

A Review of Liver Patient Analysis Methods Using Machine Learning

TEAM MEMBER'S NAME: VENI.N

NANDHINI.D

GEETHALAKSHMI.S

MONISHA.D

**CLASS : B.Sc. COMPUTER SCIENCE-III
YEAR**

TABLE OF INDEX

S.NO	DESCRIPTION	PAGE.NO
1.	INTRODUCTION	03
2.	PROBLEM DEFINITION & DESIGN THINKING	06
3.	RESULT	11
4.	ADVANTAGES AND DISADVANTAGES	31
5.	APPLICATIONS	32
6.	CONCLUSION	33
7.	FUTURE SCOPE	34
8.	APPENDIX	35

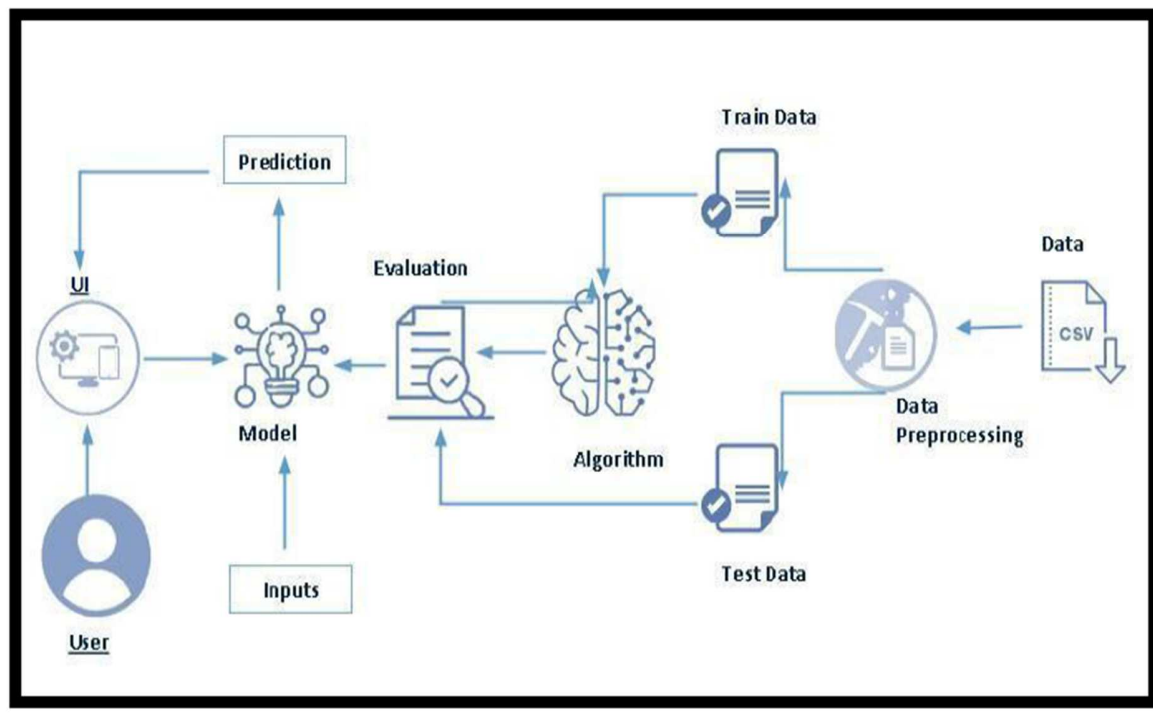
1.INTRODUCTION

LIVER PATIENT ANALYSIS USING MACHINE LEARNING:

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.

Technical Architecture:



1.1 OVERVIEW

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

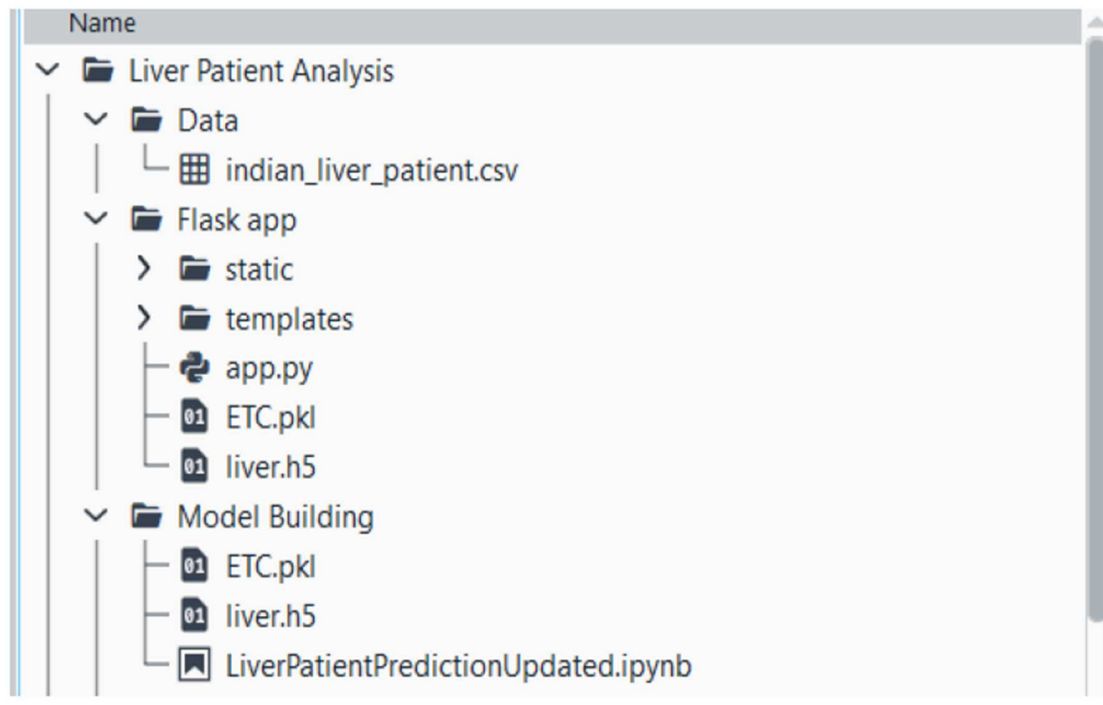
To accomplish this, we have to complete all the activities listed below,

- Define Problem / Problem Understanding
 - Specify the business problem
 - Business requirements
 - Literature Survey
 - Social or Business Impact. .
- Data Collection & Preparation
 - Collect the dataset
 - Data Preparation
- Exploratory Data Analysis
 - Descriptive statistical
 - Visual Analysis
- Model Building
 - Training the model in multiple algorithms
 - Testing the model
- Performance Testing & Hyperparameter Tuning
 - Testing model with multiple evaluation metrics
 - Comparing model accuracy before & after applying hyperparameter tuning
- Model Deployment
 - Save the best model
 - Integrate with Web Framework
- Project Demonstration & Documentation
 - Record explanation Video for project end to end solution

- Project Documentation-Step by step project development procedure

Project Structure:

Create the Project folder which contains files as shown below



- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- ETC.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains a model training file.

