ARIGNAR ANNA GOVT ARTS & SCIENCE

COLLEGE,VILLUPURAM – 605 602.

DEPARTMENT OF COMPUTER SCIENCE

MACHINE LEARNING WITH PYTHON

Project Title : A Review of Liver Patient Analysis Method using Machine Learning

Team ID : NM2023TMID16892

Team Leader : DEVANATHAN M

Team member : DHIVISH S

Team member : DINESH R

Team member : ELAYA SOORIYA E

Abstract

Around a million deaths occur due to liver diseases globally. There are several traditional methods to diagnose liver diseases, but they are expensive.

Early prediction of liver disease would benefit all individuals prone to liver diseases by providing early treatment. As technology is growing in health care, machine learning significantly affects health care for predicting conditionsat early stages. This study finds how accurate machine learningis in predicting liver disease.

This present study introduces the liver disease prediction (LDP) method in predicting liver disease that can be utilised by health professionals, stakeholders, students and researchers. Five algorithms, namely Support Vector Machine (SVM), Naïve Bayes, K-Nearest Neighbors (K-NN), Linear Discriminant Analysis (LDA), and Classification and Regression Trees (CART), are selected. The accuracy is compared to uncover the best classification method for predicting liver disease using R and Python. From the results, Random Forest Classifier obtains the best accuracy with 67.7%, and the Decision Tree Classifier achieved 61.1% accuracy, which is above the acceptable level of accuracy and can be considered for liver disease prediction.

Introduction

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 Liver.According to World Health Organization (WHO) report in 2018, the number of deaths due to liver diseases is around one million and ranked 11th in the world with a critical number of fatalities (World Total Deaths, n.d.). Unnoticed at the initial stages, these symptoms are only visible when the disease turns chronic. However, even though the liver is partially infected, it can still function (Devikanniga et al., 2020).

Diagnosis of liver diseases can be divided into three stages i.e., the first stage is liver inflammation, the second is liver scarring (cirrhosis), and the final stage is liver cancer or failure. Since these scenarios are present in liver disease, early prediction is significant to provide better health for New Zealanders. If liver disease is diagnosed early, there will be a chance of early treatment and control of deaths due to liver diseases (Arbain & Balakrishnan, 2019). But when the liver fails to function, few treatments are available except liver transplantation (Shaheamlung et al., 2020), which is very expensive, particularly in New Zealand (Hepatitis C, 2021). Apparently, in New Zealand, 35 - 40% of the population are not diagnosed with Hepatitis C at the early stages because of the asymptomatic behaviour of liver disease. Unfortunately, most of these individuals do not know the risks linked to liver disease. Due to the asymptomatic behaviour and higher costs of liver disease treatment, it is essential to prevent or diagnose early for better treatment.

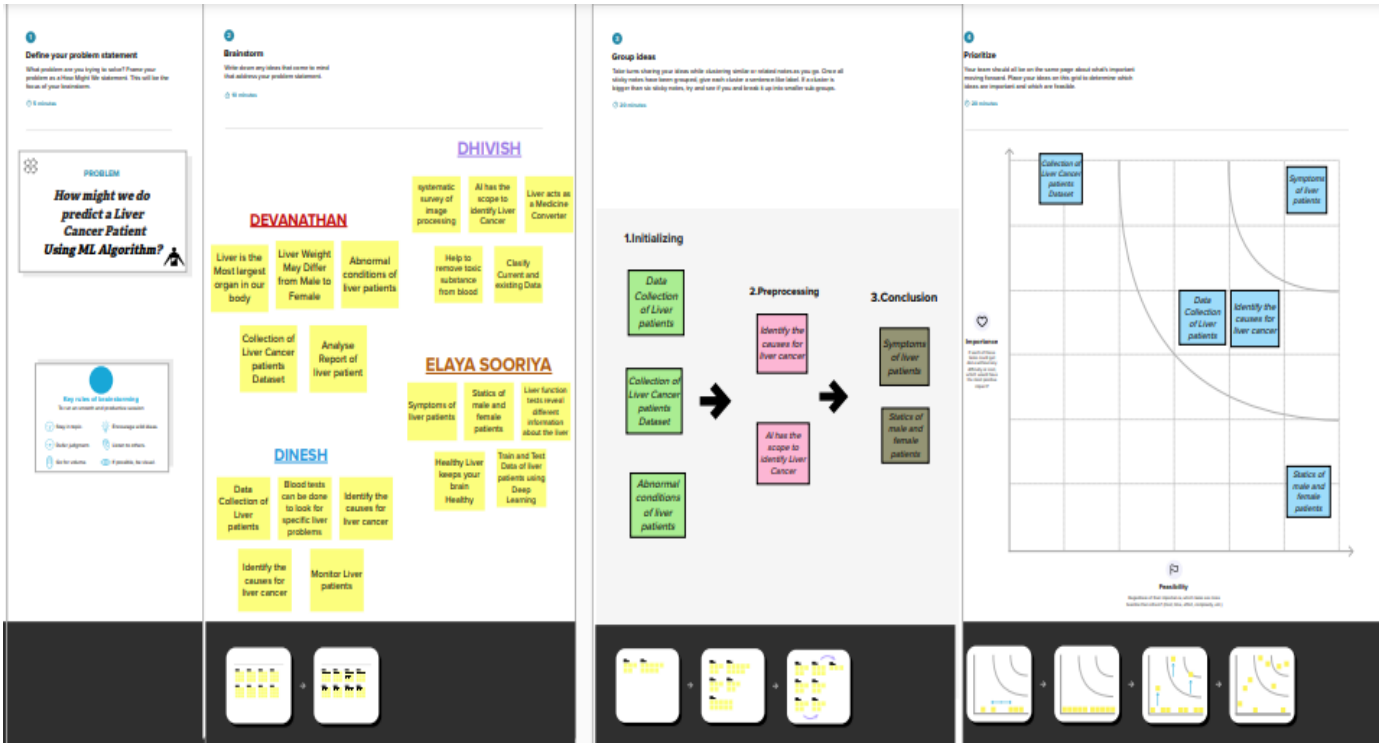
Purpose

Liver function tests can be used to: Screen for liver infections, such as hepatitis. Monitor the progression of a disease, such as viral or alcoholic hepatitis, and determine how well a treatment is working. Measure the severity of a disease, particularly scarring of the liver (cirrhosis)

Problem Definetion & Design Thinking

Empathy Map

Idea Prioritization & Brainstorm map



Result :

Liver Patients analysis using machine learning can provide accurate and realiable results for the diagnosis, prognosis, and treatment of liver disease. Machine learning algorithm can analyse large amounts of patients data and identify patterns that may be difficult for human experts to detect.

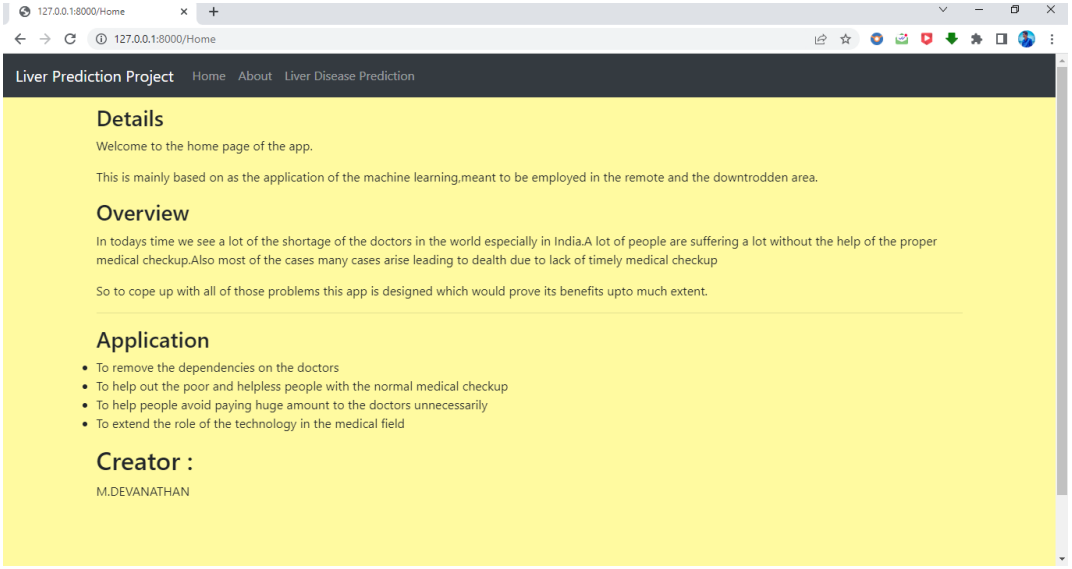
Source code:

Milestone 1:

