A Review of Liver Patient Analysis Methods using Machine Learning

Project Based Experiential Learning Program

A Project Report

A Review of Liver Patient Analysis Methods using Machine Learning

Team ID : NM2023TMID17322

Team Leader : QUEENJINNY M

Team members : PRADEEPA S

PRATHESH KUMAR A

PRIYA K

INDEX

A Review of Liver Patient Analysis Methods Using Machine Learning

1. INTRODUCTION

1.1 OVERVIEW

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 analyses 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 utilized in the prediction of liver disease and can be recommended to the user

1.2 Purpose

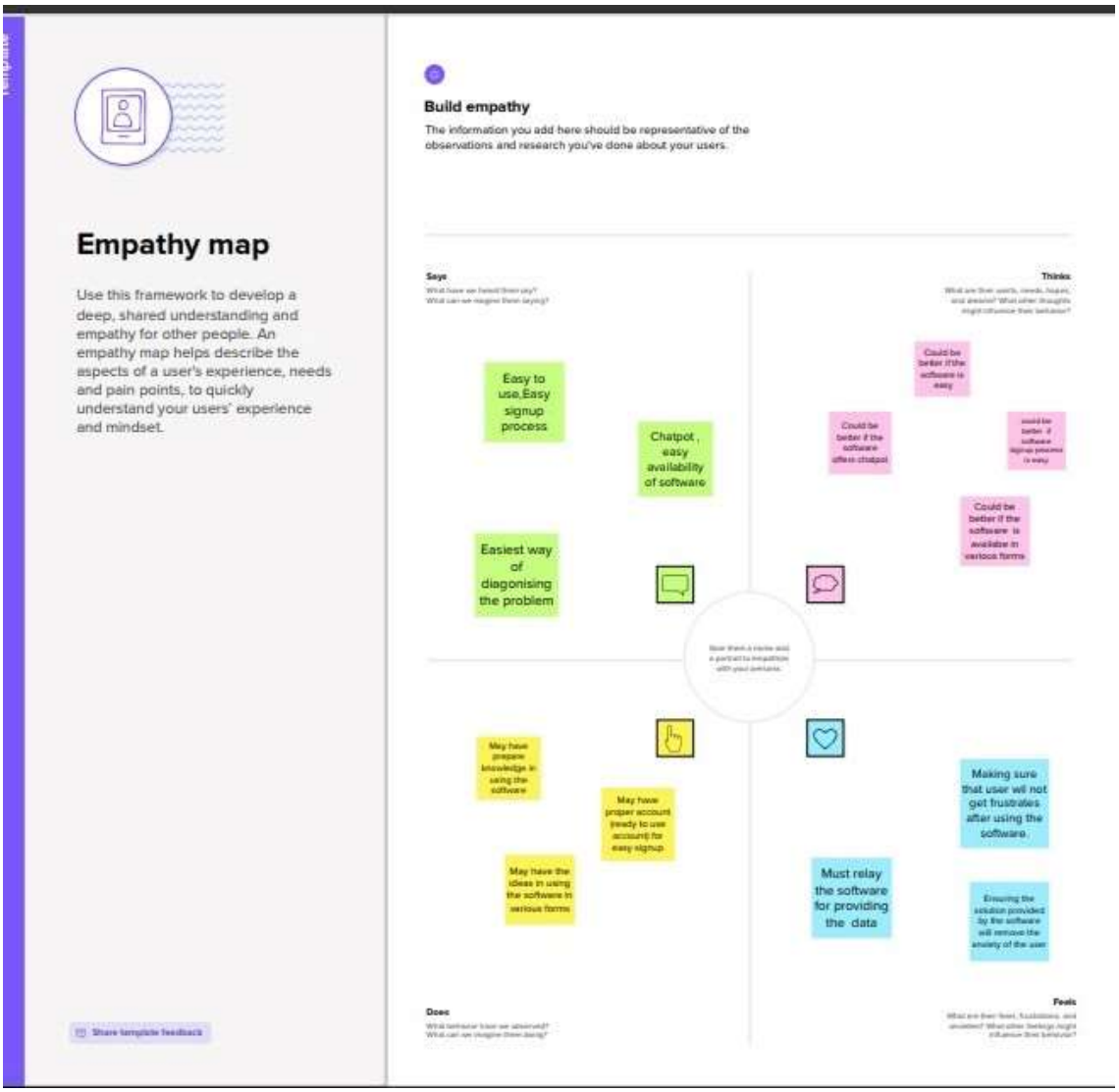
The purpose of a review of liver patient analysis methods using machine learning is to assess the effectiveness of machine learning algorithms in analysing patient data and predicting liver diseases' diagnosis, prognosis, and treatment planning. The review aims to evaluate the strengths and limitations of machine learning algorithms in liver patient analysis and identify potential areas for future research.