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. Using a machine learning prediction models, liver diseases can be predicted using those health parameters in early stages.

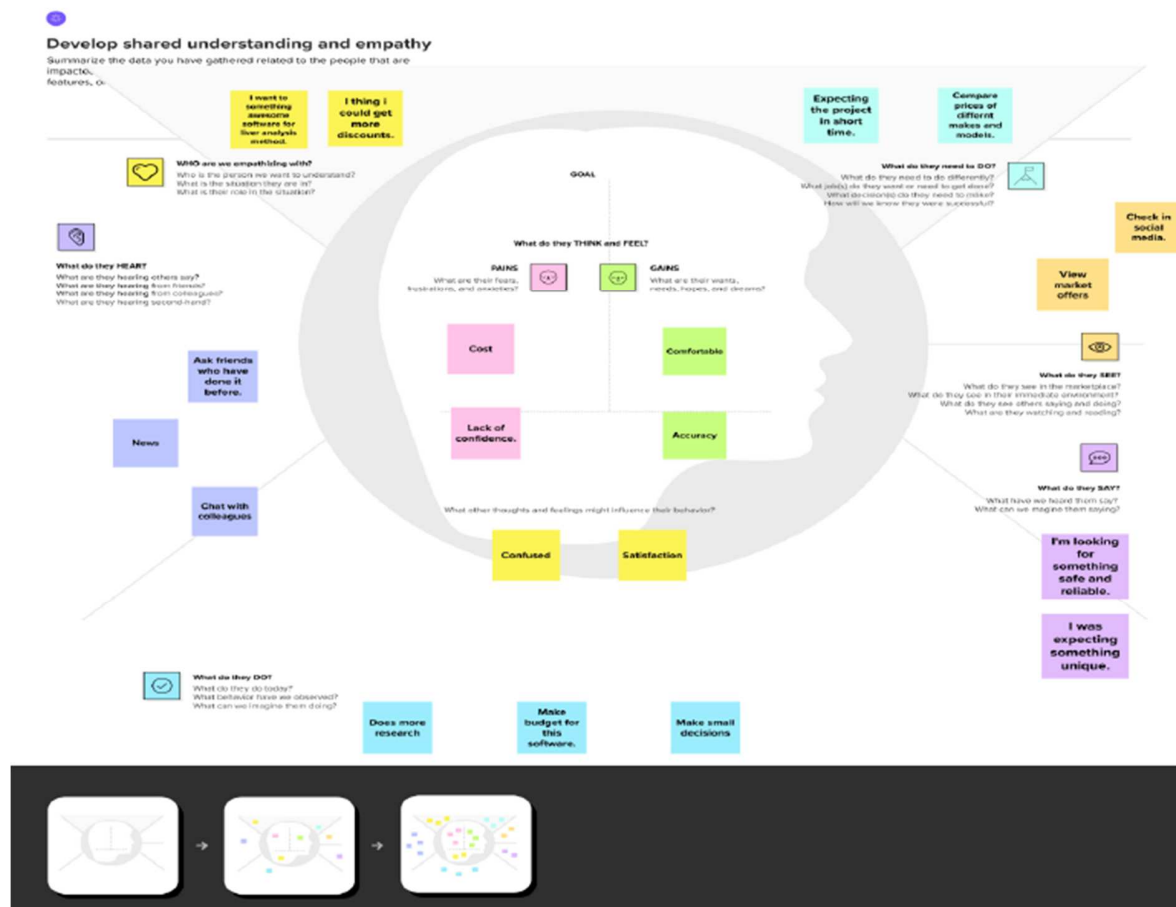
2.PROBLEM DEFINITION & DESIGN THINKING

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.

Patients with Liver disease have been continuously increasing because of excessive consumption of alcohol, inhale of harmful gases, intake of contaminated food, pickles and drugs and other factors. This dataset was used to evaluate prediction algorithms in an effort to reduce burden on doctors. Use these patient records to build a prediction model that will predict which patients have liver disease and which ones do not.

2.1 EMPATHY MAP

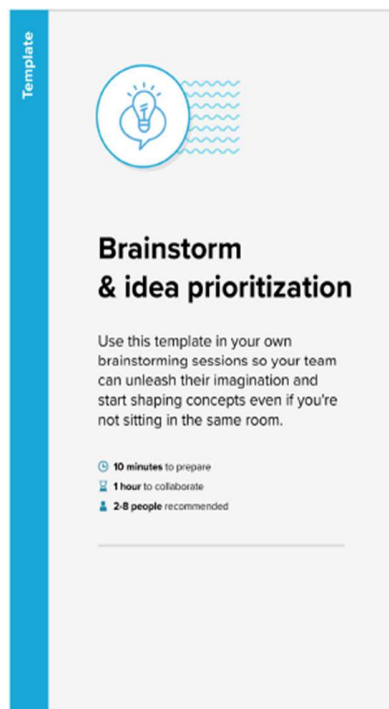
In the ideation phase we have empathized as our patient liver disease prediction using machine learning and we have acquired the details which are represented in the Empathy Map given below.



2.2 IDEATION & BRAINSTORMING

Under this activity our team members have gathered and discussed various idea to solve our project.Each member contributed 6 to 10 ideas after gathering all ideas we have assessed the impact and feasibility of each point.Finally,we have assign the priority for each point based on the impact value

STEP 1:Team Gathering , Collaboration and Select the problem



STEP-2:Brainstorm,Idea Listing and Grouping

1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

[🕒 5 minutes](#)

How might we liver patient analysis method using machine learning?

Key rules of brainstorming
To run an effective and productive session

Stay in topic.	Encourage wild ideas.
Defer judgement.	Listen to others.
Go for volume.	If possible, be visual.

Step:2- Brainstorm, idea Listing and Grouping

 Brainstorm

Write down any ideas that come to mind that address your problem statement.