Home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Home</title>
</head>
<style>
#main-footer
{color: #FFFFFF;
font-family: "Segoe UI";
background: #2B2B2B;
text-align: center;
margin-top: 108px;
padding: 16px;
bottom: 8px;}
</style>
<div class= "container">
<div class="col-md-8">
{% with messages = get_flashed_messages(with_categories=true) %}
{% if messages %}
{% for category, message in messages %}
<div class="alert alert-{{ category }}">
{{ message }}
</div>
{% endfor %}
{% endif %}
{% endwith %}
</div>
<h3 style="margin-top: 5%;">Details </h3>
<p>Welcome to the home page of the app.</p>
<p>This is mainly based on as the application of the machine learning,meant to be employed in the remote and the
downtrodden area.
<h3>Overview</h3>
In todays time we see a lot of the shortage of the doctors in the world especially in India.A lot of people are suffering a
lot without the help of the proper medical checkup.Also most of the cases many cases arise leading to dealth due to lack
of timely medical checkup<p>
<p>So to cope up with all of those problems this app is designed which would prove its benefits upto much extent.
</p><hr></p>
<h3>Application</h3>
<ul>
<li>To remove the dependencies on the doctors</li>
<li>To help out the poor and helpless people with the normal medical checkup</li>
<li>To help people avoid paying huge  amount to the doctors unnecessarily</li>
<li>To extend the role of the technology in the medical field</li>
</ul><h2>Creator :</h2>
<p>       M.DEVANATHAN</p>
</div>
</body>
</html>
```

Output :

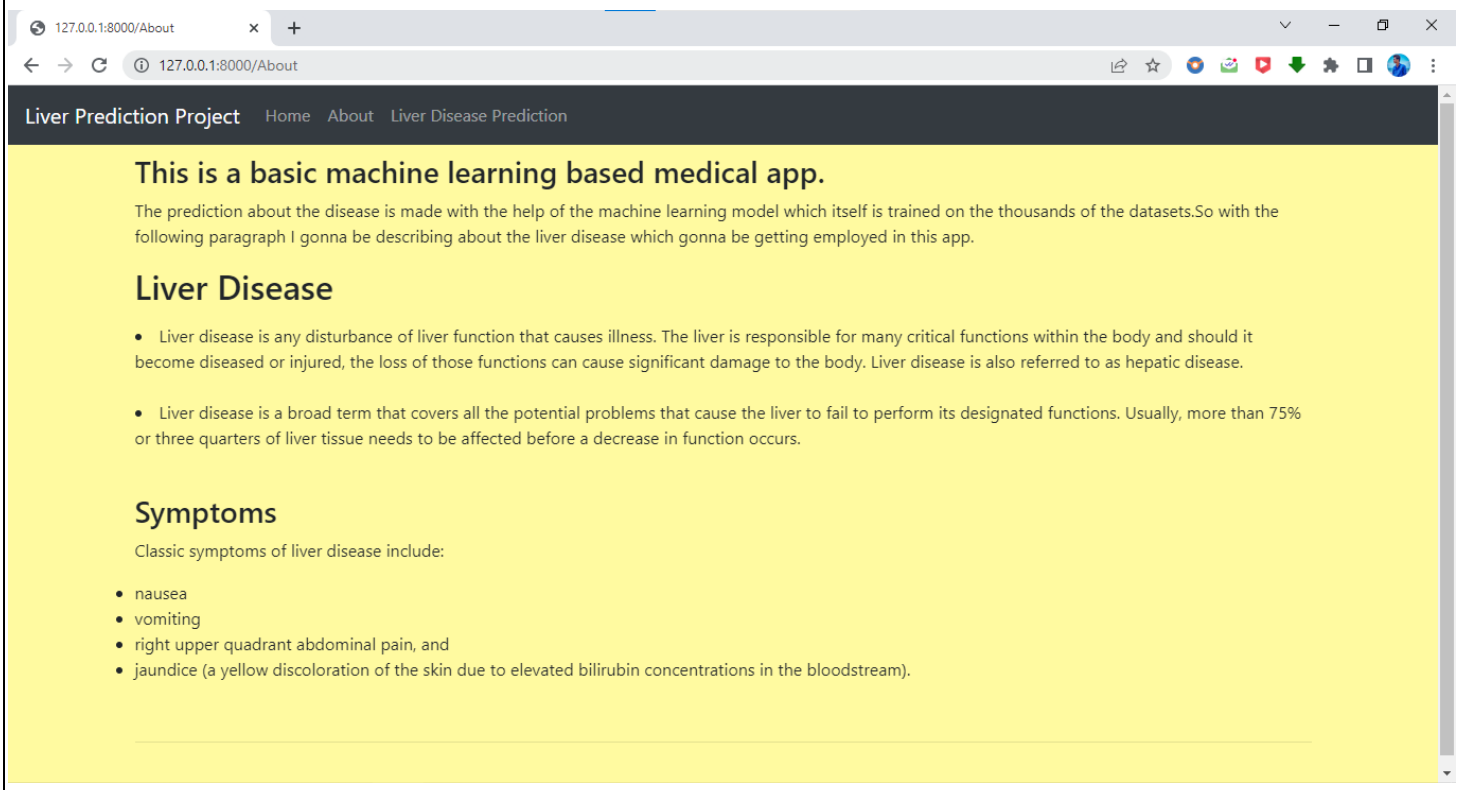


About.html

```
<html>
<body>
<style>
#main-footer
{
color: #FFFFFF;
font-family: "Segoe UI";
background: #2B2B2B;
text-align: center;
margin-top: 75px;
padding: 16px;
bottom: 8px;
}
.img_holder img{
max-width: 100%; max-height: 100%;
}
.btn-file {
position: relative;
overflow: hidden;
}
.btn-file input[type=file] {
position: absolute;
top: 0;
right: 0;
min-width: 100%;
min-height: 100%;
font-size: 100px;
text-align: right;
filter: alpha(opacity=0);
opacity: 0;
outline: none;
background: white;
cursor: inherit;
display: block;
```

```
}
</style>
<div class="container">
<h1>About : </h1>
<h3>This is a basic machine learning based medical app.</h3>
<p>The prediction about the disease is made with the help of the machine learning model which itself is trained on the thousands of the datasets.So with the following paragraph I gonna be describing about the liver disease which gonna be getting employed in this app.</p>

<h2>Liver Disease</h2>
<p><li>Liver disease is any disturbance of liver function that causes illness. The liver is responsible for many critical functions within the body and should it become diseased or injured, the loss of those functions can cause significant damage to the body. Liver disease is also referred to as hepatic disease.</li><br>