The review also aims to provide healthcare providers and researchers with an understanding of the current state of the field of machine learning in liver patient analysis, including the types of machine learning algorithms that are commonly used, the sources of patient data that are analysed, and the accuracy and reliability of machine learning models in predicting liver diseases.

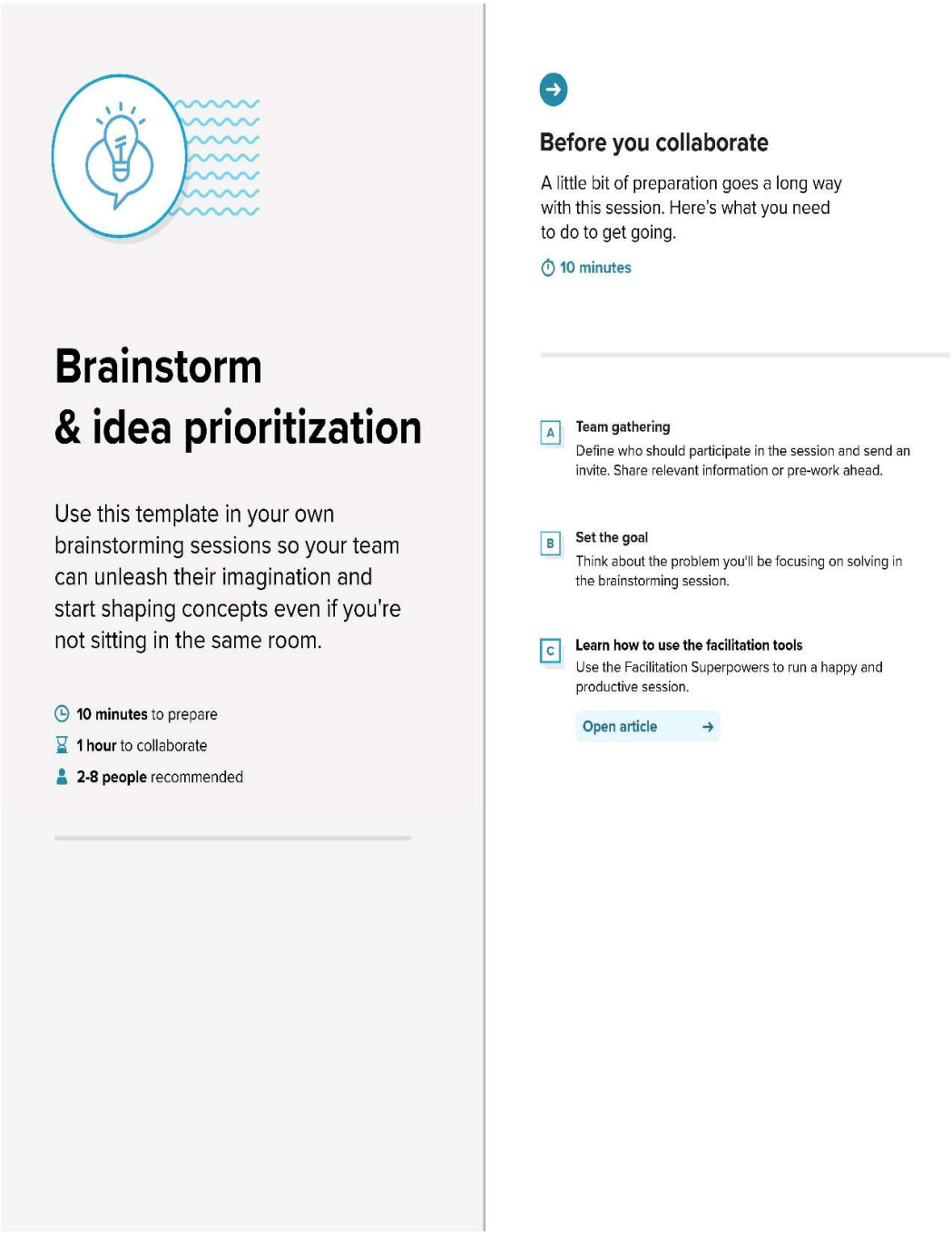
Overall, the purpose of a review of liver patient analysis methods using machine learning is to provide a comprehensive assessment of the potential benefits and challenges associated with the use of machine learning algorithms in liver patient analysis and to identify future research directions.

2. Problem Definition & Design Thinking

2.1 Empathy Map



2.2 Ideation & Brainstorming Map



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article

3

Group ideas

- Create a poster.
- Write an longer documenting proposals for future research grant applications
- Make a PowerPoint presentation.
- Design a model.
- Awareness to public
- Make a provision of funding.
- Use a bigger display board
- Make a timeline.

20 minutes

A comprehensive review of ML techniques used for liver disease diagnosis:

A comprehensive review of ML techniques used for liver disease diagnosis. This review could cover a wide range of ML methods, including supervised, unsupervised, and semi-supervised learning algorithms, and explore their application in liver disease diagnosis. The review could also discuss the advantages and limitations of these techniques and highlight potential areas for future research.