© 2010 Pearson Education, Inc.

Tip: Many water molecules in drinking water have had their oxygen atoms replaced by deuterium (D), known as "heavy" deuterium!



STEP-3:Idea prioritization

3

Group Ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

20 minutes

TIP Add customisable tags to sticky notes to make it easier to find, browse, organise, and categorise important ideas as they surface within your mind.

Step:3- Idea Prioritization

Prescription: Your doctor should tell you on the order (page 2) what's important to watch for. Place your order on file and be determined which items are important and which are desirable.

© 2018 Pearson Education, Inc.





After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons



Share the mural

Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.



Export the mural

Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward



Strategy blueprint

Define the components of a new idea or strategy.

[Open the template →](#)



Customer experience journey map

Understand customer needs, motivations, and obstacles for an experience.


[Open the template →](#)



Strengths, weaknesses, opportunities & threats

Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.

[Open the template →](#)

 [Share template feedback](#)

3.RESULT

Read the datasets

	Age	Gender	Total_ Bilirubin	Direct_ Bilirubin	Alkaline_ Phosphot ase	Alamine_A minotransf erase	Aspartate_ Aminotrans ferase	Total_ Proteins	Albumin	Albumin_and_ Globulin_Ratio	Dataset
0	65	Femal e	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

Handling missing values

<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

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

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

Age	0
Gender	0
Total_Bilirubin	0
Direct_Bilirubin	0
Alkaline_phosphotase	0
Alamine_Aminotransferase	0
Aspartate_Aminotransferase	0
Total_protiens	0
Albumin	0

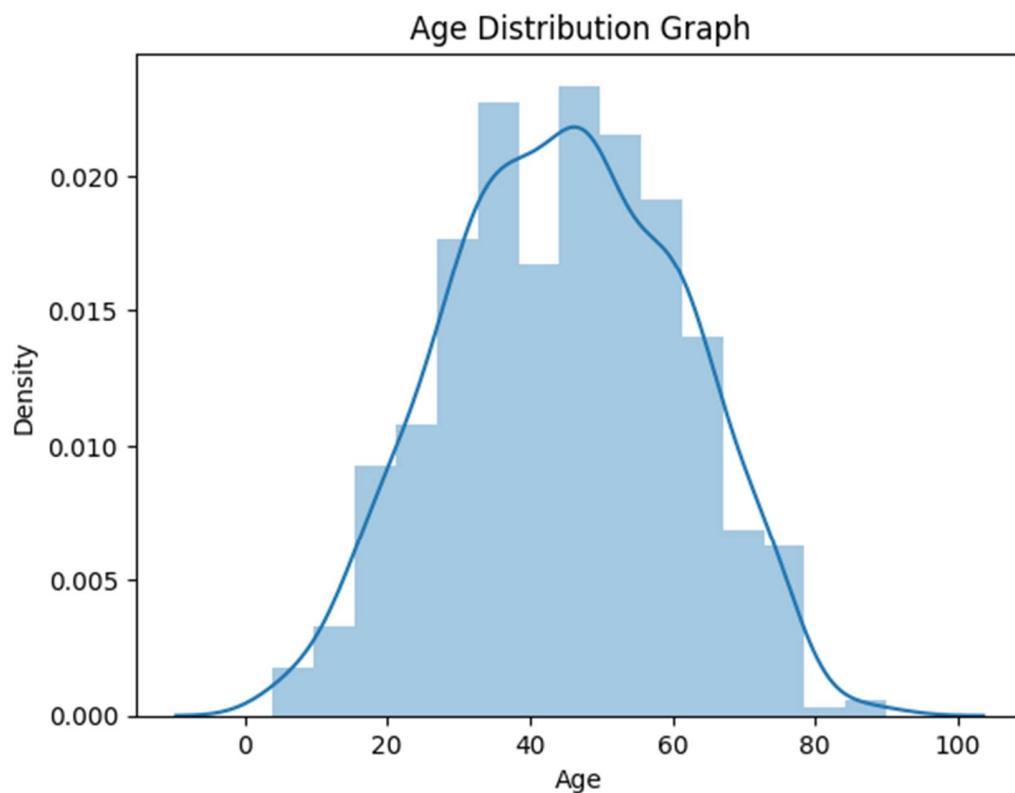
Albumin_and_Globulin_Ratio	0
Dataset	0
dtype: int64	0

EXPLORATORY ANALYSIS:
DISCRIPTIVE STATISTICAL

	Age	Gender	Total_ Bilirubin	Direct_ Bilirubin	Alkaline_ Phosphatase	Alamine_ Aminotran sferase	Aspartate_ Aminotrans ferase	Total_proteins
Count	583.000000	583.000000	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	6.483190	3.141852
Std	16.189833	6.209522	2.808498	242.937989	182.620356	288.918529	1.085451	0.795519
Min	4.000000	0.400000	0.100000	63.000000	10.000000	10.000000	2.700000	0.900000
25%	33.000000	0.800000	0.200000	175.500000	23.000000	25.000000	5.800000	2.600000
50 %	45.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000	3.100000
75%	58.000000	2.600000	1.300000	298.000000	60.500000	87.000000	7.200000	3.800000
Max	90.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.600000	5.500000