<li>Liver disease is a broad term that covers all the potential problems that cause the liver to fail to perform its designated functions. Usually, more than 75% or three quarters of liver tissue needs to be affected before a decrease in function occurs.</li><br></p>
<h3>Symptoms</h3>
<p>Classic symptoms of liver disease include:</p>
<ul>
<li>nausea</li>
<li>vomiting</li>
<li>right upper quadrant abdominal pain, and </li>
<li>jaundice (a yellow discoloration of the skin due to elevated bilirubin concentrations in the bloodstream).</li>
</ul>
<br>
<hr>
<br>
</body>
</htm>
```

Output :



liverprediction.html

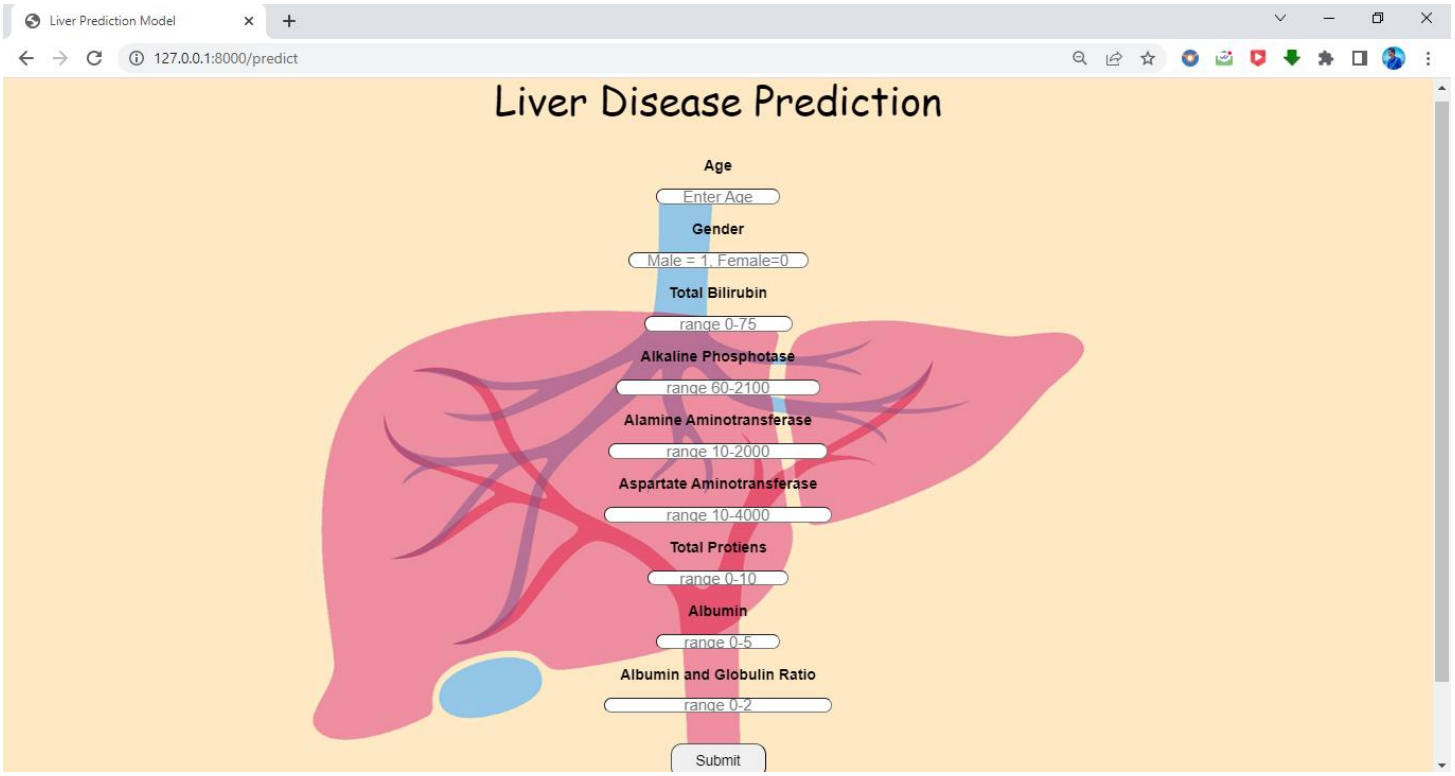
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Liver Prediction Model</title>
</head>
<body>
<div class="container">
<h2 class='container-heading'><span class="heading_font">Liver Disease Prediction</span></h2>
</div>
<div class="ml-container">
<form action="{{ url_for('predict') }}" method="POST">
<br>
<h3>Age</h3>
<input id="first" name="Age" placeholder="Enter Age" required="required">
<br>
<h3>Gender</h3>
<input id="second" name="Gender" placeholder="Male = 1, Female=0" required="required">
<br>
<h3>Total Bilirubin</h3>
<input id="third" name="Total_Bilirubin" placeholder="range 0-75" required="required">
<br>
<h3>Alkaline Phosphotase</h3>
<input id="fourth" name="Alkaline_Phosphotase" placeholder="range 60-2100" required="required">
<br>
<h3>Alamine Aminotransferase</h3>
<input id="fifth" name="Alamine_Aminotransferase" placeholder="range 10-2000" required="required">
<br>
<h3>Aspartate Aminotransferase</h3>
<input id="sixth" name="Aspartate_Aminotransferase" placeholder="range 10-4000" required="required">
<br>
<h3>Total Protiens</h3>
<input id="seventh" name="Total_Protiens" placeholder="range 0-10" required="required">
<br>
<h3>Albumin</h3>
<input id="eight" name="Albumin" placeholder="range 0-5" required="required">
<br>
<h3>Albumin and Globulin Ratio</h3>
<input id="ninth" name="Albumin_and_Globulin_Ratio" placeholder="range 0-2" required="required">
<br>
<br>
<button id="sub" type="submit ">Submit</button>
<br>
<br>
```

```
<p class='footer-description'>Dear Crush ⒹⒷ 2023</p>

</form>
</div>

<style>

/* Background Image */
body
{
background-image:url("https://raw.githubusercontent.com/SagarDhandare/Liver-Disease-Prediction-
Project/main/Images/Liver.jpg");
height: 100%;

/* Center and scale the image nicely */
background-position: center;
background-repeat: no-repeat;
background-size: 100% 100%;

}

/* Color */
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 */
.container-heading{
margin: 0;
}

.heading_font{
color: #black;
font-family: 'Pacifico', cursive;
font-size: 50px;
font-weight: normal;
}
```

```
/* Box */
#first {
border-radius: 14px;
height: 15px;
width: 150px;
font-size: 20px;
text-align: center;
}

#second {
border-radius: 14px;
height: 15px;
width: 220px;
font-size: 20px;
text-align: center;
}

#third {
border-radius: 14px;
height: 15px;
width: 180px;
font-size: 20px;
text-align: center;
}

#fourth {
border-radius: 14px;
height: 15px;
width: 250px;
font-size: 20px;
text-align: center;
}

#fifth {
border-radius: 14px;
height: 15px;
width: 270px;
font-size: 20px;
text-align: center;
}

#sixth {
border-radius: 14px;
height: 15px;
width: 280px;
font-size: 20px;
text-align: center;
}
```

```
#seventh {
border-radius: 14px;
height: 15px;
width: 170px;
font-size: 20px;
text-align: center;
}

#eight {
border-radius: 14px;
height: 15px;
width: 150px;
font-size: 20px;
text-align: center;
}

#ninth {
border-radius: 14px;
height: 15px;
width: 280px;
font-size: 20px;
text-align: center;
}



/* Submit Button */
#sub {
width: 120px;
height: 43px;
text-align: center;
border-radius: 14px;
font-size: 18px;
}

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
</style>
</body>
</html>
```

Output:



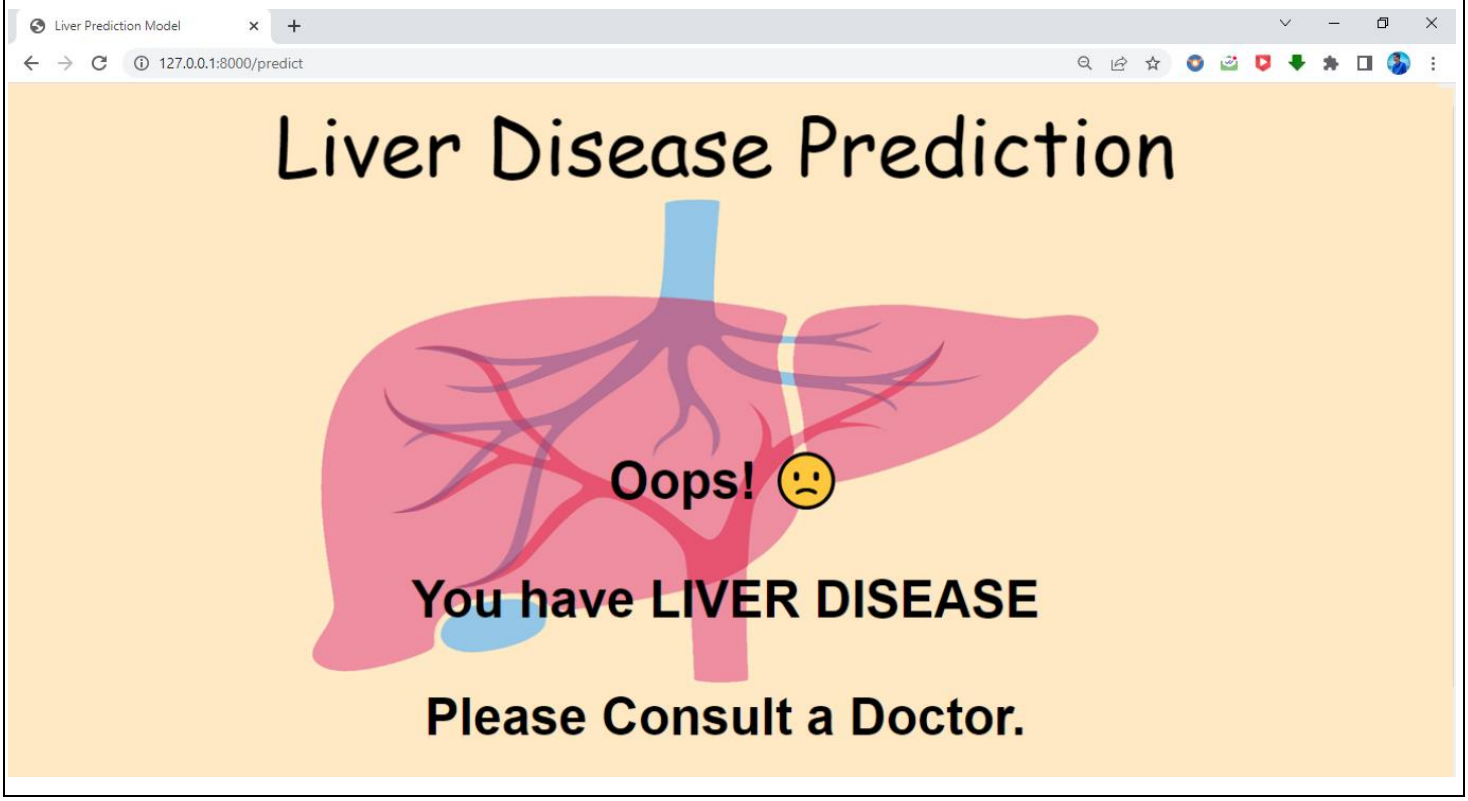
Nochange.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Liver Disease Result</title>
</head>
<body>
<div class="container">
<form action="/data_predict" method="post">
<h2 class='container-heading'><span class="heading_font">Liver Disease Prediction</span></h2>
```