Feature selection is an important step in ML-based liver disease classification, and there are several approaches to identifying relevant features from large datasets.

This review could compare different feature selection techniques, including filter, wrapper, and embedded methods, and evaluate their performance in liver disease classification tasks.

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

A review of ML-based models for liver disease prognosis

Prognosis is an important aspect of liver disease management, and ML-based models can assist in predicting disease progression and patient outcomes.

Review could evaluate different ML-based models, such as decision trees, support vector machines (SVMs), and neural networks, and compare their performance in predicting liver disease prognosis.

A Future directions:

Developing a machine learning-based tool to predict the likelihood of liver disease recurrence in patients who have undergone treatment.

Investigating the potential for machine learning to identify novel biomarkers for liver disease diagnosis and prognosis

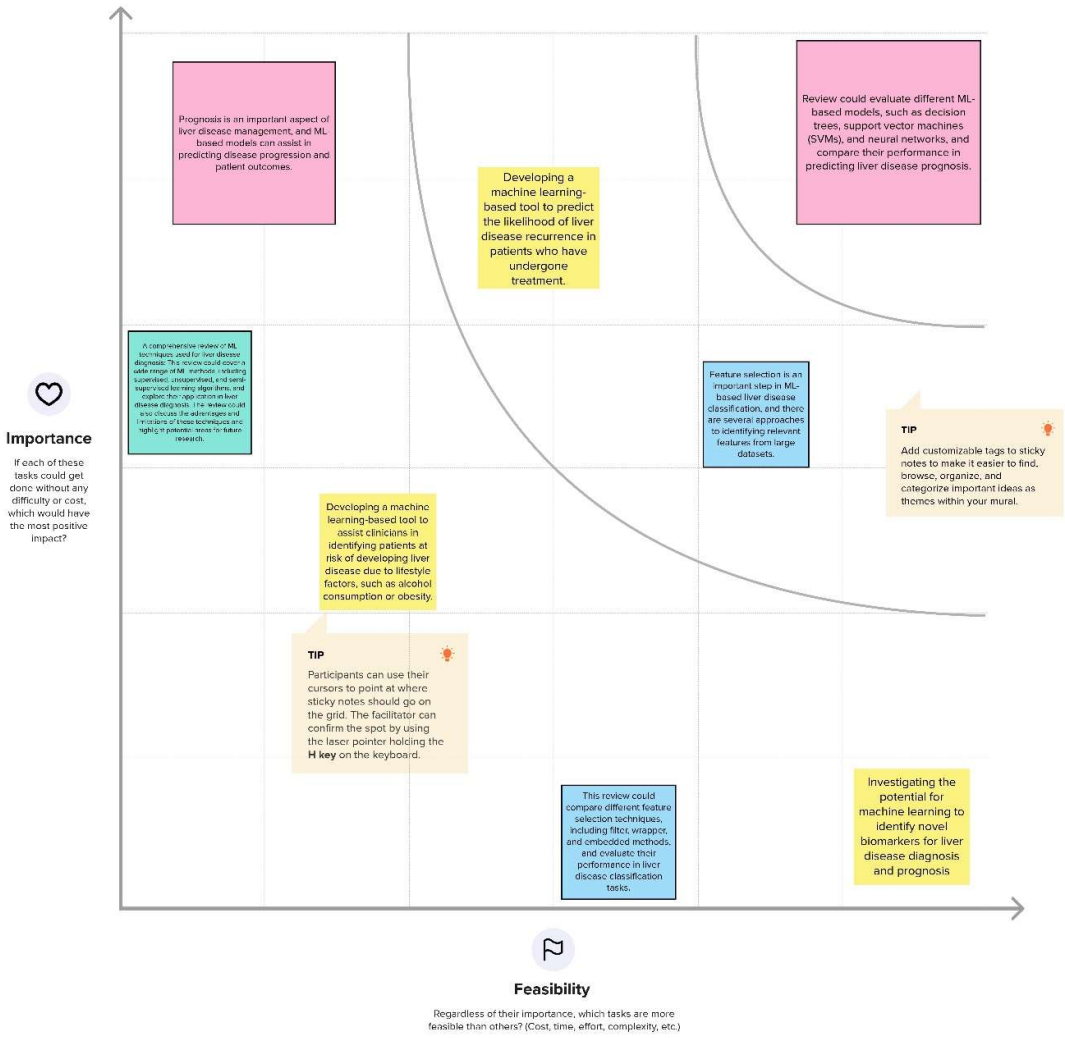
Developing a machine learning based tool to assist clinicians in identifying patients at risk of developing liver disease due to lifestyle factors, such as alcohol consumption or obesity.

4

Prioritize

Prioritization of patients on the waiting list (WL) for OLT is still a critical issue. Numerous models have been developed to predict mortality before and after OLT.