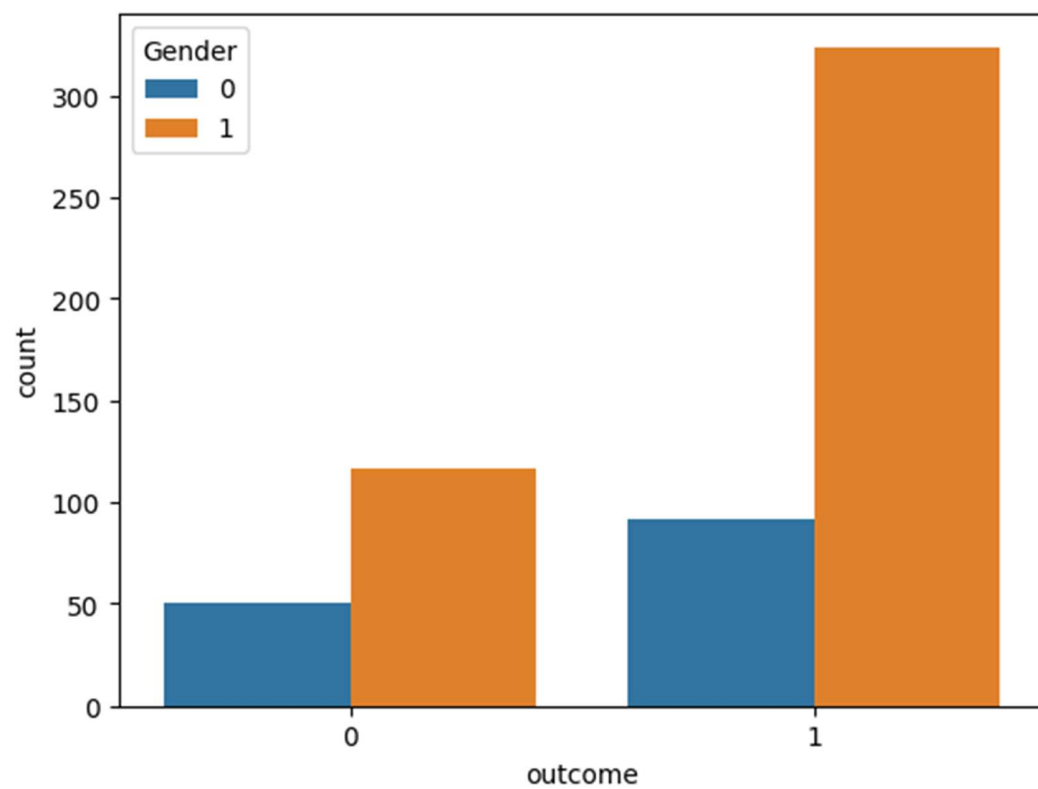
Number of columns=11

UNIVARIATE ANALYSIS

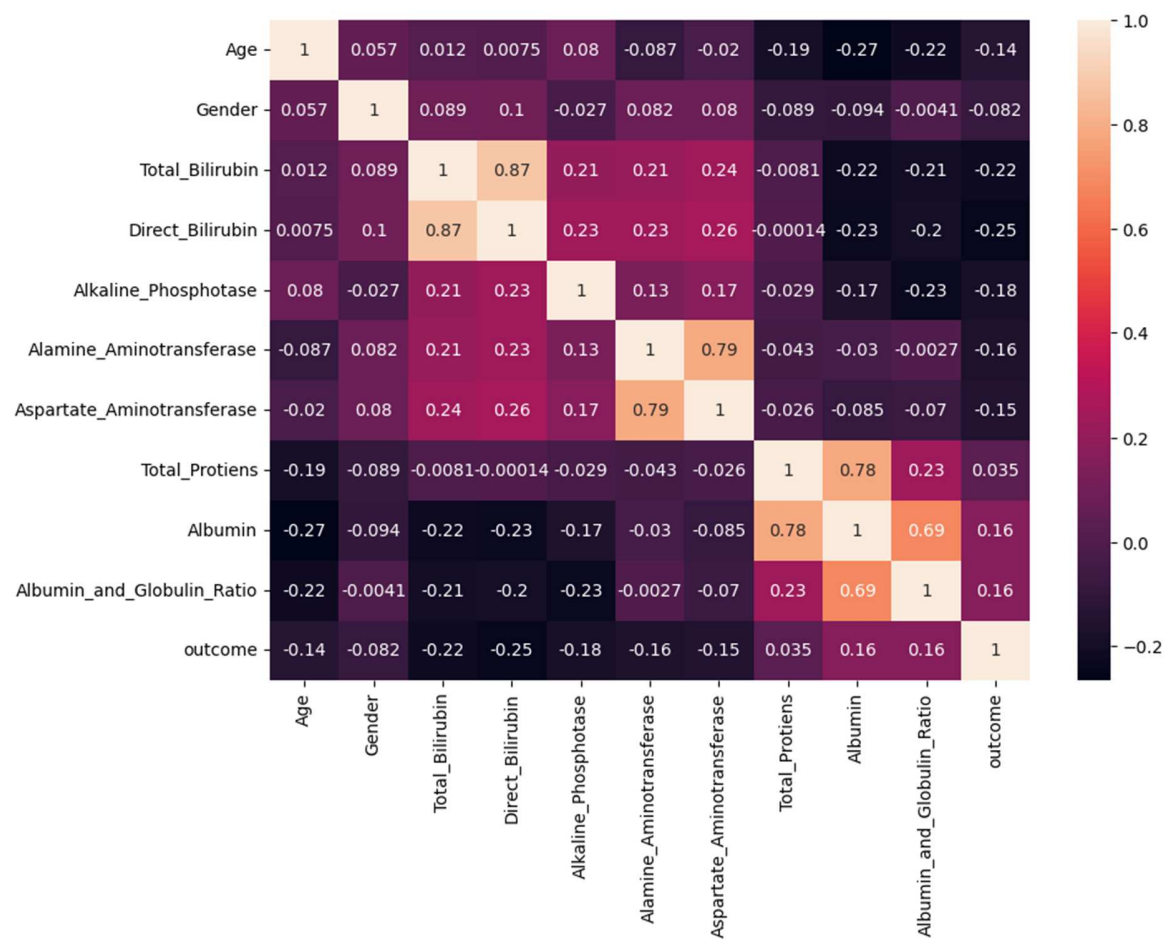


BIVARIATE ANALYSIS:

<Axes: xlabel='outcome', ylabel='count'>



MULTIVARIATE ANALYSIS:



SCALING THE DATA:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphatase
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	0.931503	0.821588
3	0.819356	0.567446	-0.379523	-0.387054	-0.447314
4	1.684839	0.567446	0.096902	0.183135	-0.393756

Number of columns=11

HANDLING IMBALANCE DATA:

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>

Collecting imblearn

Downloading imblearn-0.0-py2.py3-none-any.whl (1.9 kB)

Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.9/dist-packages (from imblearn) (0.10.1)

Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.9/dist-packages (from imbalanced-learn->imblearn) (1.10.1)

Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-packages (from imbalanced-learn->imblearn) (1.1.1)

Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from imbalanced-learn->imblearn) (3.1.0)

Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.9/dist-packages (from imbalanced-learn->imblearn) (1.22.4)

Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.9/dist-packages (from imbalanced-learn->imblearn) (1.2.2)

Installing collected packages: imblearn

Successfully installed imblearn-0.0

1 329

2 137

Name: outcome, dtype: int64

1 329

2 329

Name: outcome, dtype: int64

MODEL BUILDING

TRAINING THE MODEL IN MULTIPLE ALGORITHMS

1.RANDOM FOREST MODEL:

0.844311377245509

	precision	recall	f1-score	support
1	0.84	0.81	0.82	75
2	0.85	0.87	0.86	92
accuracy			0.84	167
macro avg	0.84	0.84	0.84	167
weighted avg	0.84	0.84	0.84	167

2.DECISION TREE MODEL:

0.7245508982035929

	precision	recall	f1-score	support
1	0.68	0.72	0.70	75
2	0.76	0.73	0.74	92
accuracy			0.72	167
macro avg	0.72	0.72	0.72	167
weighted avg	0.73	0.72	0.73	167

3.KNN MODEL:

0.7844311377245509

	precision	recall	f1-score	support
1	0.80	0.69	0.74	75
2	0.77	0.86	0.81	92
accuracy			0.78	167
macro avg	0.79	0.78	0.78	167
weighted avg	0.79	0.78	0.78	167

4.LOGISTIC REGRESSION MODEL:

0.7125748502994012

	precision	recall	f1-score	support
1	0.74	0.56	0.64	75
2	0.70	0.84	0.76	92
accuracy			0.71	167
macro avg	0.72	0.70	0.70	167
weighted avg	0.72	0.71	0.71	167

5.ANN MODEL:

Epoch 1/100

4/4 [=====] - 1s 75ms/step - loss: 0.6878 - accuracy: 0.6640 - val_loss: 0.6787 - val_accuracy: 0.7234

Epoch 2/100

4/4 [=====] - 0s 13ms/step - loss: 0.6741 - accuracy: 0.7016 - val_loss: 0.6617 - val_accuracy: 0.7234

Epoch 3/100

4/4 [=====] - 0s 13ms/step - loss: 0.6557 - accuracy: 0.7016 - val_loss: 0.6380 - val_accuracy: 0.7234

Epoch 4/100

4/4 [=====] - 0s 13ms/step - loss: 0.6300 - accuracy: 0.7016 - val_loss: 0.6051 - val_accuracy: 0.7234

Epoch 5/100

4/4 [=====] - 0s 13ms/step - loss: 0.5944 - accuracy: 0.7016 - val_loss: 0.5597 - val_accuracy: 0.7234

TESTING THE MODEL

```
array([1])
```

```
array([1])
```

```
4/4 [=====] - 0s 3ms/step
```

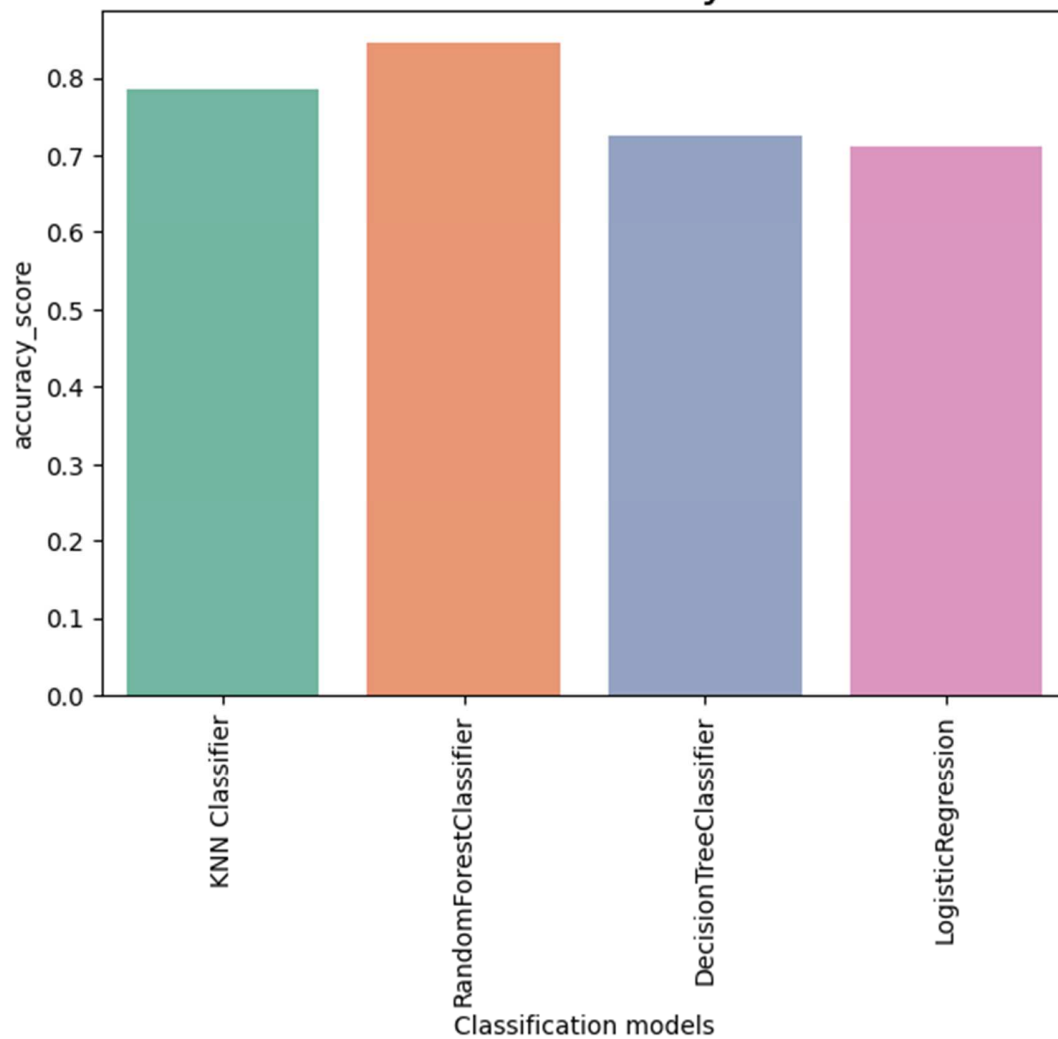
```
array([[ True],  
       [ True],  
       [ True],  
       [ True],
```

COMPARE THE MODEL:

	Classification models	accuracy_score
0	KNN Classifier	0.784431
1	RandomForestClassifier	0.844311
2	DecisionTreeClassifier	0.724551
3	LogisticRegression	0.712575

<Axes: title={'center': 'Classification models & accuracy scores after SMOTE'},
xlabel='Classification models', ylabel='accuracy_score'>