```
<br><br><br><br><br><br><br>
<!-- Result -->
<div class="results">
<h1><span class='danger'>Oops! ⚠<br><br>You have LIVER
DISEASE <br><br>Please Consult a Doctor.</span></h1>
<style>
/* Background Image */
body
{
background-
image:url("https://raw.githubusercontent.com/SagarDhandare/Li
ver-Disease-Prediction-Project/main/Images/Liver.jpg");
height: 100%;
/* Center and scale the image nicely */
background-position: center;
background-repeat: no-repeat;
background-size: 100% 100%;

}
/* Color */
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 */
.container-heading{
```

```
margin: 0;
}
.heading_font{
color: #black;
font-family: 'Pacifico', cursive;
font-size: 50px;
font-weight: normal;
}
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
</style>
</body>
</html>
```

<u>Output:</u>



Application :

- Diagnosis: Liver patients analysis can be used to diagnose liver diseases such as cirrhosis, hepatitis, and Livercancer,by analysing various biomarkers such as liver enyzmes, bilirubin, and albumin, doctors can determine the health of the liver and diagnose any underlying diseases.

- Treatment: Liver patients analysis can also to monitor the effectiveness of treatmentsfor liver diseases. By regulary analysing liver function tests and other biomarkers, doctors can achieve optimal results

Conclusion :

- Since the liver disease is not easy to diagnose, given thedelicate nature of its signs, this research is in determining the algorithms that have better accuracy in predicting this dreadful disease.
- Once the dataset is selected, the preprocessing step is conducted by replacing the missing values and balancing the dataset.

- After that, using R, five different supervised learning methods are applied (i.e., SVM, Naïve Bayes, K-NN, LDA, and CART), and the accuracy with confusion matrix metrics are recorded.

- In this study, the autoencoder with 3-layers achieved an accuracy of 92.1%, slightly higher than K-NN due to its ability to ascertain overlapping features better than conventional K-NNs. Most of the algorithms are more than the acceptable level of accuracy, which is 75%.

- The results from this study would be able to assist health professionals and relevant stakeholders in the early detection of liver disease.

Future scope

- In this paper ,we proposed and built a machine learning based on a hybrid classifier to be used as a classification model for liver diseases diagnosis to improve performance and experrts to identify the chances of disease and conscious orescription of further treatment healthcare and examination .

- In future work, the use of fast datasets technique like apache hadoop or spark can be incorporated with this technique. In addition to this, we can use distributed refined algorithm like forest tree implement in apache hadoop to increase scalability and efficieny.

Appendix:

Jupyter File

Mliestone 2:

In [1]:
```
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
```

In [2]:
```
data=pd.read_csv("indian_liver_patient.csv")
display(data.head())
display(data.tail())
```

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

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albu
578	60	Male	0.5	0.1	500	20	34	5.9	1.6	
579	40	Male	0.6	0.1	98	35	31	6.0	3.2	
580	52	Male	0.8	0.2	245	48	49	6.4	3.2	
581	31	Male	1.3	0.5	184	29	32	6.8	3.4	
582	38	Male	1.0	0.3	216	21	24	7.3	4.4	

In [3]:
```
data.shape
```

Out[3]:
```
(583, 11)
```

In [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
```

In [5]:
```
data.isnull().any()
```

Out[5]:
```
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
```

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
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
```

In [7]:
```
data['Albumin_and_Globulin_Ratio'].fillna(data['Albumin_and_Globulin_Ratio'].mode()[0],inplace=True)
data.isnull().sum()
```

Out[7]:
```
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 [8]:
```
from sklearn.preprocessing import LabelEncoder
lc = LabelEncoder()
data['Gender']=lc.fit_transform(data['Gender'])
```

In [9]:
```
data.describe()
```

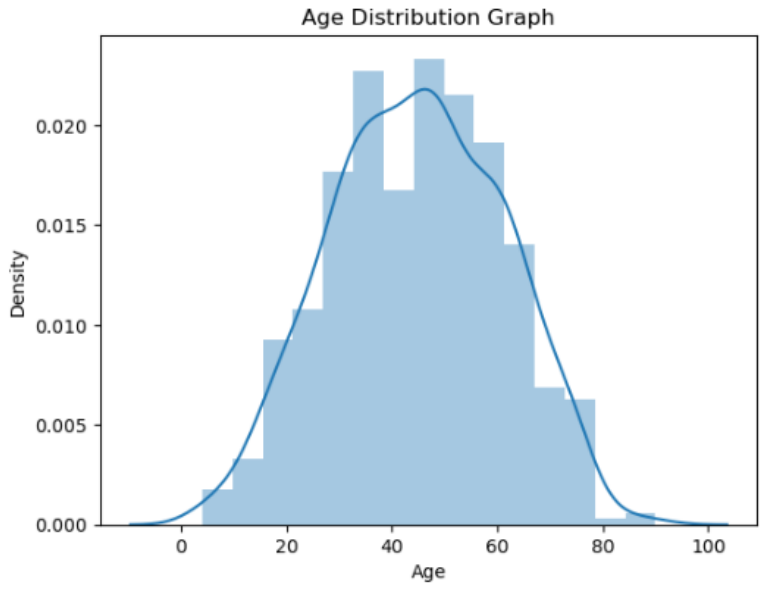
Out[9]:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	
count	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	58
mean	44.746141	0.756432	3.298799	1.486106	290.576329	80.713551	109.910806	6.483190	
std	16.189833	0.429603	6.209522	2.808498	242.937989	182.620356	288.918529	1.085451	
min	4.000000	0.000000	0.400000	0.100000	63.000000	10.000000	10.000000	2.700000	
25%	33.000000	1.000000	0.800000	0.200000	175.500000	23.000000	25.000000	5.800000	

Milestone 3:

In [10]:
```
sns.distplot(data['Age'])
plt.title('Age Distribution Graph')
plt.show()
```