20 minutes



After you collaborate

This study is to identify that whether the patient has liver disease or not. Some of the parameter are used for predicting the liver disease and compare the performance of the various decision tree techniques.

Quick add-ons

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

B **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

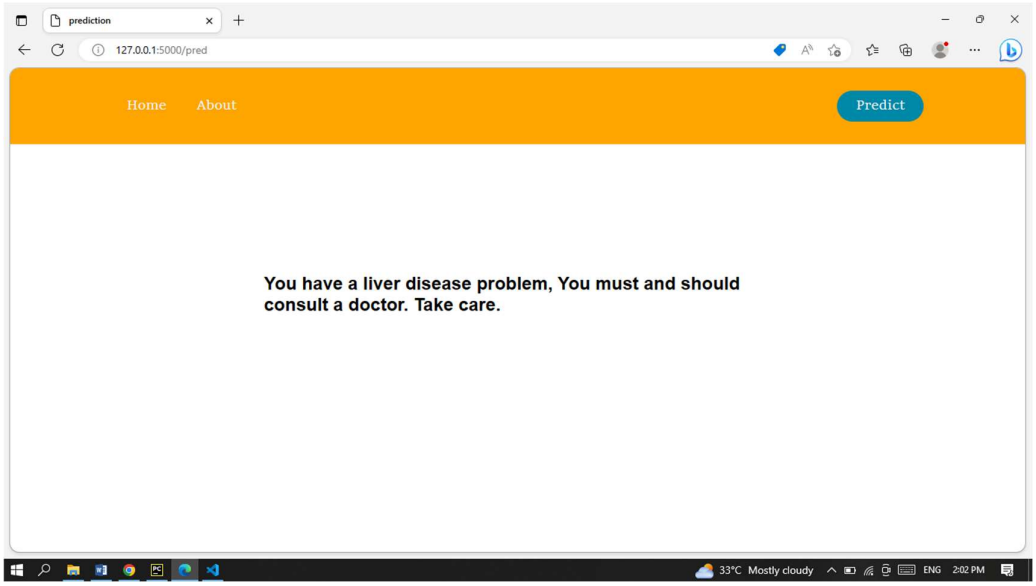
Strategy blueprint
Define the components of a new idea or strategy.
Open the template →

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


prediction
127.0.0.1:5000/pred
Home
About
Predict
You have a liver disease problem, You must and should consult a doctor. Take care.
33°C Mostly cloudy
ENG
2:02 PM

4. ADVANTAGES & DISADVANTAGES

Advantages

1. Improved Accuracy: Machine learning algorithms can identify patterns and correlations in large amounts of data that may not be easily discernible by humans. As a result, machine learning algorithms can improve the accuracy of liver patient analysis methods.
2. Speed: Machine learning algorithms can analyse large amounts of data quickly, which can be particularly useful when dealing with large data sets. This can save time and resources when compared to traditional analysis methods.
3. Personalized Treatment: Machine learning algorithms can be used to develop personalized treatment plans for liver patients. By analysing individual patient data, machine learning algorithms can help doctors tailor treatment plans to specific patient needs.
4. Early Detection: Machine learning algorithms can be used to detect liver diseases at an early stage, allowing doctors to provide early intervention and improve the patient's prognosis.

Disadvantages:

1. Limited Data: Machine learning algorithms rely on large amounts of data to function effectively. If there is a limited amount of data available, the accuracy of the algorithm may be compromised.
2. Bias: Machine learning algorithms can be biased if the data used to train the algorithm is biased. This can lead to inaccurate results and may have negative consequences for liver patients.
3. Complexity: Machine learning algorithms can be complex, and it can be difficult for non-experts to understand how they work. This can make it difficult for doctors to use these methods in clinical practice.
4. Data Privacy: Machine learning algorithms rely on patient data to function effectively. As a result, there may be concerns about data privacy and patient confidentiality. It is important to ensure that patient data is protected when using these methods.

5. APPLICATION

Application of Job Prediction

Job prediction is not directly applicable to liver patient analysis methods using machine learning. However, the use of machine learning algorithms in liver patient analysis can help healthcare providers identify patients who may be at higher risk of developing liver diseases, and as a result, they may need to modify their lifestyle or take preventive measures.

Machine learning algorithms can also help healthcare providers predict the prognosis of patients with liver diseases, which can help them develop more personalized treatment plans. By analysing patient data such as age, gender, medical history, and test results, machine learning algorithms can help healthcare providers predict how patients with liver diseases will respond to different treatments.

Overall, the application of machine learning in liver patient analysis can help healthcare providers make more informed decisions and improve patient outcomes. It is important to note that the use of machine learning algorithms should be complemented with clinical expertise and human judgment to ensure that patients receive the best possible care.

6. CONCLUSION

Conclusion:

The use of machine learning algorithms in liver patient analysis has the potential to improve the accuracy of diagnosis, prognosis, and treatment planning for liver diseases. Machine learning algorithms can analyze large amounts of patient data quickly and identify patterns and correlations that may not be easily discernible by humans. As a result, healthcare providers can make more informed decisions and provide more personalized care to liver patients.

