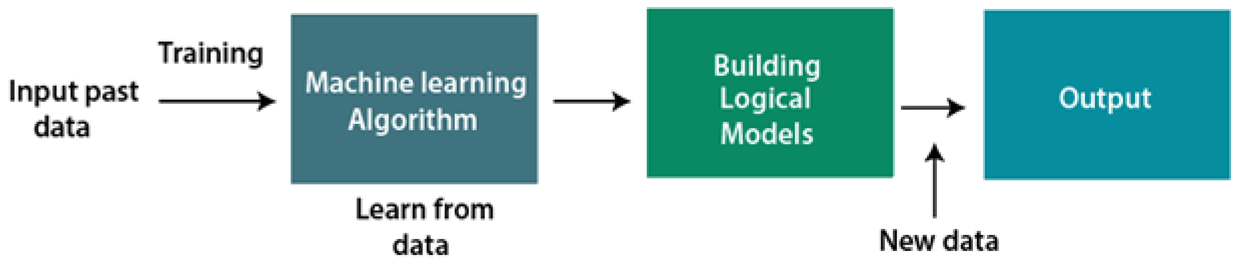
A REVIEW OF LIVER PATIENT ANALYSIS METHODS USING MACHINE LEARNING

Overview:

Machine Learning is said as a subset of **artificial intelligence** that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by **Arthur Samuel** in **1959**.Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for **building mathematical models and making predictions using historical data or information**. Currently, it is being used for various tasks such as **image recognition**, **speech recognition**, **email filtering**, **Facebook auto-tagging**, **recommender system**, and many more. Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed.



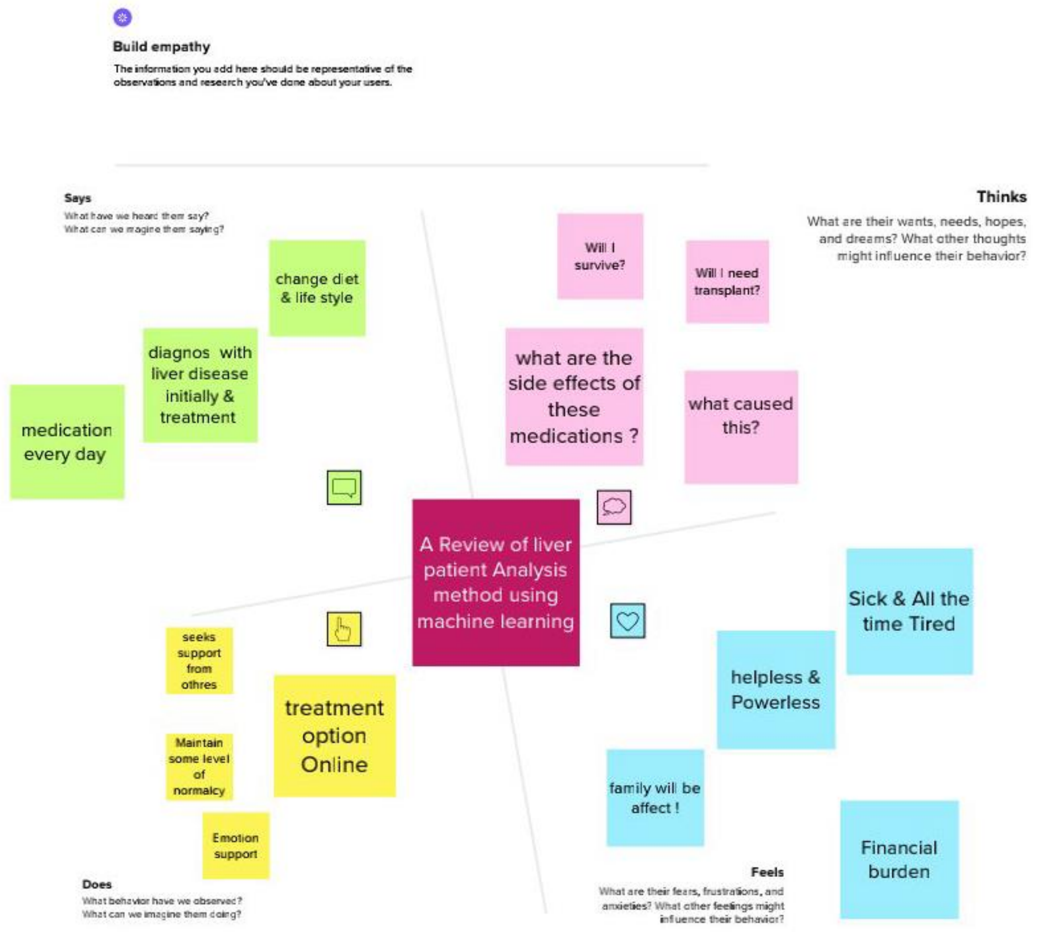
INTRODUCTION:

Liver diseases averts the normal function of the liver. This disease is caused by an assortment of elements that harm the liver. Diagnosis of liver infection at the preliminary stage is important for better treatment. In today’s scenario devices like sensors are used for detection of infections. Accurate classification techniques are required for automatic identification of disease samples. This disease diagnosis is very costly and complicated. Therefore, the goal of this work is to evaluate the performance of different Machine Learning algorithms in order to reduce the high cost of liver disease diagnosis. 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. In this project we will analyse the parameters of various classification algorithms and compare their predictive accuracies so as to find out the best classifier for determining the liver disease. This project compares various classification algorithms such as Random Forest, Logistic Regression, KNN and ANN Algorithm with an aim to identify the best technique. Based on this study, Random Forest with the highest accuracy outperformed the other algorithms and can be further utilised in the prediction of liver disease and can be recommended to the user.