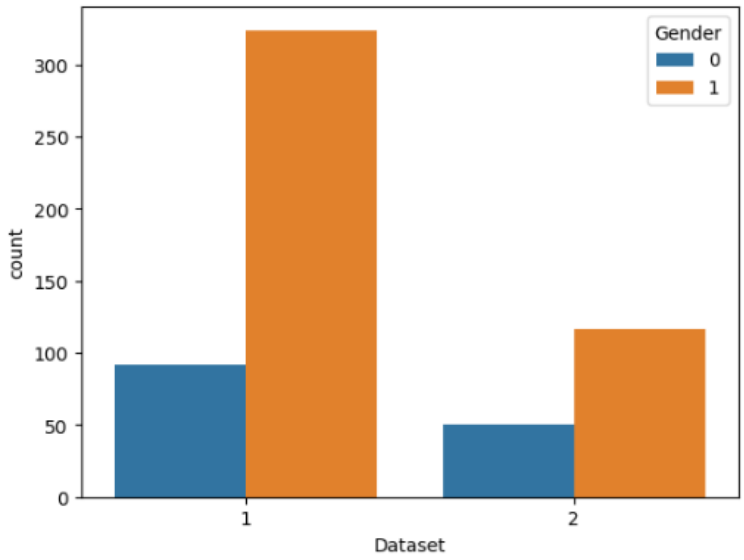
```
C:\Users\DEVIL\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function an
d will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [11]:
```
sns.countplot(data['Dataset'], hue=data['Gender'])
```

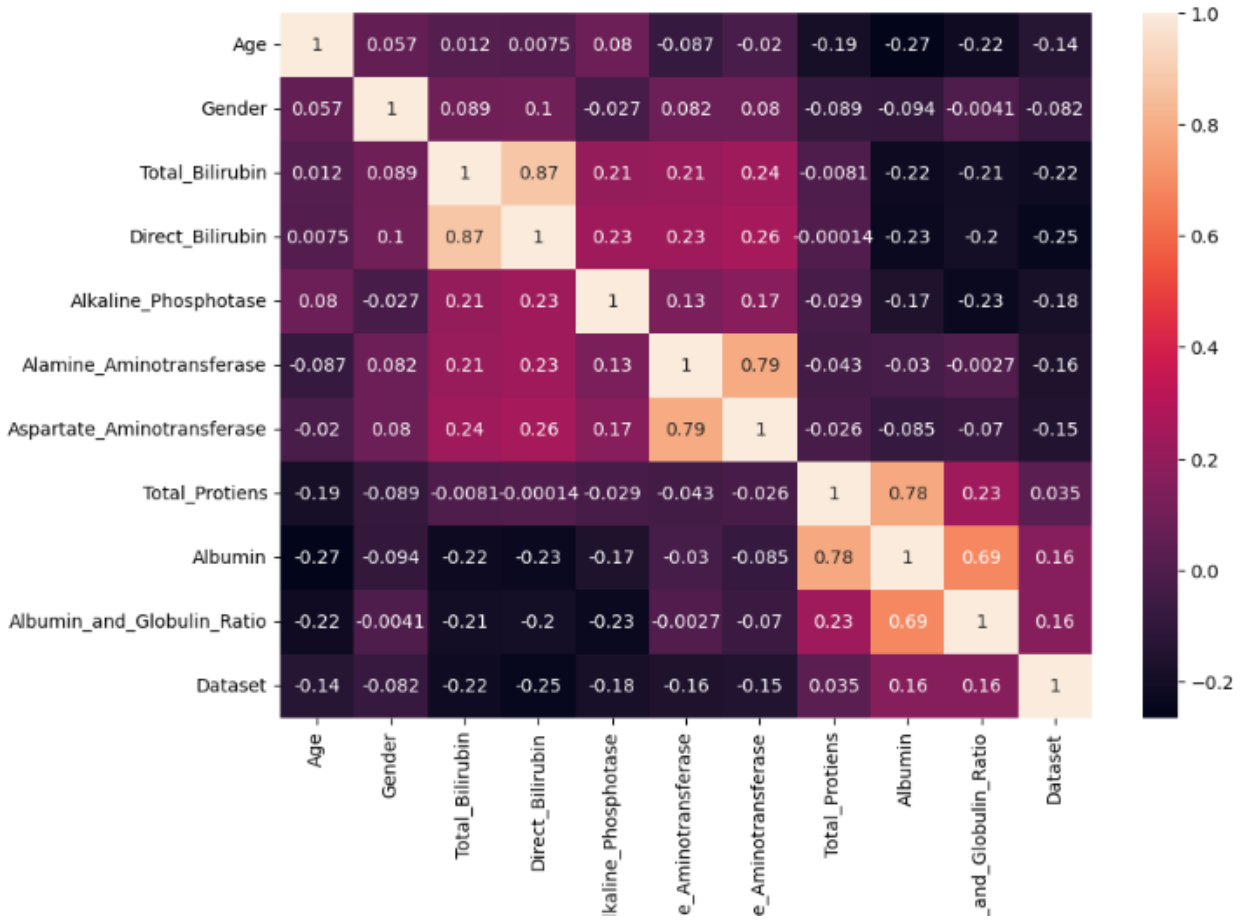
```
C:\Users\DEVIL\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword a
rg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit ke
yword will result in an error or misinterpretation.
  warnings.warn(
```

Out[11]:
```
<AxesSubplot:xlabel='Dataset', ylabel='count'>
```



In [12]:
```
plt.figure(figsize=(10,7))
sns.heatmap(data.corr(),annot=True)
```

Out[12]: <AxesSubplot:>



In [13]:
```
X=data.iloc[:,:-1]
y=data.Dataset
```

In [14]:
```
from sklearn.preprocessing import scale
X_scaled=pd.DataFrame (scale(X), columns=X.columns)
```

In [15]:
```
X_scaled.head()
```

Out[15]:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin
0	1.252098	-1.762281	-0.418878	-0.493964	-0.426715	-0.354665	-0.318393	0.292120	0.198969
1	1.066637	0.567446	1.225171	1.430423	1.682629	-0.091599	-0.034333	0.937566	0.073157
2	1.066637	0.567446	0.644919	0.931508	0.821588	-0.113522	-0.145186	0.476533	0.198969
3	0.819356	0.567446	-0.370523	-0.387054	-0.447314	-0.365626	-0.311465	0.292120	0.324781
4	1.684839	0.567446	0.096902	0.183135	-0.393756	-0.294379	-0.176363	0.753153	-0.933340

In [16]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled,y, test_size=0.2,random_state=12)
```

In [17]:
```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
```

In [18]:
```
y_train.value_counts()
```

Out[18]:
```
1    328
2    138
Name: Dataset, dtype: int64
```

In [19]:
```
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

In [20]:
```
y_train_smote.value_counts()
```

Out[20]:
```
1    328
2    328
Name: Dataset, dtype: int64
```

MILESTONE 4:

In [21]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
model1=RandomForestClassifier()
model1.fit(X_train_smote, y_train_smote)
y_predict = model1.predict(X_test)
rfc1=accuracy_score(y_test,y_predict)
rfc1
pd.crosstab(y_test, y_predict)
print(classification_report (y_test, y_predict))
```

```
              precision    recall  f1-score   support

           1       0.82      0.73      0.77        88
           2       0.38      0.52      0.44        29

    accuracy                           0.68       117
   macro avg       0.60      0.62      0.61       117
weighted avg       0.71      0.68      0.69       117
```

In [22]:
```
from sklearn.tree import DecisionTreeClassifier
model4 = DecisionTreeClassifier()
model4.fit(X_train_smote, y_train_smote)
y_predict=model4.predict(X_test)
dtc1=accuracy_score(y_test,y_predict)
dtc1
pd.crosstab(y_test,y_predict)
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           1       0.77      0.69      0.73        88
           2       0.29      0.38      0.33        29

    accuracy                           0.62       117
   macro avg       0.53      0.54      0.53       117
weighted avg       0.65      0.62      0.63       117
```

In [23]:
```
from sklearn.neighbors import KNeighborsClassifier
model2=KNeighborsClassifier()
model2.fit(X_train_smote, y_train_smote)
y_predict = model2.predict(X_test)
knn1=(accuracy_score (y_test, y_predict))
knn1
pd.crosstab(y_test,y_predict)
print(classification_report (y_test, y_predict))
```

```
              precision    recall  f1-score   support

           1       0.83      0.56      0.67        88
           2       0.33      0.66      0.44        29

    accuracy                           0.58       117
   macro avg       0.58      0.61      0.55       117
weighted avg       0.71      0.58      0.61       117
```

```
C:\Users\DEVIL\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction func
tions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, th
is behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is taken will b
e eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [24]:
```
from sklearn.linear_model import LogisticRegression
model5=LogisticRegression()
model5.fit(X_train_smote, y_train_smote)
y_predict=model5.predict(X_test)
logi1=accuracy_score(y_test, y_predict)
logi1
pd.crosstab(y_test,y_predict)
print(classification_report (y_test, y_predict))
```

```
              precision    recall  f1-score   support

           1       0.94      0.51      0.66        88
           2       0.38      0.90      0.53        29

    accuracy                           0.61       117
   macro avg       0.66      0.70      0.60       117
weighted avg       0.80      0.61      0.63       117
```

In [25]:
```
import tensorflow.keras
from  tensorflow.keras.models import Sequential
from  tensorflow.keras.layers import Dense
```

In [26]:
```
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'])
model_history=classifier.fit(X_train,y_train,batch_size=100,validation_split=0.2,epochs=100)
```

```
Epoch 1/100
4/4 [==============================] - 6s 292ms/step - loss: 1.0012 - accuracy: 0.0027 - val_loss: 0.8586 - val_accuracy: 0.063
8
Epoch 2/100
4/4 [==============================] - 0s 42ms/step - loss: 0.7599 - accuracy: 0.2339 - val_loss: 0.6218 - val_accuracy: 0.5319
Epoch 3/100
4/4 [==============================] - 0s 39ms/step - loss: 0.5569 - accuracy: 0.6478 - val_loss: 0.4276 - val_accuracy: 0.6809
Epoch 4/100
4/4 [==============================] - 0s 38ms/step - loss: 0.3915 - accuracy: 0.7070 - val_loss: 0.2566 - val_accuracy: 0.6809
Epoch 5/100
4/4 [==============================] - 0s 40ms/step - loss: 0.2408 - accuracy: 0.7097 - val_loss: 0.0926 - val_accuracy: 0.6809
Epoch 6/100
4/4 [==============================] - 0s 39ms/step - loss: 0.0933 - accuracy: 0.7097 - val_loss: -0.0684 - val_accuracy: 0.680
9
Epoch 7/100
4/4 [==============================] - 0s 38ms/step - loss: -0.0543 - accuracy: 0.7097 - val_loss: -0.2299 - val_accuracy: 0.68
09
Epoch 8/100
4/4 [==============================] - 0s 40ms/step - loss: -0.1996 - accuracy: 0.7097 - val_loss: -0.3923 - val_accuracy: 0.68
09
Epoch 9/100
4/4 [==============================] - 0s 40ms/step - loss: -0.3512 - accuracy: 0.7097 - val_loss: -0.5588 - val_accuracy: 0.68
09
Epoch 10/100
4/4 [==============================] - 0s 37ms/step - loss: -0.5028 - accuracy: 0.7097 - val_loss: -0.7356 - val_accuracy: 0.68
09
Epoch 11/100
4/4 [==============================] - 0s 41ms/step - loss: -0.6673 - accuracy: 0.7097 - val_loss: -0.9190 - val_accuracy: 0.68
09
Epoch 12/100
4/4 [==============================] - 0s 40ms/step - loss: -0.8241 - accuracy: 0.7097 - val_loss: -1.1146 - val_accuracy: 0.68
09
Epoch 13/100
4/4 [==============================] - 0s 45ms/step - loss: -1.0050 - accuracy: 0.7097 - val_loss: -1.3224 - val_accuracy: 0.68
```