7. FUTURE SCOPE

Future Scope:

The future scope of liver patient analysis methods using machine learning is vast and promising. Here are some potential areas of future research:

1. Developing more accurate and reliable machine learning models: Researchers can continue to refine machine learning algorithms to improve their accuracy and reliability in liver patient analysis. This can include developing new machine learning models or improving existing ones.
2. Integration of multiple data sources: Researchers can explore the integration of multiple data sources, including genetic data, lifestyle data, and environmental data, to improve the accuracy of liver patient analysis. This can lead to more personalized treatment plans and better patient outcomes.
3. Development of decision support systems: Machine learning algorithms can be integrated into decision support systems that can help healthcare providers make more informed decisions about liver patient analysis and treatment planning.
4. Real-time monitoring: Machine learning algorithms can be used to monitor liver patients in real-time, providing healthcare providers with up-to-date information on patient health and enabling timely interventions when necessary.
5. Integration with other technologies: Machine learning algorithms can be integrated with other emerging technologies, such as wearable devices and telemedicine, to improve the accuracy and efficiency of liver patient analysis and treatment.

In conclusion, the future of liver patient analysis using machine learning is bright and holds a lot of promise for improving patient outcomes. Further research and development in this area will be crucial to fully realize the potential of machine learning algorithms in liver patient analysis.

8. APPENDIX

8.1 Source Code

1.app.py(Source Code)

```
from flask import Flask, render_template, request
import pickle
import joblib
import numpy as np
from sklearn.preprocessing import scale

app = Flask(__name__)

model = joblib.load('ETC.pkl')

@app.route('/')
def home():
  return render_template('home.html')
@app.route('/about')
def about():
  return render_template('about.html')
@app.route('/predict')
def perdict():
  return render_template("predict.html")
@app.route('/pred',methods=['post'])
def predict():
    sen1 = request.form['sen1']
    sen2 = request.form['sen2']
    sen3 = request.form['sen3']
    sen4 = request.form['sen4']
    sen5 = request.form['sen5']
    sen6 = request.form['sen6']
    sen7 = request.form['sen7']
    sen8 = request.form['sen8']
    sen9 = request.form['sen9']
    sen10 = request.form['sen10']
    sample_value = [[float(sen1), float(sen2), float(sen3), float(sen4), float(sen5),
float(sen6), float(sen7), float(sen8),
          float(sen9), float(sen10)]]
    sample_value = np.array(sample_value)
    sample_value = sample_value.reshape(1, -1)
    # Scale the data
    sample_value = scale(sample_value)
    # Use the model to predict the outcome
    prediction = model.predict(sample_value)
```

```
    output = ' '
    if prediction[0] == 1:
        output = 'Liver Patient'
    else:
        output = 'Healthy'
    return render_template('submit.html', prediction=output)
if __name__ == "__main__":
  app.run(debug=True)
```

1. home.html

```
<!DOCTYPE html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Welcome to Liver Patient Analysis</title>
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="/static/styles.css">
    <link href="https://fonts.googleapis.com/css?family=Montserrat:500&display=swap"
rel="stylesheet">
  </head>
  <body>
    <header>

      <nav>
        <ul class="nav__links">
          <li><a href="/">Home</a></li>
          <li><a href="/about">About</a></li>
        </ul>
      </nav>
      <a class="cta" href="/predict">Predict</a>

    </header>

   <div class="idp-text">
   <h1> A Review of Liver Patient Analysis Methods Using Machine Learning</h1>
   </div>
   <div class="idp-b">
   <a class="cta" href="/predict">Predict</a>
    </div>
  </body>
</html>
```

2. about.html

```
<!DOCTYPE html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Prediction</title>
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="static/styles.css">
    <link href="https://fonts.googleapis.com/css?family=Montserrat:500&display=swap"
rel="stylesheet">
  </head>
  <body>
    <header>

      <nav>
        <ul class="nav__links">
          <li><a href="/">Home</a></li>
          <li><a href="/about">About</a></li>
        </ul>
      </nav>
      <a class="cta" href="/predict">Predict</a>

    </header>


    <div class="idp-p">
    <h3> Introduction - Liver Patient Analysis</h3>

    <p>
      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
      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 stages a complex task for doctors. The main objective of this paper
is to Analysis 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 leaning 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 identity the best technique Based on
this study, Random Forest with the highest accuracy out performed the other algorithms
and can be further utilized in the prediction of liver disease commended.
    </p>
```

```
      </div>
      <div class="idp-c">
      <a class="cta" href="/predict">Predict</a>
     </div>
   </body>
</html>
```

3. predict.html

```
<!DOCTYPE html>
   <head>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <title>Prediction</title>
      <meta name="description" content="">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link rel="stylesheet" href="static/styles.css">
      <link href="https://fonts.googleapis.com/css?family=Montserrat:500&display=swap"
rel="stylesheet">
   </head>
   <body>
      <header>

         <nav>
            <ul class="nav__links">
               <li><a href="/">Home</a></li>
               <li><a href="/about">About</a></li>
            </ul>
         </nav>
         <a class="cta" href="/predict">Predict</a>

      </header>


      <div class="idp-p">
      <h3> Introduction - Liver Patient Analysis</h3>

      <p>
         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
         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 doctors. The main objective of this paper
is to Analysis 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 leaning 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 identity the best technique Based on this study, Random Forest with the highest accuracy outperformed the other algorithms and can be further utilized in the prediction of liver disease recommended.