Define Problem / Problem Understanding

Emphathy map:

An empathy map is a template that organizes a user's behaviours and feelings to create a sense of empathy between the user and your team. The empathy map represents a principal user and helps teams better understand their motivations, concerns, and user experience.


Build empathy
The information you add here should be representative of the observations and research you've done about your users.
Says
What have we heard them say? What can we imagine them saying?
Thinks
What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?
change diet & life style
diagnos with liver disease initially & treatment
medication every day
Will I survive?
Will I need transplant?
what are the side effects of these medications ?
what caused this?
A Review of liver patient Analysis method using machine learning
Sick & All the time Tired
helpless & Powerless
family will be affect !
Financial burden
seeks support from othres
Maintain some level of normalcy
Emotion support
treatment option Online
Does
What behavior have we observed? What can we imagine them doing?
Feels
What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible**.**

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

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

Importing the libraries

```
import pandas as pd

import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib  import rcParams
from scipy import stats

data=pd.read_csv('/content/indian_liver_patient.csv')
data.head()
```

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		

```
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
```

```
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.isnull().sum()
```

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

```
['Albumin_and_Globulin_Ratio']=data.fillna(data['Albumin_and_Globulin_Rati
o'].mode()[0])
data.isnull().sum
```

```
Age                                  0
```

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

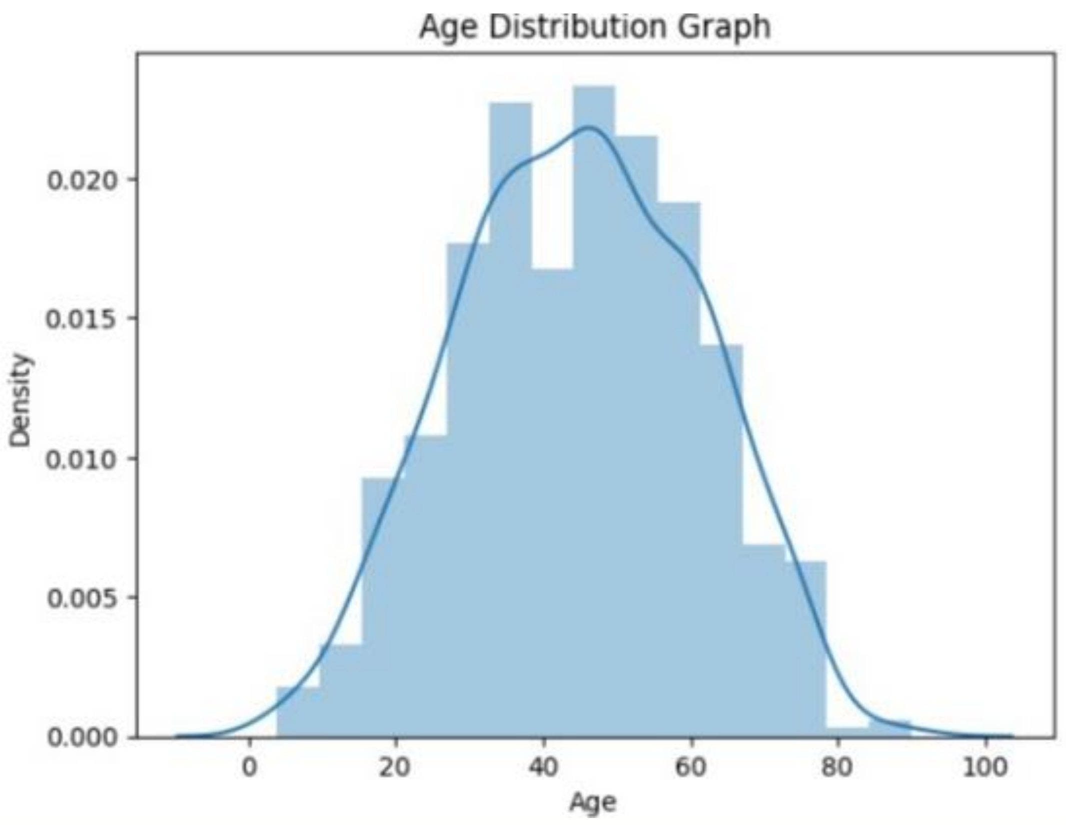
```
data.describe()
```

	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	Dataset
count	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	579.000000	583.000000
mean	44.746141	3.298799	1.486106	290.576329	80.713551	109.910806	6.483190	3.141852	0.947064	1.286449
std	16.189833	6.209522	2.808498	242.937989	182.620356	288.918529	1.085451	0.795519	0.319592	0.452490
min	4.000000	0.400000	0.100000	63.000000	10.000000	10.000000	2.700000	0.900000	0.300000	1.000000
25%	33.000000	0.800000	0.200000	175.500000	23.000000	25.000000	5.800000	2.600000	0.700000	1.000000
50%	45.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000	3.100000	0.930000	1.000000

	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	Dataset
75%	58.000000	2.600000	1.300000	298.000000	60.500000	87.000000	7.200000	3.800000	1.100000	2.000000
max	90.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.600000	5.500000		