Classification models & accuracy scores after SMOTE



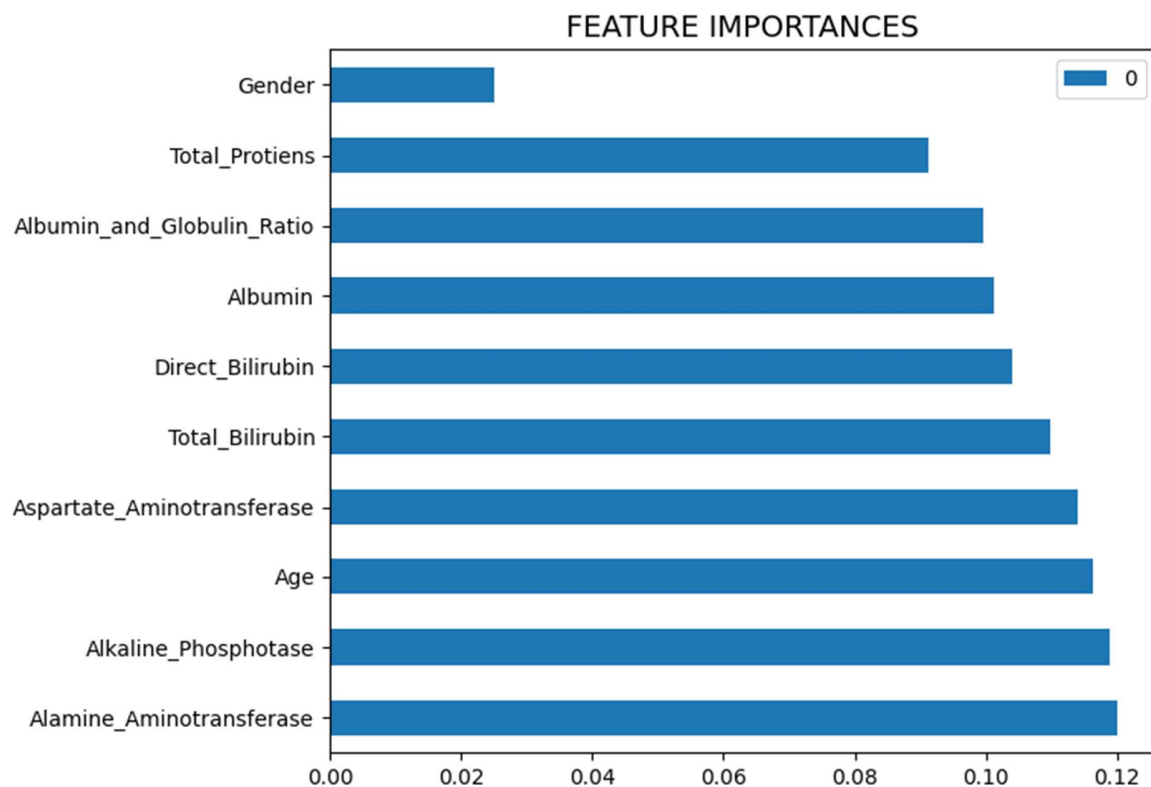
ExtraTreesClassifier()

array ([0.11629797, 0.02507542, 0.10992013, 0.1039708 , 0.11891955, 0.11997948,
0.11389892, 0.09125799, 0.10115149, 0.09952825])

Alamine_Aminotransferase	0.119979
Alkaline_Phosphotase	0.118920
Age	0.116298
Aspartate_Aminotransferase	0.113899
Total_Bilirubin	0.109920
Direct_Bilirubin	0.103971
Albumin	0.101151
Albumin_and_Globulin_Ratio	0.099528
Total_Protiens	0.091258
Gender	0.025075

IDENTIFICATION IMPORTANT FEATURES:

Text(0.5, 1.0, 'FEATURE IMPORTANCES')



INTEGRATE WITH WEB FRAMEWORK

BUILDING HTML PAGES:

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <title>Liver Prediction Model</title>
```

```
</head>
```

```
<body>
```

```
<h1>Liver Disease Prediction</h1>
```

```
<h2>Enter the details to check whether you have liver disease or not!</h2>
```

```
<div class="ml-container">
```

```
<form action="{ { url_for('predict') } }" method="POST">
```


<h3>Age</h3>

<input id="first" name="Age" placeholder="in Year" required="required">

<h3>Gender</h3>

<input id="second" name="Gender" placeholder="Male = 1, Female=0" required="required">

<h3>Total Bilirubin</h3>

<input id="third" name="Total_Bilirubin" placeholder="Total Bilirubin" required="required">

<h3>Alkaline Phosphatase</h3>

<input id="fourth" name="Alkaline_Phosphatase" placeholder="Alkaline Phosphatase" required="required">

<h3>Alamine Aminotransferase</h3>

<input id="fifth" name="Alamine_Aminotransferase" placeholder="Alamine Aminotransferase" required="required">

<h3>Aspartate Aminotransferase</h3>

<input id="sixth" name="Aspartate_Aminotransferase" placeholder="Aspartate Aminotransferase" required="required">

<h3>Total Protiens</h3>

<input id="seventh" name="Total_Protiens" placeholder="Total Protiens" required="required">

```
<br>
```

```
<h3>Albumin</h3>
```

```
<input id="eight" name="Albumin" placeholder="Albumin" required="required">
```

```
<br>
```

```
<h3>Albumin and Globulin Ratio</h3>
```

```
<input id="ninth" name="Albumin_and_Globulin_Ratio" placeholder="Albumin  
and Globulin Ratio" required="required">
```

```
<br>
```

```
<br>
```

```
<br>
```

```
<button id="predict" type="predict ">Predict</button>
```

```
<br>
```

```
<br>
```

```
<br>
```

```
<br>
```

```
</form>
```

```
</div>
```

```
<style>
```

```
body
```

```
{
```

```
background-image:url("liver.jpeg");
```

```
background-position: center;
```

```
background-repeat: no-repeat;
```

```
background-size: 100% 100%;
```

```
}
```



```
h1{
color:red
}
h2{
color:navy
}
body{
font-family: Arial, Helvetica,sans-serif;
text-align: center;
margin: 0;
padding: 0;
width: 100%;
height: 100%;
display: flex;
flex-direction: column;
}
.heading_font{
font-family: 'cambria', cursive;
font-size: 50px;
font-weight: normal;
}
```