```
    </p>
     </div>
     <div class="idp-c">
     <a class="cta" href="/predict">Predict</a>
    </div>
  </body>
</html>
```

4. submit.html

```
<!DOCTYPE html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>prediction</title>
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="static/styles.css">
    <link href="https://fonts.googleapis.com/css?family=Montserrat:500&display=swap"
rel="stylesheet">
  </head>
  <body>
    <header>

      <nav>
        <ul class="nav__links">
          <li><a href="/">Home</a></li>
          <li><a href="/about">About</a></li>
        </ul>
      </nav>
      <a class="cta" href="/predict">Predict</a>

    </header>

   <div class="idp-text">
   {% if prediction %}
   {% if prediction == 'Liver Patient' %}
     <h2>Prediction: Liver Patient</h2>
   {% else %}
     <h2>Prediction: Healthy</h2>
   {% endif %}
  {% endif %}
        </div>
    </body>
</html>
```

5. styles.css(Stylesheet)

```
@import
url('https://fonts.googleapis.com/css?family=Poppins:400,500,600,700&display=swap');
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}
header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 30px 10%;
  background-color: orange;
}
.nav__links a,
.cta,
.overlay__content a {
  font-family: 'Sitka Small';
  font-weight: bold;
  font-size:18px;
  font-weight: 500;
  color: #edf0f1;
  text-decoration: none;
}

.nav__links {
  list-style: none;
  display: flex;
}
.nav__links li {
```

```
  padding: 0px 20px;
}
.nav__links li a {
  transition: color 0.3s ease 0s;
}
.nav__links li a:hover {
  color: #047f9e;
}
.cta {
  padding: 9px 25px;
  background-color: rgba(0, 136, 169, 1);
  border: none;
  border-radius: 50px;
  cursor: pointer;
  transition: background-color 0.3s ease 0s;
}
.cta:hover {
  background-color: rgba(0, 136, 169, 0.8);
}
body{
        background-color: white
}.idp-text{
        position:absolute;
        top: 45%;
        left: 50%;
        transform:translate(-50% , -30%);
        user-select:none;
}
.idp-text h1{
  font-family: cursive;
  font-size:30px;
color: orange;
```

```
font-weight: lighter;
width:1000px;
}
.idp-text h3{
  color: white;
  font-size: 20px;
  font-weight: lighter;
  padding-left: 30px;
  padding-top: 10px;
}
.idp-text h3 span{
  color: red;
}
.idp-b {
        position:absolute;
        top:50%;
        left:15%;
        margin:60px 380px;
}
.idp-p h3{
  text-align: center;
  margin-top: 15px;
  font-family: Cooper;
  font-size:29px;
  color: black;
  font-weight: lighter;
  padding: 12px 120px;
position: relative;
}
```

```
.idp-p p{
  font-size:24px;
  font-family:Sitka Subheading;
  color: black;
  font-weight: lighter;
  text-align: justify-all;
  padding: 10px 120px;
position: relative;
}
.idp-c {
  position:absolute;
  top:50%;
  left:15%;
  margin:275px 380px;
}
input::-webkit-outer-spin-button,
input::-webkit-inner-spin-button{
  -webkit-appearance: none;
  margin: 0;
}
input[type=submit]{
  -moz-appearance: textfield;
}
html {
  height: 100%;
}
```

6. Liver_ Patient. Analysis .ipynb


```
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_Aminotr
0	65	Female	0.7	0.1	187	
1	62	Male	10.9	5.5	699	
2	62	Male	7.3	4.1	490	
3	58	Male	1.0	0.4	182	
4	72	Male	3.9	2.0	195	

```
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
Alamine_Aminotransferase    0
Aspartate_Aminotransferase  0
Total_Protiens              0
Albumin                     0
Albumin_and_Globulin_Ratio  4
```

```
Dataset                     0
dtype: int64
```

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

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

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

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

```
data.head()
```

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	A
0	65	0	0.7	0.1	187	
1	62	1	10.9	5.5	699	
2	62	1	7.3	4.1	490	
3	58	1	1.0	0.4	182	
4	72	1	3.9	2.0	195	