```
Epoch 14/100

4/4 [==============================] - 0s 36ms/step - loss: -790572.3125 - accuracy: 0.7016 - val_loss: -712313.8125 - val_accuracy: 0.7234

Epoch 15/100

4/4 [==============================] - 0s 25ms/step - loss: -801802.5000 - accuracy: 0.7016 - val_loss: -721993.8125 - val_accuracy: 0.7234

Epoch 16/100

4/4 [==============================] - 0s 30ms/step - loss: -812847.3750 - accuracy: 0.7016 - val_loss: -731739.5625 - val_accuracy: 0.7234

Epoch 17/100

4/4 [==============================] - 0s 42ms/step - loss: -823909.3125 - accuracy: 0.7016 - val_loss: -741573.6875 - val_accuracy: 0.7234

Epoch 18/100

4/4 [==============================] - 0s 43ms/step - loss: -834615.9375 - accuracy: 0.7016 - val_loss: -751650.8125 - val_accuracy: 0.7234

Epoch 19/100

4/4 [==============================] - 0s 20ms/step - loss: -846041.3125 - accuracy: 0.7016 - val_loss: -761723.0000 - val_accuracy: 0.7234

Epoch 20/100

4/4 [==============================] - 0s 27ms/step - loss: -857278.9375 - accuracy: 0.7016 - val_loss -771933.6875 - val_accuracy: 0.7234

Epoch 21/100

4/4 [==============================] - 0s 20ms/step - loss: -868970.2500 - accuracy: 0.7016 - val_loss: -782164.2500 - val_accuracy: 0.7234

Epoch 22/100

4/4 [==============================] - 0s 33ms/step - loss: -880613.0000 - accuracy: 0.7016 - val_loss: -792485.2500 - val_accuracy: 0.7234

Epoch 23/100

4/4 [==============================] - 0s 67ms/step - loss: -891858.9375 - accuracy: 0.7016 - val_loss
```

:

-803042.0625 - val_accuracy: 0.7234

Epoch 24/100

4/4 [==========] - 0s 57ms/step - loss: -903902.8750 - accuracy: 0.7016 - val_loss: -813523.7500 - val_accuracy: 0.7234

Epoch 25/100

4/4 [==========] - 0s 56ms/step - loss: -915529.8750 - accuracy: 0.7016 - val_loss: -824129.3750 - val_accuracy: 0.7234

Epoch 26/100

4/4 [==========] - 0s 55ms/step - loss: -927790.4375 - accuracy: 0.7016 - val_loss: -834718.0625 - val_accuracy: 0.7234

Epoch 27/100

4/4 [==========] - 0s 53ms/step - loss: -939390.9375 - accuracy: 0.7016 - val_loss: -845578.3750 - val_accuracy: 0.7234

Epoch 28/100

4/4 [==========] - 0s 20ms/step - loss: -951746.6875 - accuracy: 0.7016 - val_loss: -856461.6250 - val_accuracy: 0.7234

Epoch 29/100

4/4 [==========] - 0s 13ms/step - loss: -964017.0625 - accuracy: 0.7016 - val_loss: -867335.2500 - val_accuracy: 0.7234

Epoch 30/100

4/4 [==========] - 0s 14ms/step - loss: -976092.1875 - accuracy: 0.7016 - val_loss: -878283.5625 - val_accuracy: 0.7234

Epoch 31/100

4/4 [==========] - 0s 18ms/step - loss: -988350.1875 - accuracy: 0.7016 - val_loss: -889272.2500 - val_accuracy: 0.7234

Epoch 32/100

4/4 [==========] - 0s 16ms/step - loss: -1000706.9375 - accuracy: 0.7016 - val_loss: -900362.7500 - val_accuracy: 0.7234

Epoch 33/100

4/4 [==========] - 0s 19ms/step - loss: -1013206.9375 - accuracy: 0.7016 - val_loss: -911565.0000 - val_accuracy: 0.7234

Epoch 34/100

4/4 [==========] - 0s 21ms/step - loss: -1026137.4375 - accuracy: 0.7016 - val_loss: -922823.0000 - val_accuracy: 0.7234

Epoch 35/100

4/4 [==========] - 0s 19ms/step - loss: -1037931.1875 - accuracy: 0.7016 - val_loss: -934481.6250 - val_accuracy: 0.7234

Epoch 36/100

4/4 [==========] - 0s 14ms/step - loss: -1051233.7500 - accuracy: 0.7016 - val_loss: -945996.4375 - val_accuracy: 0.7234

Epoch 37/100

4/4 [==========] - 0s 19ms/step - loss: -1064053.8750 - accuracy: 0.7016 - val_loss: -957543.7500 - val_accuracy: 0.7234

Epoch 38/100

4/4 [==========] - 0s 19ms/step - loss: -1077281.3750 - accuracy: 0.7016 - val_loss: -969030.5625 - val_accuracy: 0.7234

Epoch 39/100

4/4 [==========] - 0s 19ms/step - loss: -1090125.3750 - accuracy: 0.7016 - val_loss: -980694.3125 - val_accuracy: 0.7234

Epoch 40/100

4/4 [==========] - 0s 25ms/step - loss: -1103089.1250 - accuracy: 0.7016 - val_loss: -992526.8125 - val_accuracy: 0.7234

Epoch 41/100

4/4 [==========] - 0s 19ms/step - loss: -1116665.6250 - accuracy: 0.7016 - val_loss: -1004365.9375 - val_accuracy: 0.7234

Epoch 42/100

```
4/4 [==============================] - 0s 14ms/step - loss: -1129659.7500 - accuracy: 0.7016 - val_loss: -1016444.1875 - val_accuracy: 0.7234

Epoch 43/100

4/4 [==============================] - 0s 13ms/step - loss: -1143343.8750 - accuracy: 0.7016 - val_loss: -1028518.8125 - val_accuracy: 0.7234

Epoch 44/100

4/4 [==============================] - 0s 20ms/step - loss: -1156914.8750 - accuracy: 0.7016 - val_loss: -1040690.8750 - val_accuracy: 0.7234

Epoch 45/100

4/4 [==============================] - 0s 17ms/step - loss: -1170839.7500 - accuracy: 0.7016 - val_loss: -1052845.2500 - val_accuracy: 0.7234

Epoch 46/100

4/4 [==============================] - 0s 20ms/step - loss: -1184115.1250 - accuracy: 0.7016 - val_loss: -1065299.1250 - val_accuracy: 0.7234

Epoch 47/100

4/4 [==============================] - 0s 19ms/step - loss: -1198001.7500 - accuracy: 0.7016 - val_loss: -1077800.1250 - val_accuracy: 0.7234

Epoch 48/100

4/4 [==============================] - 0s 19ms/step - loss: -1212449.0000 - accuracy: 0.7016 - val_loss: -1090211.3750 - val_accuracy: 0.7234

Epoch 49/100

4/4 [==============================] - 0s 19ms/step - loss: -1226363.3750 - accuracy: 0.7016 - val_loss: -1102746.2500 - val_accuracy: 0.7234

Epoch 50/100

4/4 [==============================] - 0s 20ms/step - loss: -1240148.2500 - accuracy: 0.7016 - val_loss: -1115452.1250 - val_accuracy: 0.7234

Epoch 51/100

4/4 [==============================] - 0s 15ms/step - loss: -1254514.6250 - accuracy: 0.7016 - val_loss: -1128213.5000 - val_accuracy: 0.7234

Epoch 52/100

4/4 [==============================] - 0s 19ms/step - loss: -1268405.0000 - accuracy: 0.7016 - val_loss: -1141175.3750 - val_accuracy: 0.7234

Epoch 53/100
```

```
4/4 [==============================] - 0s 13ms/step - loss: -1283378.7500 - accuracy: 0.7016 - val_loss: -1153975.8750 - val_accuracy: 0.7234

Epoch 54/100

4/4 [==============================] - 0s 13ms/step - loss: -1297842.6250 - accuracy: 0.7016 - val_loss: -1166885.2500 - val_accuracy: 0.7234