```
#first {
border-radius: 14px;
height: 25px;
```

```
        width: 150px;
        font-size: 20px;
        text-align: center;
    }
    #second {
        border-radius: 14px;
        height: 25px;
        width: 220px;
        font-size: 20px;
        text-align: center;
    }
    #third {
        border-radius: 14px;
        height: 25px;
        width: 180px;
        font-size: 20px;
        text-align: center;
    }
    #fourth {
        border-radius: 14px;
        height: 25px;
        width: 250px;
        font-size: 20px;
        text-align: center;
    }
```

```
#fifth {  
    border-radius: 14px;  
    height: 25px;  
    width: 270px;  
    font-size: 20px;  
    text-align: center;  
}
```

```
#sixth {  
    border-radius: 14px;  
    height: 25px;  
    width: 280px;  
    font-size: 20px;  
    text-align: center;  
}
```

```
#seventh {  
    border-radius: 14px;  
    height: 25px;  
    width: 170px;  
    font-size: 20px;  
    text-align: center;  
}
```

```
#eight {  
    border-radius: 14px;  
    height: 25px;  
    width: 150px;
```

```
        font-size: 20px;
        text-align: center;
    }
    #ninth {
        border-radius: 14px;
        height: 25px;
        width: 280px;
        font-size: 20px;
        text-align: center;
    }
    #predict {
        width: 120px;
        height: 43px;
        text-align: center;
        border-radius: 14px;
        font-size: 18px;
    }
</style>
<h2>
<b>
</body>
</html>
```

BUILDING PYTHON CODE:

```
import flask
from flask import Flask, render_template, request
import pickle
import numpy as np
import sklearn
from flask_ngrok import run_with_ngrok
import warnings
warnings.filterwarnings('ignore')

app = Flask(__name__)
run_with_ngrok(app)
model1 = pickle.load(open('ETC.pkl', 'rb'))

@app.route('/', methods=['GET'])
def home():
    return render_template('index.html')

@app.route('/', methods=['GET', "POST"])
def predict():
    input_values = [float(x) for x in request.form.values()]
    inp_features = [input_values]
    print(inp_features )
    prediction = model.predict(inp_features)
    if prediction == 1:
```

```

    return render_template('index.html', prediction_text=('You have a liver
disease,please consult a doctor'))

else:

    return rende_ template('index.html', prediction_text=('You don't have a liver
disease '))

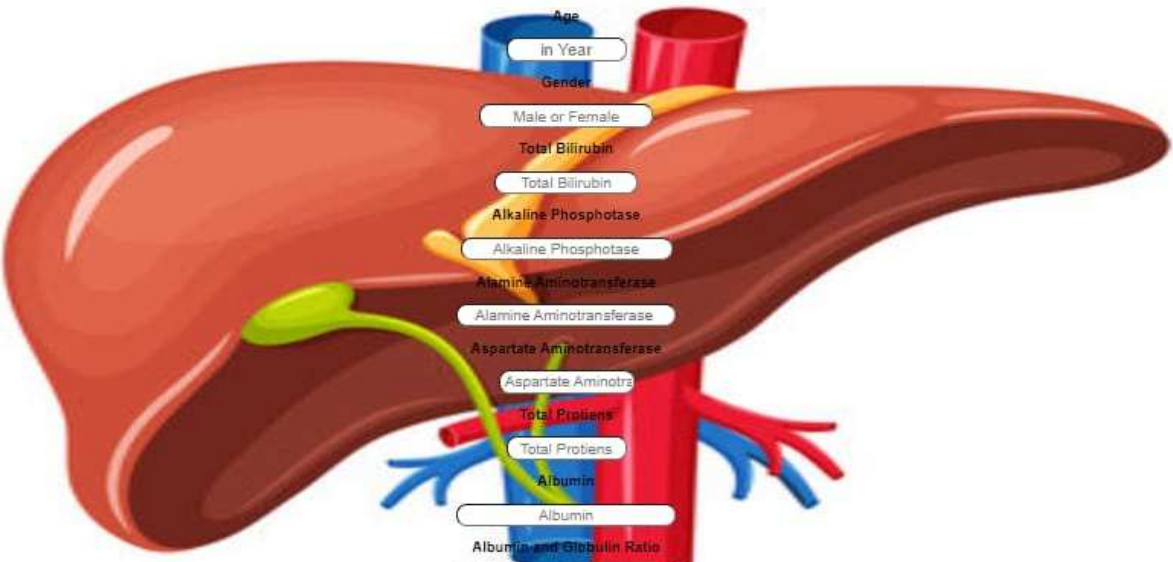
app.run()

```

RUN THE WEB APPLIACTIONS

Liver Disease Prediction

Enter the details to check whether you have liver disease or not!



Age

in Year

Gender

Male or Female

Total Bilirubin

Alkaline Phosphatase

Alamine Aminotransferase

Aspartate Aminotransferase

Total Proteins

Albumin

Albumin and Globulin Ratio

Predict

{{ prediction_text }}

4.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

- Liver disease can be Predict in early stages for using this method.
- Diagnostic criterion standard confirmed diagnostic value Etiologic suggestion differential diagnosis.
- Grade and stage evaluation Therapeutic decision.
- Treatment evaluation follow-up comparison of treated and untreated patients.
- Can perform differential diagnosis.
- Assess the degree and stage of liver damage.
- It can decide the therapy.

DISADVANTAGES:

- Highly invasive test.
- The potential complications include death.
- Significant sampling error .
- High cost.
- Inter-observer variation
- Available in only a small number of countries.

5. APPLICATIONS

- The prediction of liver disease is an important factor for effective treatment and reducing serious health consequences. We have to construct a prediction model based on machine learning. Early prediction using this model might bring benefits from treatment reduction, and medical cost decrease.
- Therefore, building a model that will help doctors to predict whether a patient is likely to have liver disease, at its early stages will be a great advantage.
- Diagnosis of liver disease at a preliminary stage is important for better treatment. We also compare different algorithms for better accuracy.

6. CONCLUSION

In this study machine learning techniques were used to predict liver disease at an early stage and Random Forest Regression model showed better performance than other classification techniques.

This prediction outcome has the potential to help clinicians make more potential and meaningful decisions about liver disease diagnosis and treatment.

7.FUTURE SCOPE

The scope of the study will focus on the correct o liver patients as early as possible.

- To use classification approaches that help successful early liver diagnosis and treatment. As patients are not to spend more time in the hospital and they cure immediately.
- To medical classification approaches as automatic or real time classification tools which may useful for experts to identify the chances of disease and conscious prescription of future medical examinations and treatment.
- This research will help medical practitioners to demonstrate awareness of evidence-based treatment and therefore this application will give more support to such society for their future work and assessments.
- Moreover this also helps the number of trainees for performing their research work and practices.

8.APPENDIX

SOURCE CODE

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
```