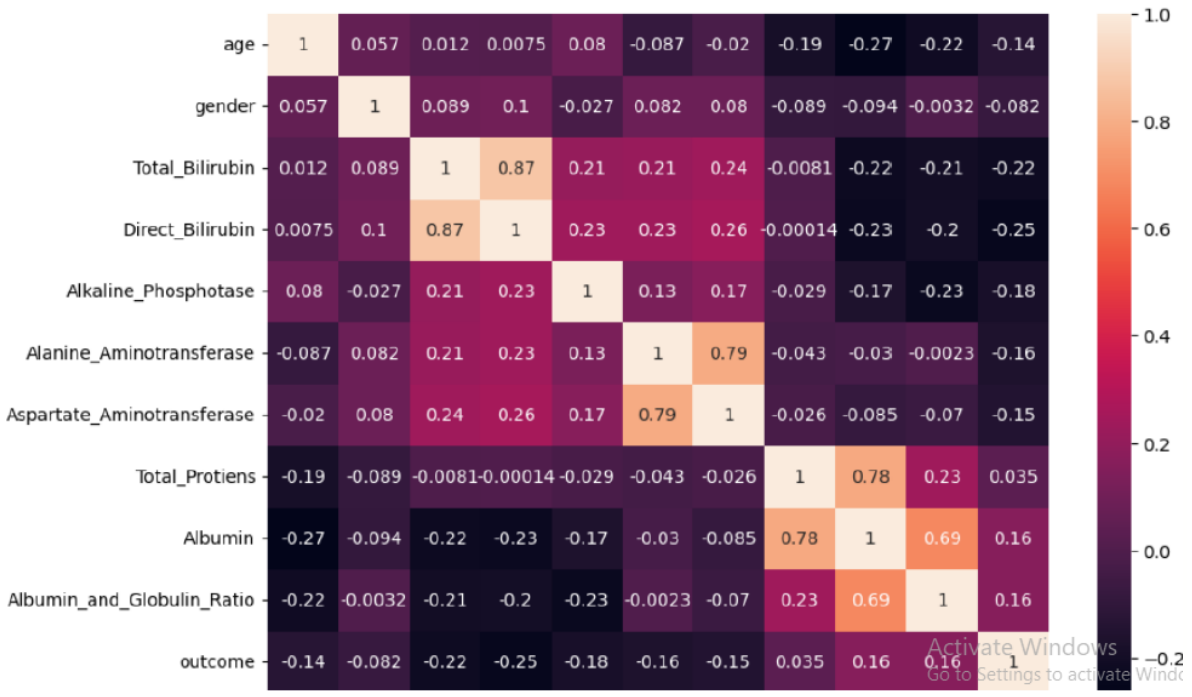
```
sns.distplot(data['Age'])
plt.title('Age Distribution Graph')
plt.show()
```

```
sns.distplot(data['Age'])
```

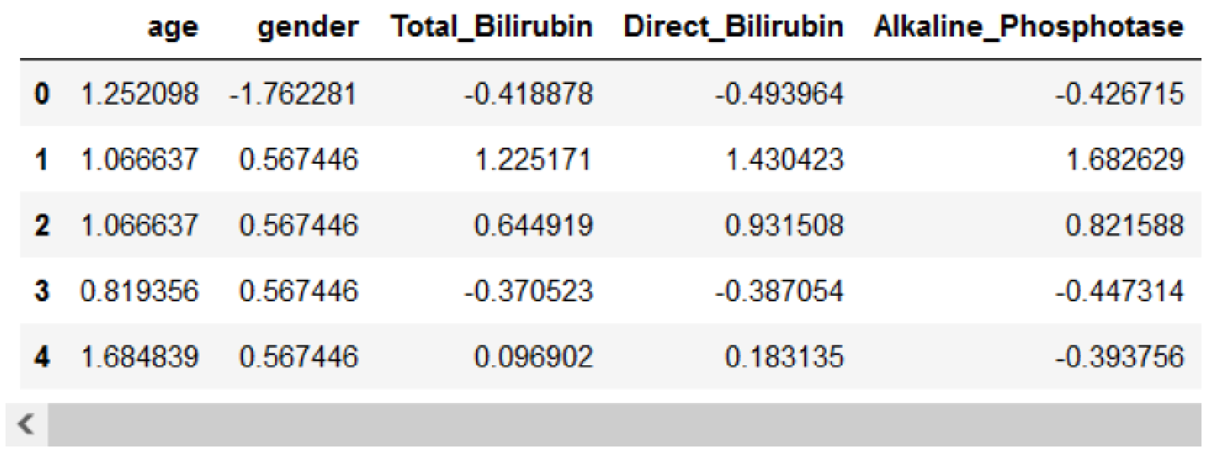


```
sns.countplot(data['outcome']),hue=data['gender']
```

```
plt.figure(figsize=(10,7))
```



```
from sklearn.preprocessing import scale
X_scaled=PD.DataFrame(scale(X),Columns=X.Columns
X_scaled()
```



	age	gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase
0	1.252098	-1.762281	-0.418878	-0.493964	-0.426715
1	1.066637	0.567446	1.225171	1.430423	1.682629
2	1.066637	0.567446	0.644919	0.931508	0.821588
3	0.819356	0.567446	-0.370523	-0.387054	-0.447314
4	1.684839	0.567446	0.096902	0.183135	-0.393756

Random Forest model:

A function named RandomForestClassifier is imported and train and test data are passed as the parameters. Inside the function, RandomForestClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
 import sklearn
 from RandomForestClassifier()
 model1=RandomForestClassifier()
 y_predict=model1.predict(x_test)
 rfc1
 rfc1=accurancy(y_test,y_predict)
 print(classification_report(X_test,y_predict))
```

Decision tree model:

A function named DecisionTreeClassifier is imported and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
from sklearn.tree import DecisionTreeClassifier
model1=DecisionTreeClassifier()
model1.fit(x_train_test,y_train_test)
y_predict=model4.predict(x_test)
dfc1=accurancy_score(y_test,y_predict)
dfc1
pd.crosstab(y_test,y_predict)
print(classification_report(y_test,y_predict))
```