Epoch 55/100

4/4 [==============================] - 0s 19ms/step - loss: -1313219.6250 - accuracy: 0.7016 - val_loss: -1179737.3750 - val_accuracy: 0.7234

Epoch 56/100

4/4 [==============================] - 0s 14ms/step - loss: -1327222.6250 - accuracy: 0.7016 - val_loss: -1193090.7500 - val_accuracy: 0.7234

Epoch 57/100

4/4 [==============================] - 0s 19ms/step - loss: -1341815.2500 - accuracy: 0.7016 - val_loss: -1206648.1250 - val_accuracy: 0.7234

Epoch 58/100

4/4 [==============================] - 0s 19ms/step - loss: -1356829.8750 - accuracy: 0.7016 - val_loss: -1220226.7500 - val_accuracy: 0.7234

Epoch 59/100

4/4 [==============================] - 0s 14ms/step - loss: -1372226.3750 - accuracy: 0.7016 - val_loss: -1233714.5000 - val_accuracy: 0.7234

Epoch 60/100

4/4 [==============================] - 0s 13ms/step - loss: -1387510.3750 - accuracy: 0.7016 - val_loss: -1247228.3750 - val_accuracy: 0.7234

Epoch 61/100

4/4 [==============================] - 0s 14ms/step - loss: -1402555.0000 - accuracy: 0.7016 - val_loss: -1260871.1250 - val_accuracy: 0.7234

Epoch 62/100

4/4 [==============================] - 0s 23ms/step - loss: -1417701.6250 - accuracy: 0.7016 - val_loss: -1274690.7500 - val_accuracy: 0.7234

Epoch 63/100

4/4 [==============================] - 0s 14ms/step - loss: -1433544.1250 - accuracy: 0.7016 - val_loss: -1288476.2500 - val_accuracy: 0.7234

Epoch 64/100
```

```
4/4 [==============================] - 0s 19ms/step - loss: -1449153.0000 - accuracy: 0.7016 - val_loss: -1302399.7500 - val_accuracy: 0.7234

Epoch 65/100

4/4 [==============================] - 0s 13ms/step - loss: -1464224.3750 - accuracy: 0.7016 - val_loss: -1316549.5000 - val_accuracy: 0.7234

Epoch 66/100

4/4 [==============================] - 0s 12ms/step - loss: -1479801.0000 - accuracy: 0.7016 - val_loss: -1330783.8750 - val_accuracy: 0.7234

Epoch 67/100

4/4 [==============================] - 0s 13ms/step - loss: -1496706.6250 - accuracy: 0.7016 - val_loss: -1344791.1250 - val_accuracy: 0.7234

Epoch 68/100

4/4 [==============================] - 0s 14ms/step - loss: -1512144.8750 - accuracy: 0.7016 - val_loss: -1359089.2500 - val_accuracy: 0.7234

Epoch 69/100

4/4 [==============================] - 0s 14ms/step - loss: -1528065.8750 - accuracy: 0.7016 - val_loss: -1373541.0000 - val_accuracy: 0.7234

Epoch 70/100

4/4 [==============================] - 0s 19ms/step - loss: -1544517.3750 - accuracy: 0.7016 - val_loss: -1388008.6250 - val_accuracy: 0.7234

Epoch 71/100

4/4 [==============================] - 0s 13ms/step - loss: -1561332.0000 - accuracy: 0.7016 - val_loss: -1402453.5000 - val_accuracy: 0.7234

Epoch 72/100

4/4 [==============================] - 0s 19ms/step - loss: -1577190.0000 - accuracy: 0.7016 - val_loss: -1417268.3750 - val_accuracy: 0.7234

Epoch 73/100

4/4 [==============================] - 0s 14ms/step - loss: -1594071.0000 - accuracy: 0.7016 - val_loss: -1432073.5000 - val_accuracy: 0.7234

Epoch 74/100

4/4 [==============================] - 0s 13ms/step - loss: -1610046.1250 - accuracy: 0.7016 - val_loss: -1447196.8750 - val_accuracy: 0.7234

Epoch 75/100
```

4/4 [==========] - 0s 20ms/step - loss: -1627160.6250 - accuracy: 0.7016 - val_loss: -1462295.5000 - val_accuracy: 0.7234

Epoch 76/100

4/4 [==========] - 0s 14ms/step - loss: -1644354.2500 - accuracy: 0.7016 - val_loss: -1477444.3750 - val_accuracy: 0.7234

Epoch 77/100

4/4 [==========] - 0s 19ms/step - loss: -1661460.2500 - accuracy: 0.7016 - val_loss: -1492718.0000 - val_accuracy: 0.7234

Epoch 78/100

4/4 [==========] - 0s 19ms/step - loss: -1679104.5000 - accuracy: 0.7016 - val_loss: -1508000.1250 - val_accuracy: 0.7234

Epoch 79/100

4/4 [==========] - 0s 14ms/step - loss: -1695154.1250 - accuracy: 0.7016 - val_loss: -1523874.7500 - val_accuracy: 0.7234

Epoch 80/100

4/4 [==========] - 0s 19ms/step - loss: -1713723.8750 - accuracy: 0.7016 - val_loss: -1539401.7500 - val_accuracy: 0.7234

Epoch 81/100

4/4 [==========] - 0s 20ms/step - loss: -1730503.8750 - accuracy: 0.7016 - val_loss: -1555277.7500 - val_accuracy: 0.7234

Epoch 82/100

4/4 [==========] - 0s 20ms/step - loss: -1748175.6250 - accuracy: 0.7016 - val_loss: -1571173.1250 - val_accuracy: 0.7234

Epoch 83/100

4/4 [==========] - 0s 22ms/step - loss: -1766677.6250 - accuracy: 0.7016 - val_loss: -1586805.2500 - val_accuracy: 0.7234

Epoch 84/100

4/4 [==========] - 0s 14ms/step - loss: -1784156.8750 - accuracy: 0.7016 - val_loss: -1602683.8750 - val_accuracy: 0.7234

Epoch 85/100

4/4 [==========] - 0s 14ms/step - loss: -1801669.3750 - accuracy: 0.7016 - val_loss: -1618731.8750 - val_accuracy: 0.7234

Epoch 86/100

4/4 [==========] - 0s 20ms/step - loss: -1819357.3750 - accuracy: 0.7016 - val_loss: -1634839.8750 - val_accuracy: 0.7234

Epoch 87/100

4/4 [==========] - 0s 20ms/step - loss: -1838195.5000 - accuracy: 0.7016 - val_loss: -1650720.6250 - val_accuracy: 0.7234

Epoch 88/100

4/4 [==========] - 0s 14ms/step - loss: -1855447.3750 - accuracy: 0.7016 - val_loss: -1667040.0000 - val_accuracy: 0.7234

Epoch 89/100

4/4 [==========] - 0s 20ms/step - loss: -1874518.3750 - accuracy: 0.7016 - val_loss: -1683222.5000 - val_accuracy: 0.7234

Epoch 90/100

4/4 [==========] - 0s 18ms/step - loss: -1891730.3750 - accuracy: 0.7016 - val_loss: -1699956.3750 - val_accuracy: 0.7234

Epoch 91/100

4/4 [==========] - 0s 19ms/step - loss: -1910361.5000 - accuracy: 0.7016 - val_loss: -1716632.1250 - val_accuracy: 0.7234

Epoch 92/100

4/4 [==========] - 0s 20ms/step - loss: -1929668.5000 - accuracy: 0.7016 - val_loss: -1733106.0000 - val_accuracy: 0.7234

Epoch 93/100

4/4 [==========] - 0s 13ms/step - loss: -1948571.8750 - accuracy: 0.7016 - val_loss: -1749683.3750 - val_accuracy: 0.7234

Epoch 94/100

4/4 [==========] - 0s 14ms/step - loss: -1966975.2500 - accuracy: 0.7016 - val_loss: -1766571.7500 - val_accuracy: 0.7234

Epoch 95/100

4/4 [==========] - 0s 14ms/step - loss: -1985814.0000 - accuracy: 0.7016 - val_loss: -1783625.3750 - val_accuracy: 0.7234

Epoch 96/100

4/4 [==========] - 0s 19ms/step - loss: -2004937.6250 - accuracy: 0.7016 - val_loss: -1800817.8750 - val_accuracy: 0.7234

Epoch 97/100

4/4 [==========] - 0s 13ms/step - loss: -2024367.1250 - accuracy: 0.7016 - val_loss: -1818068.7500 - val_accuracy: 0.7234

Epoch 98/100

4/4 [==========] - 0s 14ms/step - loss: -2043511.3750 - accuracy: 0.7016 - val_loss: -1835457.7500 - val_accuracy: 0.7234

Epoch 99/100

4/4 [==========] - 0s 19ms/step - loss: -2062323.6250 - accuracy: 0.7016 - val_loss: -1853025.7500 -