Exploratory Data Analysis

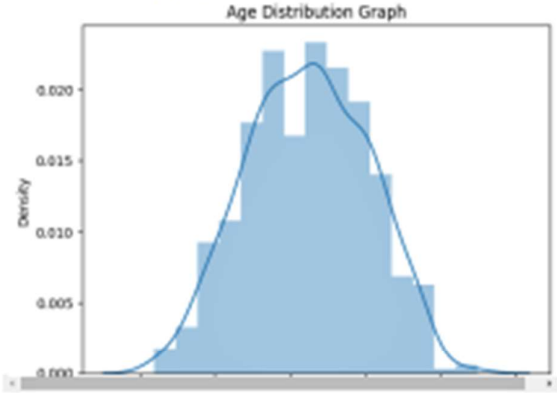
```
data.describe()
```

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_
count	583.000000	583.000000	583.000000	583.000000	
mean	44.746141	0.756432	3.298799	1.486106	
std	16.189833	0.429603	6.209522	2.808498	
min	4.000000	0.000000	0.400000	0.100000	
25%	33.000000	1.000000	0.800000	0.200000	
50%	45.000000	1.000000	1.000000	0.300000	
75%	58.000000	1.000000	2.600000	1.300000	

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

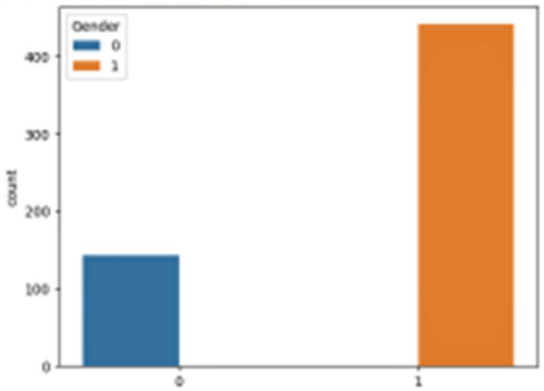
```
<ipython-input-190-a9633a3b6a8d>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0

Please adapt your code to use either `displot` (a figure-level function wi
similar flexibility) or `histplot` (an axes-level function for histograms)

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

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



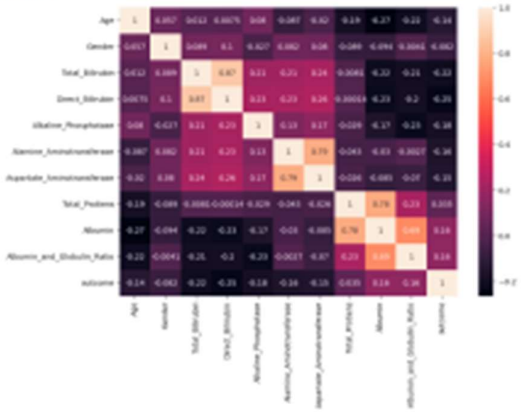

```
sns.countplot(x="Gender", hue="Gender", data=data)
```

```
<Axes: xlabel='Gender', ylabel='count'>
```




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

```
<Axes: >
```




```
from sklearn.preprocessing import scale
X_scaled=pd.DataFrame (scale(X), columns= X.columns)
```

```
X_scaled.head()
```

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phospho
0	1.252098	-1.702281	-0.418878	-0.493964	-0.42
1	1.066037	0.587446	1.225171	1.430423	1.08
2	1.066037	0.587446	0.644919	0.931508	0.82
3	0.819356	0.587446	-0.370523	-0.387054	-0.44
4	1.684839	0.587446	0.096902	0.183135	-0.39

```
y
```

```
0      1
1      1
2      1
3      1
4      1
      ..
579    2
579    1
580    1
581    1
582    2
Name: outcome, Length: 583, dtype: int64
```

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

```
X
```

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase
0	65	0	0.7	0.1	187
1	62	1	10.9	5.5	699
2	62	1	7.3	4.1	490
3	58	1	1.0	0.4	182
4	72	1	3.9	2.0	195
...	...	...	...	...	...
578	60	1	0.5	0.1	500
579	40	1	0.6	0.1	98
580	52	1	0.8	0.2	245
581	31	1	1.3	0.5	184

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X ,y, test_size=0.2, random_state=42)
```

```
pip install imblearn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: imblearn in /usr/local/lib/python3.9/dist-packages (0.0)
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.9/dist-packages (from imblearn) (0.10.1)
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: scipy>=1.3.2 in /usr/local/lib/python3.9/dist-packages (from imbalanced-learn->imblearn) (1.10.1)
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: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-packages (from imbalanced-learn->imblearn) (1.1.1)
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.9/dist-packages (from imbalanced-learn->imblearn) (1.2.2)
```

```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
```

```
y_train.value_counts()
```

```
1    329
2    137
Name: outcome, dtype: int64
```

```
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

