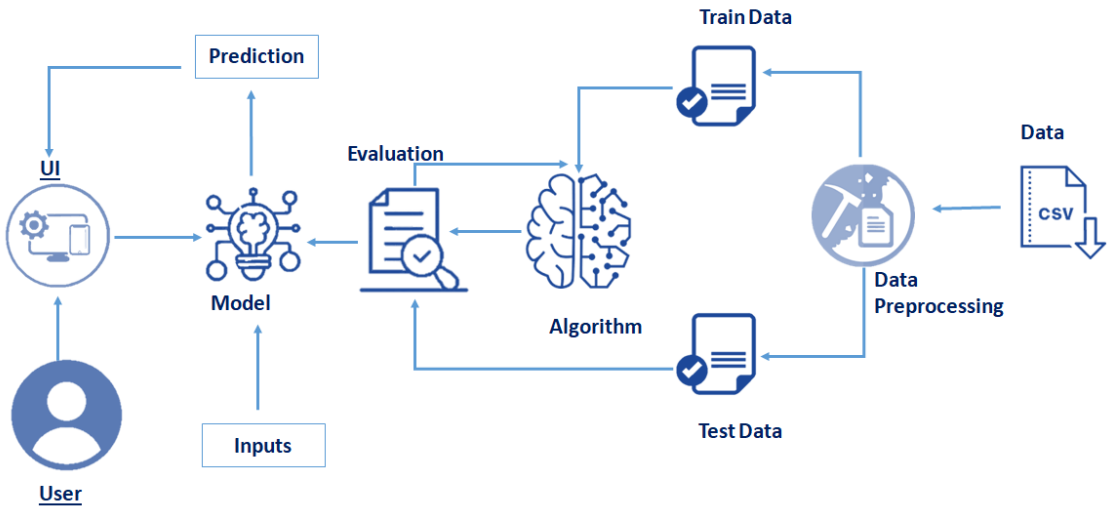
- User interacts with the UI (User Interface) to upload the input features.
- Uploaded features/input is analyzed by the model which is integrated.

- 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

- Data Collection.
  - Collect the dataset or Create the dataset
- Data Preprocessing.
  - Import the Libraries.
  - Importing the dataset.
  - Analyze the data
  - Taking care of Missing Data
  - Data Visualization
  - Splitting Data into Train and Test.
- Model Building
  - Import the model building Libraries
  - Initializing the model
  - Adding Machine Learning Model
  - Training the model
  - Model Evaluation
  - Save the Model
- Application Building
  - Create an HTML file
  - Build Python Code
  -

Flow CHART :



Dataset Collection :

I downloaded data set from the following link

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

Dataset Preprocessing :

Import the Libraries

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
```

Reading the Dataset

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

Exploratory Data Analysis

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

Checking & Handling Of Null Data

This isnull().any() method returns 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
```

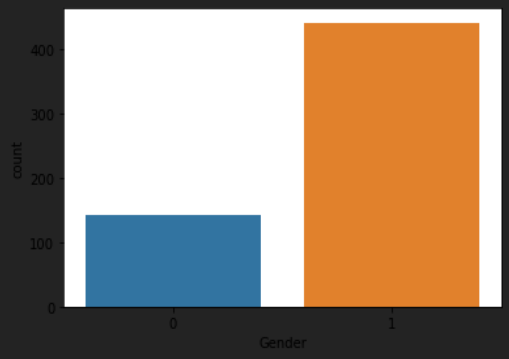
Data Visualization

1. Bar plot between Gender and Count number of diseases.

2. Bar Plot which describes the total

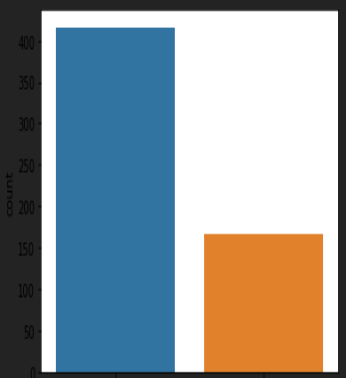
```
# Counting patients who are Male and who are Female
sns.countplot(data=data, x = 'Gender', label='Count')
m,f=data['Gender'].value_counts()
print("No of Males:",m)
print("No of Females:",f)
```

```
No of Males: 441
No of Females: 142
```




```
# Counting patients who are diagnosed and not diagnoised with liver disease
sns.countplot(data=data, x = 'Dataset')
LD,NLD=data['Dataset'].value_counts()
print("liver disease patinets:",LD)
print("Non-liver disease patinets:",NLD)
```

```
liver disease patinets: 416
Non-liver disease patinets: 167
```



Splitting The Dataset Into Dependent And Independent Variable

```
# dividing the data into input and output
x=data.iloc[:,0:-1]
y=data.iloc[:,-1]
```

Split The Dependent And Independent Features Into Train Set And Test Set

```
# importing the train_test_split from scikit-learn
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
```

Model Building:

Train & Test the Model Using Classification Algorithm

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

You will need to train the datasets to run smoothly and see an incremental improvement in the prediction rate.