READ THE DATASET:

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

HANDLING MISSING VALUES:

```
data.info()

#returns the sum of all null values

data.columns

#RENAME THE COLUMN NAME

data.rename(columns={'Dataset':'outcome'},inplace=True)

data.isnull().any()

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

HAVING MISSING VALUES:

```
data['Albumin_and_Globulin_Ratio'].fillna(data['Albumin_and_Globulin_Ratio'].mode()[0],inplace=True)
data.isnull().sum()
```

HANDLING CATAGORIAL VALUES:

```
from sklearn.preprocessing import LabelEncoder
lc=LabelEncoder()
```

```
data['Gender']= lc.fit_transform(data['Gender'])
```

EXPLORATORY DATA ANALYSIS

DESCRIPTIVE STATISTICAL:

```
data.describe()
```

VISUAL ANALYSIS

UNIVARIATE ANALYSIS:

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

BIVARIATE ANALYSIS:

```
sns.countplot(x=data['outcome'],apply(lambda x:1 if x == 1 else 0),hue=data['Gender'])
```

MULTIVARIATE ANALYSIS:

```
plt.figure(figsize=(10,7))  
sns.heatmap(data.corr(),annot=True)
```

SCALING THE DATA

```
x=data.iloc[:, :-1]  
y=data.iloc[:, -1]
```

```
x
```

```
from sklearn.preprocessing import scale  
x_scaled= pd.DataFrame (scale(x),columns=x.columns)  
x_scaled.head()
```

SPLITTING DATA INTO TRAIN AND TEST

```
x=data.iloc[:, :-1]  
y= data.outcome
```

```
from sklearn.preprocessing import StandardScaler  
scaler = StandardScaler().fit(x)  
x_scaled = scaler.transform(x)
```

```
from imblearn.over_sampling import SMOTE
smote= SMOTE()
x_smote, y_smote = smote.fit_resample(x, y)
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_smote, y_smote,test_size=0.2,random_state=42)
```

HANDLING IBALANCE DATA:

```
pip install imblearn
```

```
y_train.value_counts()
```

```
x_train_smote,y_train_smote= smote.fit_resample(x_train,y_train)
```

```
y_train_smote.value_counts()
```

MODEL BUILDING

TRAINING THE MODEL IN MULTIPLE ALGORITHMS

RANDOM FOREST MODEL:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
model1= RandomForestClassifier(random_state=100)
model1.fit(x_train, y_train)
y_train = model1.predict(x_train)
y_test_sm= model1.predict(x_test)
rfc1=(accuracy_score(y_test, y_test_sm))
rfc1

print (classification_report(y_test,y_test_sm))
```

DECISION TREE MODEL:

```
from sklearn.tree import DecisionTreeClassifier
model2=DecisionTreeClassifier()
model2.fit(x_train,y_train)
y_train = model2.predict(x_train)
y_test_sm = model2.predict(x_test)
dct1=accuracy_score(y_test, y_test_sm)
dct1
```

```
print(classification_report(y_test,y_test_sm))
```

KNN MODEL:

```
from sklearn.neighbors import KNeighborsClassifier
model3=KNeighborsClassifier()
model3.fit(x_train_smote,y_train_smote)
y_train= model3.predict(x_train)
y_test_sm = model3.predict(x_test)
knn1= accuracy_score(y_test, y_test_sm)
knn1
```

```
print(classification_report(y_test,y_test_sm))
```

LOGISTIC REGRESSION MODEL:

```
from sklearn.linear_model import LogisticRegression
model4=LogisticRegression ()
model4.fit(x_train,y_train)
y_train= model4.predict(x_train)
y_test_sm = model4.predict(x_test)
logi1=accuracy_score(y_test, y_test_sm)
logi1
```

```
print(classification_report(y_test,y_test_sm))
```

ANN MODEL:

```
import tensorflow.keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

classifier = Sequential()
classifier.add(Dense(units =100, activation = 'relu', input_dim = 10))
classifier.add(Dense(units =50, activation = 'relu'))
classifier.add(Dense(units =1, activation = 'sigmoid'))
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
classifier.fit(x_train, y_train, batch_size = 100, validation_split=0.4, epochs = 100)
```

TESTING THE MODEL

```
model4.predict([[50,1,1.2,0.8,150,70,80,7.2,3.4,0.8]])
```

```
model4.predict([[50,1,1.2,0.8,150,70,80,7.2,3.4,0.8]])
```

```
classifier.save("liver.h5")
y_pred=classifier.predict(x_test)
```

```
y_pred=(y_pred>0.5)
y_pred
```

```
def predict_exit(sample_value):
    sample_value=np.array(sample_value)
    sample_value=sample_value.reshape(1,-1)
    sample_value=scale(sample_value)
    return classifier.predict(sample_value)
```

```
sample_value=[[50,1,1.2,0.8,150,70,80,7.2,3.4,0.8]]
if predict_exit(sample_value) > 0.5:
    print('Prediction: Liver Patient')
else:
    print('Prediction: Healthy')
```

COMPARE THE MODEL

```
acc_smote=[[ 'KNN Classifier',knn1],[ 'RandomForestClassifier',rfc1],[ 'DecisionTreeClassifier',dt
c1],[ 'LogisticRegression',logi1]]
Liverpatient_pred=pd.DataFrame(acc_smote, columns=[ 'Classification models','accuracy_score']
)
Liverpatient_pred
```

CLASSIFICATION MODEL

```
plt.figure(figsize=(7,5))
plt.xticks(rotation=90)
plt.title('Classification models & accuracy scores after SMOTE',fontsize=18)
sns.barplot(x='Classification models',y='accuracy_score',data=Liverpatient_pred,palette='Set2')
```

#Random forest model performing is well

```
from sklearn.ensemble import ExtraTreesClassifier
model=ExtraTreesClassifier ()
model.fit(x,y)
```

```
model.feature_importances_
```

IDENTIFYING IMPORTANT FEATURES

```
dd=pd.DataFrame(model.feature_importances_,index=x.columns).sort_values(0, ascending=False)
dd
```