KNN model :

A function named K KNeighborsClassifier is imported and train and test data are passed as the parameters. Inside the function, KNeighborsClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```
from sklearn. neighbour’s import KNeighborsClassifier
model2.fit(x_train_smote,y_train_smote)
model2=KNeighborsClassifier()
y_predict=model2.predict(x_test)
knn1=(accurancy_score(y_test,y_predict))
knn1
pd.crosstab(y_test,y_predict)
print(classification_report(y_test,y_predict))
```

Logistic Regression model :

A function named Logistic Regression is imported and train and test data are passed as the parameters. Inside the function, Logistic Regression algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```
from sklearn.linear_model import LogisticRegression
model5=LogisticRegression()
model.fit(x_train_smote,y_train_smote)
y_predict=model5.predict(x_test)
logil.accurancy_score(y_test,y_predict)
logil
pd.crosstab(y_test,y_predict)
print(classification_report(y_test,y_predict))
```

ANN model :

Building and training an Artificial Neural Network (ANN) using the Keras library with TensorFlow as the backend. The ANN is initialised as an instance of the Sequential class, which is a linear stack of layers. Then, the input layer and two hidden layers are added to the model using the Dense class, where the number of units and activation function are specified. The output layer is also added using the Dense class with a sigmoid activation function. The model is then compiled with the Adam optimizer, binary cross-entropy loss function, and accuracy metric. Finally, the model is fit to the training data with a batch size of 100, 20% validation split, and 100 epochs

Performance Testing & Hyperparameter Tuning:

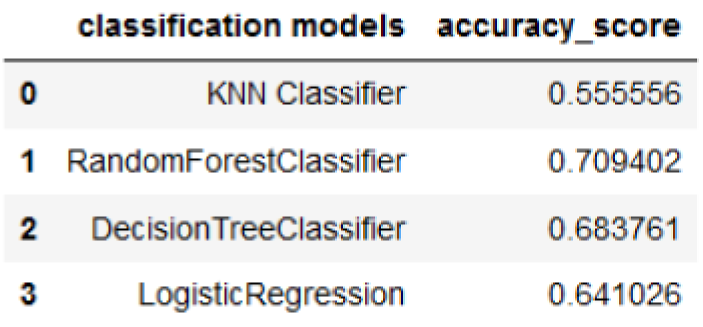
Testing model with multiple evaluation metrics:

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using accuracy, score to compare between models.

Compare the model :

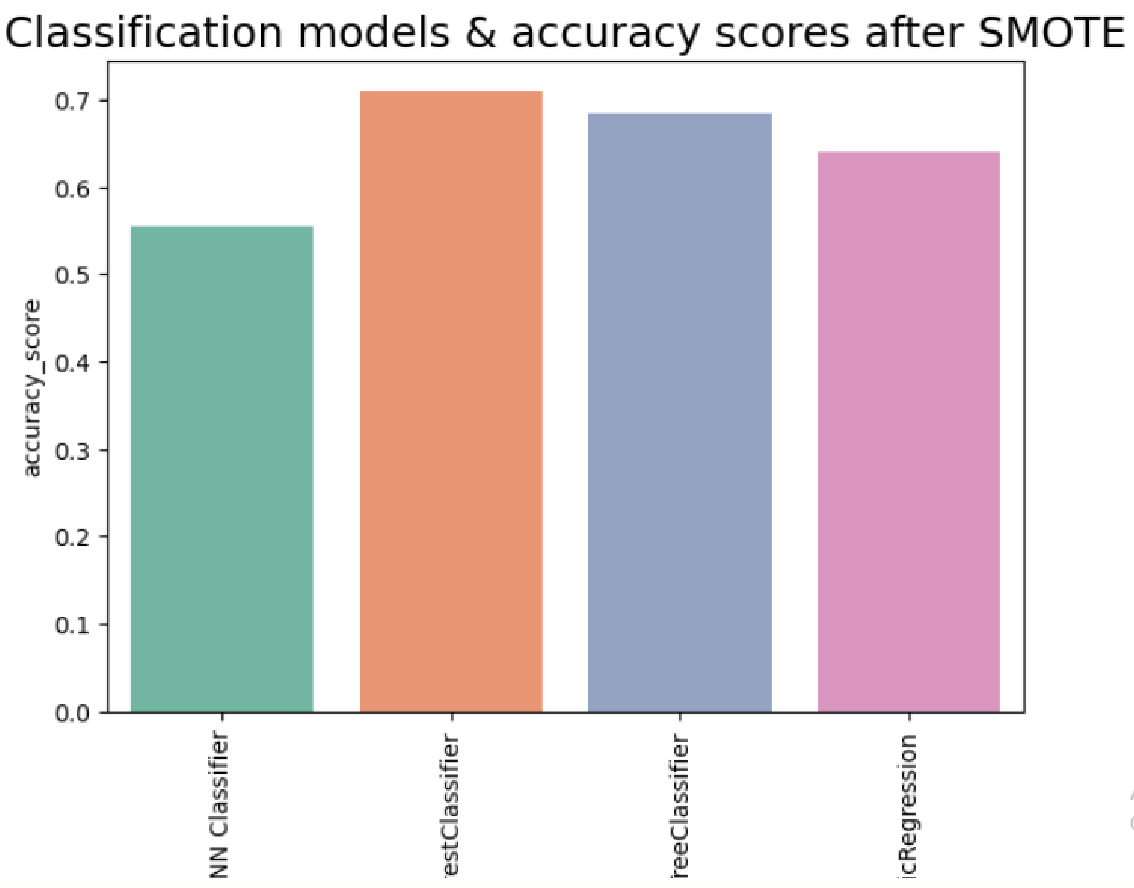
For comparing the above four models, the Accuracy function is defined.
acc_smote=[[‘KNN Classifier’,knn1]
[‘RandomClassiefier’,rfc1],[‘DecisionTreeClassiefier’,dfc1],[‘LogisticRegression’,logil]]
LiverPatient_pred=pd.DataFrame(acc_smote,columns=[‘classification models’,accurancy_score’])

```
LiverPatient_pred
```