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()
```

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 compared to KNN and Random forest algorithms, we are proceeding with the support vector machine model.

Save The Model

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.

Application Building :

Build Python Code

Let us build a flask file 'Liver_Flask_App.py' which is a web framework written in python for server-side scripting.

Let's see the step by step procedure for building the backend application.

- App starts running when the "__name__" constructor is called in main.
- render_template is used to return an html file.
- "GET" method is used to take input from the user.
- "POST" method is used to display the output to the user.

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 then 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('/') # 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")
```

Firstly, we are rendering the home.html template and from there we are navigating to our prediction page that is upload.html. We enter input values here and these values are sent to the loaded model and the resultant output is displayed on 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()
```

Here the route for prediction is given and necessary steps are performed in order to get the predicted output.

Build the HTML page and Execute:

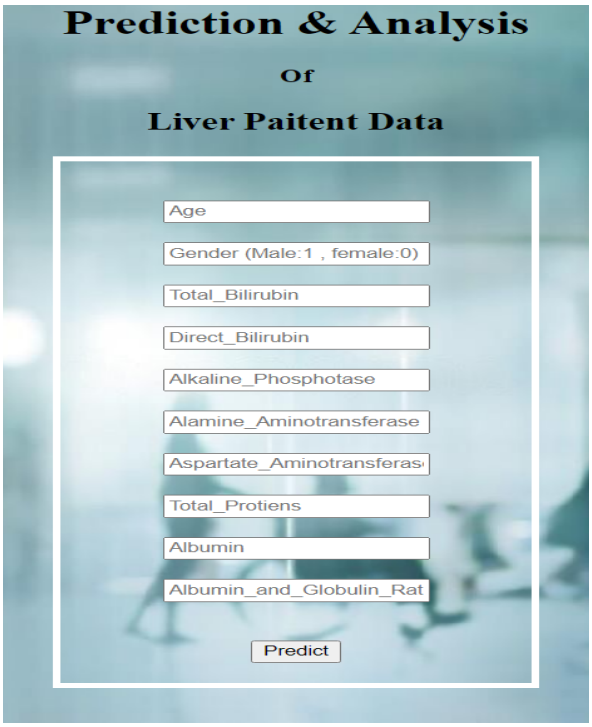
To build the HTML page and execute.we have to follow the below steps

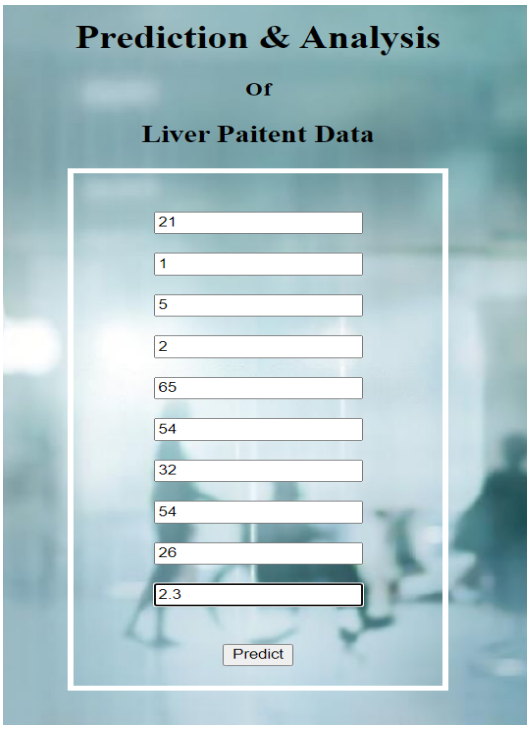
Step 1: Run the application In the anaconda prompt, navigate to the folder in which the flask app is present. When the python file is executed the localhost is activated on port 5000 and can be accessed through it.

```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)
```

These are the Home page:





IBM Watson:

For Account Creation,follow the given link :

https://youtu.be/QuTDhYeJh0k

Training model on IBM watson:

https://youtu.be/BzouqMGJ41k

I have done local deployment from the jupyter note book.And modify the code In IBM watson studio.I got scoring end point and python code by modifying the code in IBM watson studio.After that I have converted the app from local deployment to public deployment.Then everyone will use it to classify the ships.

Advantages & Disadvantages :

Advantages

- Good reproducibility
- High applicability (95%)
- Can be performed in the outpatient clinic
- No cost and wide availability
- Prognostic value of some has been validated for some a etiologies of chronic liver disease on population

Disadvantages

- Performance not as good as TE and patented serum markers
- False positive results with FIB-4 and NFS in case of age > 65 yrs

Conclusion & Future Scope :

Diseases related to the liver and heart are becoming more and more common with time. With continuous technological advancements, these are only going to increase in the future. Although people are becoming more conscious of health nowadays and are joining yoga classes, dance classes; still the sedentary lifestyle and luxuries that are continuously being introduced and enhanced; the problem is going to last long.

So, in such a scenario, our project will be extremely helpful to society. With the dataset that we used for this project, we got 100 % accuracy for SVM model, and though it might be difficult to get such accuracies with very large datasets, from this projects results, one can clearly conclude that we can predict the risk of liver diseases with accuracy of 90 % or more.

Today almost everybody above the age of 12 years has smartphones with them, and so we can incorporate these solutions into an android app or ios app. Also it can be incorporated into a website and these app and websites will be highly beneficial for a large section of society.

Bibliography :

1. Biomarkers for prediction of liver fibrosis in patients with chronic alcoholic liver disease written by Sylvie Naveau and Bruno Runyard.

2. Strategic analysis in prediction of liver disease using classification algorithms written by Piyush Kr Shukla andBinish Khan.

3. Software based prediction of liver disease with feature selection and classification techniques witten Jagdeep Singh, Sandeep Bagga and Ranjodh Kaur.

4. Liver disease prediction using SVM and NaÃ¯ve Bayes algorithm written by S Dhayanand.

5. Prediction and analysis of liver diseases using data mining written Shambel Kefelgen, Pooja Kamat