```
0.6809
Epoch 100/100
4/4 [==========] - 0s 35ms/step - loss: -824.5608 - accuracy: 0.7097 - val_loss: -897.3958 - val_accuracy:
0.6809
```

```
In [27]: model4.predict([[50,1,1.2,0.8,150,70,80,7.2,3.4,0.8]])
```

```
C:\Users\DEVIL\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but DecisionT
reeClassifier was fitted with feature names
  warnings.warn(
```

```
Out[27]: array([2], dtype=int64)
```

```
In [28]: model1.predict([[50,1,1.2,0.8,150,70,80,7.2,3.4,0.8]])
```

```
C:\Users\DEVIL\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but RandomFor
estClassifier was fitted with feature names
  warnings.warn(
```

```
Out[28]: array([1], dtype=int64)
```

```
In [29]: classifier.save("liver.h5")
```

```
In [30]: y_pred = classifier.predict(X_test)
```

```
4/4 [==========] - 0s 5ms/step
```

```
In [31]: y_pred
         y_pred = (y_pred > 0.5)
         y_pred
```

```
Out[31]: array([[ True],
                [ True],
                [ True],
                [ True],
                [ True],
                [ True],
                [ True],
                [ True],
                [ True],
                [ True],
                [ True],
                [ True],
                [ True],
                [ True],
                [ True],
```

In [32]:
```
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)
```

In [33]:
```
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')
```

```
1/1 [==============================] - 0s 184ms/step
Prediction: Liver Patient
```

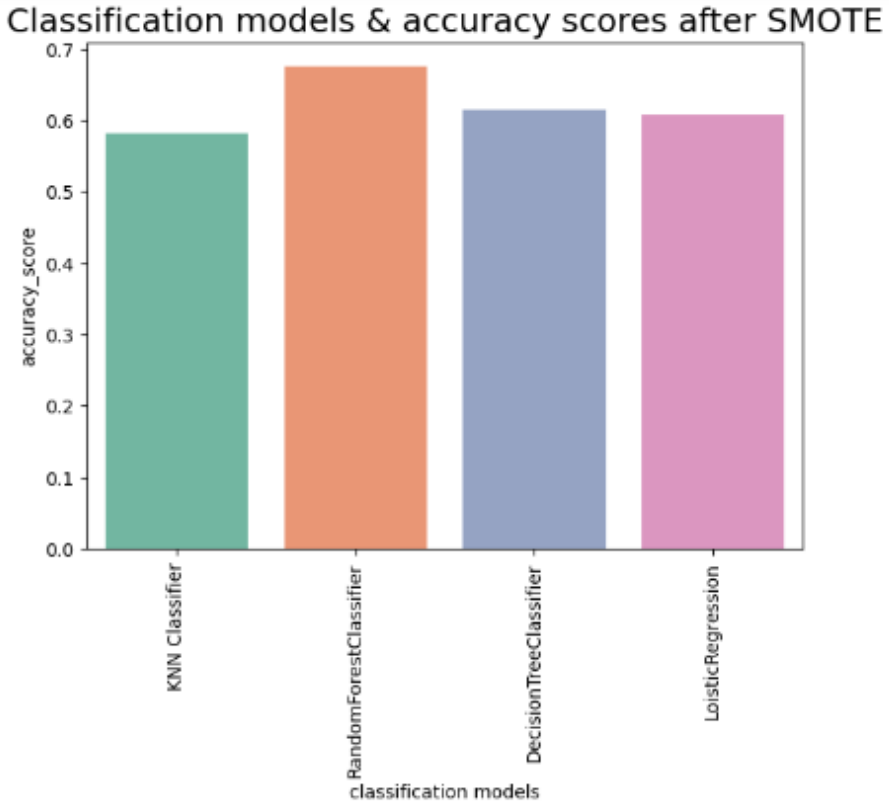
In [34]:
```
acc_smote=[['KNN Classifier',knn1],['RandomForestClassifier',rfc1],['DecisionTreeClassifier',dtc1],['LoisticRegression',logi1]]
Liverpatient_pred=pd.DataFrame(acc_smote,columns=['classification models','accuracy_score'])
Liverpatient_pred
```

Out[34]:

	classification models	accuracy_score
0	KNN Classifier	0.581197
1	RandomForestClassifier	0.675214
2	DecisionTreeClassifier	0.615385
3	LoisticRegression	0.606838

In [35]:
```
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")
```

Out[35]:
```
<AxesSubplot:title={'center':'Classification models & accuracy scores after SMOTE'}, xlabel='classification models', ylabel='ac
curacy_score'>
```



In [36]:
```
from sklearn.ensemble import ExtraTreesClassifier
model=ExtraTreesClassifier()
model.fit(X,y)
```

Out[36]:
```
ExtraTreesClassifier()
```

In [37]:
```
model.feature_importances_
```

Out[37]:
```
array([0.12026859, 0.024727  , 0.10979189, 0.10245196, 0.11654257,
       0.11399954, 0.11870815, 0.09126762, 0.1023186 , 0.09992409])
```

MILESTONE 5:

In [37]:
```
model.feature_importances_
```

Out[37]:
```
array([0.12026859, 0.024727  , 0.10979189, 0.10245196, 0.11654257,
       0.11399954, 0.11870815, 0.09126762, 0.1023186 , 0.09992409])
```

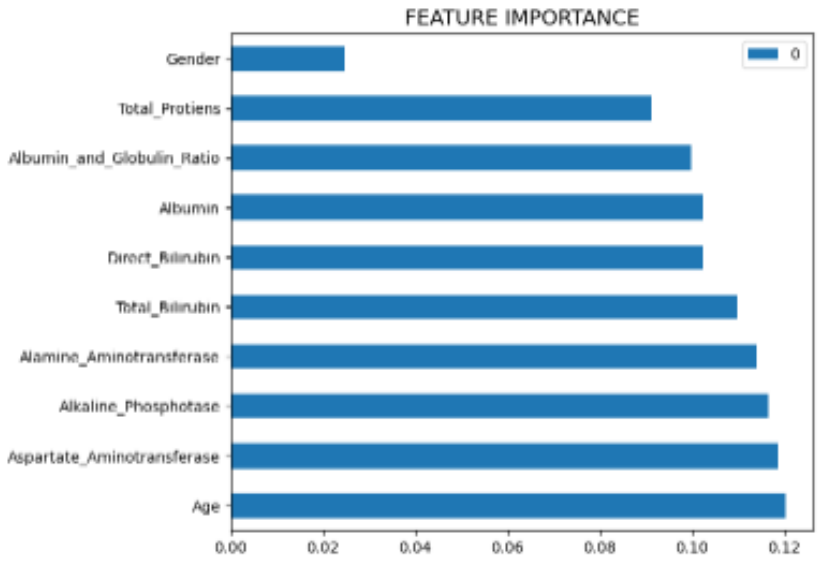
In [38]:
```
dd=pd.DataFrame(model.feature_importances_,index=X.columns).sort_values(0,ascending=False)
dd
```

Out[38]:

	0
Age	0.120269
Aspartate_Aminotransferase	0.118708
Alkaline_Phosphotase	0.116543
Alamine_Aminotransferase	0.114000
Total_Bilirubin	0.109792
Direct_Bilirubin	0.102452
Albumin	0.102319
Albumin_and_Globulin_Ratio	0.099924
Total_Protens	0.091268
Gender	0.024727

In [39]:
```
dd.plot(kind='barh',figsize=(7,6))
plt.title("FEATURE IMPORTANCE",fontsize=14)
```

Out[39]:
```
Text(0.5, 1.0, 'FEATURE IMPORTANCE')
```



In [40]:
```
import joblib
joblib.dump(model1,'ETC.pkl')
```

Out[40]:
```
['ETC.pkl']
```

Milestone 6:

python file (app.p)

```
from flask import Flask,render_template,request
 import numpy as np
 import pickle
 app=Flask(__name__,template_folder = 'template')
 @app.route('/')
 def base():
   return render_template('base.html')
 @app.route('/Home')
 def home():
   return render_template('home.html')
 @app.route('/About')
 def about():
   return render_template('about.html')
 @app.route('/predict')
 def index():
  return render_template("liverprediction.html")


 @app.route('/data_predict',methods = ['POST'])
 def predict():
     age = request.form['age']
     gender=request.form['gender']
```

```
tb=request.form['tb']
    db=request.form['db']
    ap=request.form['ap']
    aa1=request.form['aa1']
    aa2=request.form['aa2']
    tp=request.form['tp']
    a=request.form['a']
    agr=request.form['agr']
    data =
[[float(age),float(gender),float(db),float(ap),float(aa1),float(aa2),float(t
p),float(a),float(agr)]]
    model = pickle.load(open('ETC.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 should consult a doctor.Take
care')
    else:
      return render_template('chance.html', prediction='You dont have a
liver desease problem')

if __name__=='__main__' :
  app.debug= True
  app.run(port=8000)
```

THE END