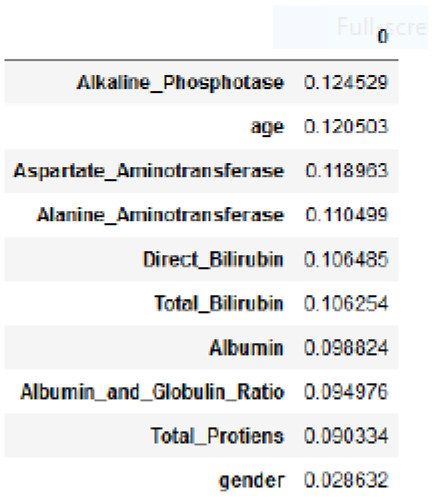
	classification models	accuracy_score
0	KNN Classifier	0.555556
1	RandomForestClassifier	0.709402
2	DecisionTreeClassifier	0.683761
3	LogisticRegression	0.641026

```
plt.figure(figsize= (7.5))
plt.xticks(rotation=90)
plt.title(‘Classification models & accurancy_score after SMOTE’ fontsize=18
sns.barplot(X=”Classification models”,y=”accurancy_score”, data= LiverPatient_pred,palette=”Set2”)
```



```
from sklearn.ensemble import ExtraClassiefier
model= ExtraClassiefier()
model.fit(X,y)
```

```
dd.pd.DataFrame(model.features_importances,index=x.columns)sort
values(ascending=False)
dd
```

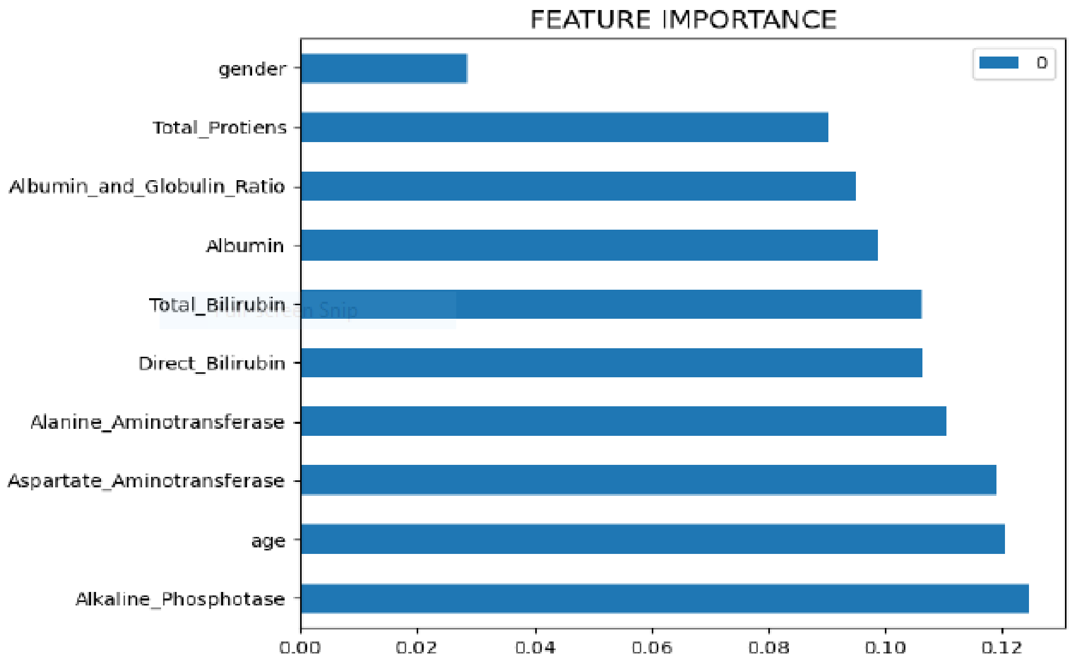


	0
Alkaline_Phosphotase	0.124529
age	0.120503
Aspartate_Aminotransferase	0.118963
Alanine_Aminotransferase	0.110499
Direct_Bilirubin	0.106485
Total_Bilirubin	0.106254
Albumin	0.098824
Albumin_and_Globulin_Ratio	0.094976
Total_Protiens	0.090334
gender	0.028632

Identifying Important Features:

10 attributes are passed to predict the actucal outcome, Its necessary to identify the l important feature to determine the output. Here we are using function called feature_importance to identify the important features among the available attributes and understand with a visualization.

```
dd.plot (kind='barh',fisize(5,6))
plt.title(FEATURE.IMPORTANCE,fontsize=14)
```

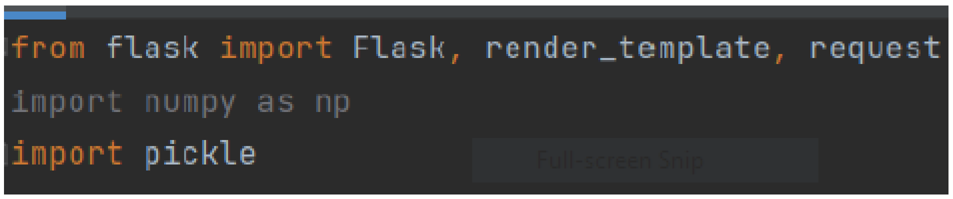


Direct_Bilirubin & Total_Bilirubin are the most important features to predict the outcome

Model Deployment

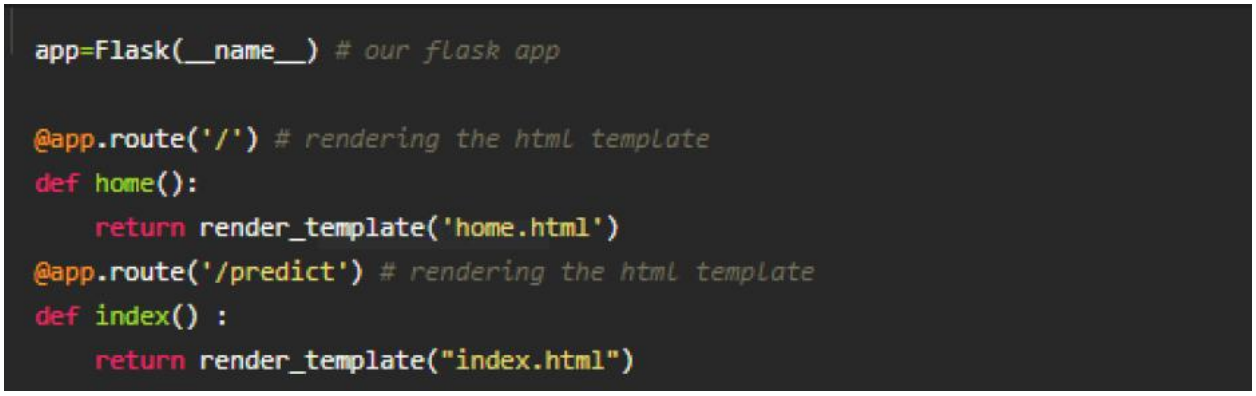
Save the best model:

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.



```
from flask import Flask, render_template, request
import numpy as np
import pickle
```

Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument. And render HTML page:

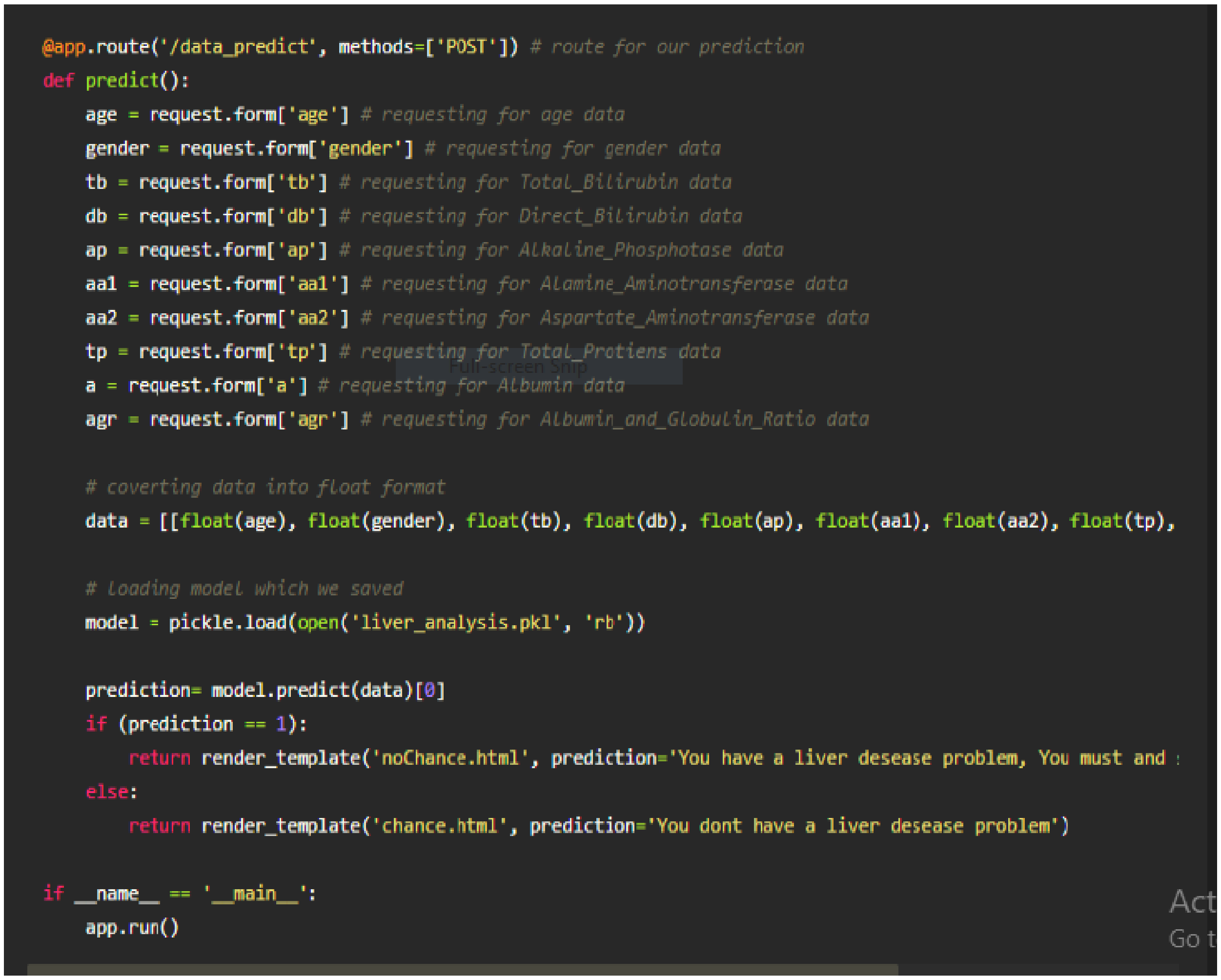

```
app=Flask(__name__) # our flask app

@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")
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.
In the above example, '/' 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 will be rendered.

Whenever you enter the values from the html page the values can be retrieved using

POST Method.

Retrieves the value from UI:



```
@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 we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:

```
If _name_=’_main_’:
    app.run()
```

Run the web application:

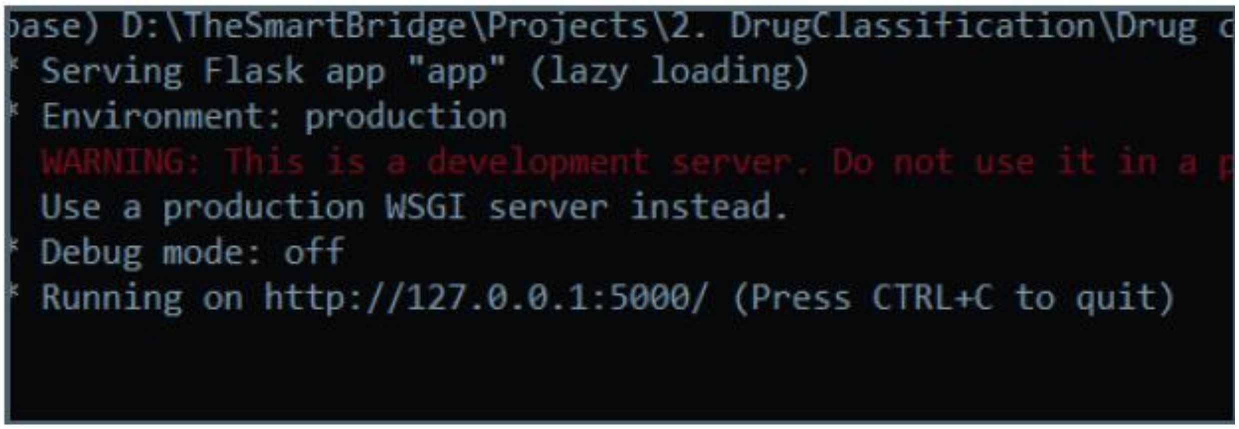
● Open anaconda prompt from the start menu

● Navigate to the folder where your python script is.

● Now type “python app.py” command

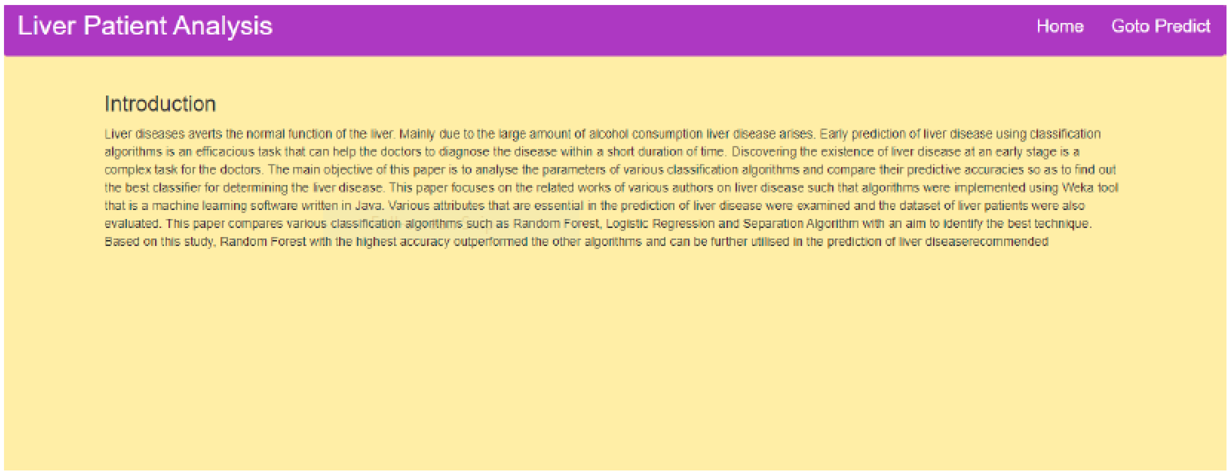
● Navigate to the localhost where you can view your web page.

● Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



```
base) D:\TheSmartBridge\Projects\2. DrugClassification\Drug c
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a p
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Now,Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result
Now,



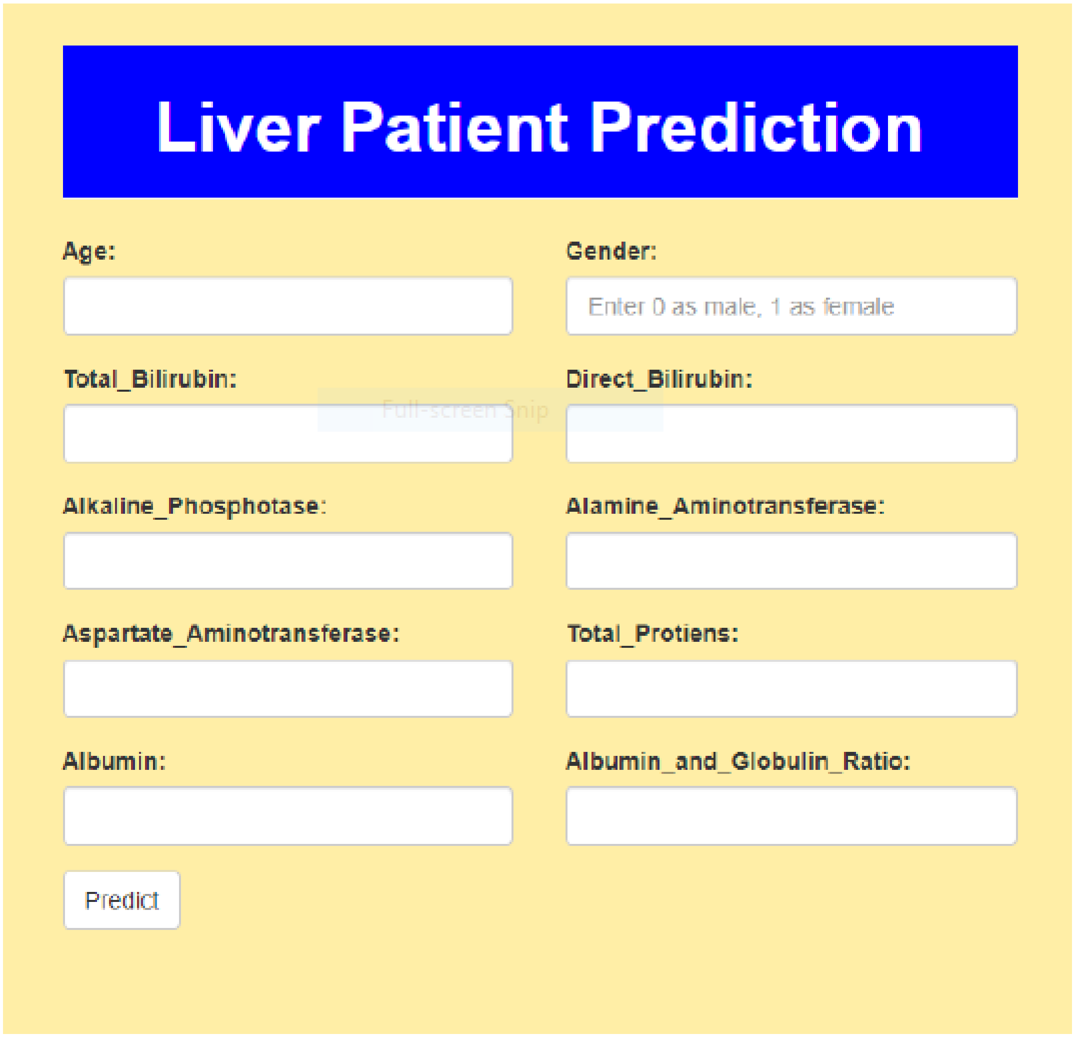
Liver Patient Analysis

Home Goto Predict

Introduction

Liver diseases averts 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. Discovering the existence of liver disease at an early stage is a complex task for the doctors. The main objective of this paper is to analyse the parameters of various classification algorithms and compare their predictive accuracies so as to find out the best classifier for determining the liver disease. This paper focuses on the related works of various authors on liver disease such that algorithms were implemented using Weka tool that is a machine learning software written in Java. Various attributes that are essential in the prediction of liver disease were examined and the dataset of liver patients were also evaluated. This paper compares various classification algorithms such as Random Forest, Logistic Regression and Separation Algorithm with an aim to identify the best technique. Based on this study, Random Forest with the highest accuracy outperformed the other algorithms and can be further utilised in the prediction of liver diseaserecommended

Now,when you click Go to predict the button from the banner you will get redirected to the prediction page.



Liver Patient Prediction

Age:

Gender:

Enter 0 as male, 1 as female

Total_Bilirubin:

Direct_Bilirubin:

Alkaline_Phosphotase:

Alamine_Aminotransferase:

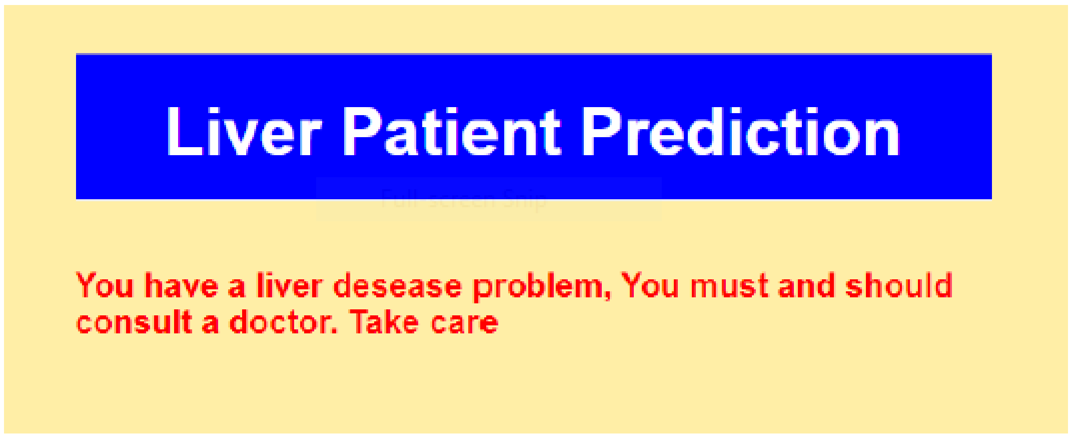
Aspartate_Aminotransferase:

Total_Protiens:

Albumin:

Albumin_and_Globulin_Ratio:

Inputs- Now, the user will give inputs to get the predicted page after giving details user has to click on Predict Button to get the result.


Liver Patient Prediction

You have a liver desease problem, You must and should consult a doctor. Take care