```
y_train_smote.value_counts()
```

```
1    329
2    329
Name: outcome, dtype: int64
```

Model Building

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
import pandas as pd
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       0.80      0.77      0.78        87
```

```
           2       0.39      0.43      0.41        30

    accuracy                           0.68       117
   macro avg       0.60      0.60      0.60       117
weighted avg       0.69      0.68      0.69       117
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-193-8c4e61c2efdb> in <cell line: 1>()
----> 1 frrr

NameError: name 'frrr' is not defined
```

```
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       0.80      0.68      0.73        87
           2       0.35      0.50      0.41        30

    accuracy                           0.63       117
   macro avg       0.57      0.59      0.57       117
weighted avg       0.68      0.63      0.65       117
```

```
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       0.80      0.63      0.71        87
           2       0.33      0.53      0.41        30

    accuracy                           0.61       117
   macro avg       0.57      0.58      0.56       117
weighted avg       0.68      0.61      0.63       117
```

```
from sklearn.linear_model import LogisticRegression
model5=LogisticRegression()
model5.fit(X_train_smote, y_train_smote)
y_predict=model5.predict(X_test)
log11=accuracy_score (y_test, y_predict)
log11
pd.crosstab (y_test,y_predict)
print (classification_report (y_test, y_predict))
```

```
              precision    recall  f1-score   support

           1       0.96      0.63      0.76        87
           2       0.47      0.93      0.62        30

    accuracy                           0.71       117
   macro avg       0.72      0.78      0.69       117
weighted avg       0.84      0.71      0.73       117

/usr/local/lib/python3.8/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
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'])
```

```
model_history = classifier.fit(X_train, y_train, batch_size=100, validation_split=0.2, epochs=100)
```

```
Epoch 1/100
4/4 [==============================] - 3s 406ms/step - loss: -9.7616 - accuracy: 0.7016 - val_loss: -13.5747 - val_accuracy: 0.7234
Epoch 2/100
4/4 [==============================] - 0s 39ms/step - loss: -18.1570 - accuracy: 0.7016 - val_loss: -20.8987 - val_accuracy: 0.7234
Epoch 3/100
4/4 [==============================] - 0s 62ms/step - loss: -26.6301 - accuracy: 0.7016 - val_loss: -28.3389 - val_accuracy: 0.7234
Epoch 4/100
4/4 [==============================] - 0s 88ms/step - loss: -36.9400 - accuracy: 0.7016 - val_loss: -36.1598 - val_accuracy: 0.7234
Epoch 5/100
4/4 [==============================] - 0s 50ms/step - loss: -46.0960 - accuracy: 0.7016 - val_loss: -44.2829 - val_accuracy: 0.7234
Epoch 6/100
4/4 [==============================] - 0s 57ms/step - loss: -53.5357 - accuracy: 0.7016 - val_loss: -53.3573 - val_accuracy: 0.7234
Epoch 7/100
4/4 [==============================] - 0s 46ms/step - loss: -66.3457 - accuracy: 0.7016 - val_loss: -63.7904 - val_accuracy: 0.7234
Epoch 8/100
4/4 [==============================] - 0s 39ms/step - loss: -76.9120 - accuracy: 0.7016 - val_loss: -75.6600 - val_accuracy: 0.7234
Epoch 9/100
4/4 [==============================] - 0s 62ms/step - loss: -90.1789 - accuracy: 0.7016 - val_loss: -89.4181 - val_accuracy: 0.7234
Epoch 10/100
4/4 [==============================] - 0s 36ms/step - loss: -106.6882 - accuracy: 0.7016 - val_loss: -105.0504 - val_accuracy: 0.7234
Epoch 11/100
4/4 [==============================] - 0s 20ms/step - loss: -125.9354 - accuracy: 0.7016 - val_loss: -122.8825 - val_accuracy: 0.7234
Epoch 12/100
4/4 [==============================] - 0s 30ms/step - loss: -146.2602 - accuracy: 0.7016 - val_loss: -143.3069 - val_accuracy: 0.7234
Epoch 13/100
4/4 [==============================] - 0s 24ms/step - loss: -170.9753 - accuracy: 0.7016 - val_loss: -166.5192 - val_accuracy: 0.7234
Epoch 14/100
4/4 [==============================] - 0s 36ms/step - loss: -197.2821 - accuracy: 0.7016 - val_loss: -193.0482 - val_accuracy: 0.7234
Epoch 15/100
4/4 [==============================] - 0s 58ms/step - loss: -228.3844 - accuracy: 0.7016 - val_loss: -222.8955 - val_accuracy: 0.7234
Epoch 16/100
4/4 [==============================] - 0s 48ms/step - loss: -262.4264 - accuracy: 0.7016 - val_loss: -256.4129 - val_accuracy: 0.7234
Epoch 17/100
4/4 [==============================] - 0s 43ms/step - loss: -301.1494 - accuracy: 0.7016 - val_loss: -293.9831 - val_accuracy: 0.7234
Epoch 18/100
4/4 [==============================] - 0s 40ms/step - loss: -346.2448 - accuracy: 0.7016 - val_loss: -335.6877 - val_accuracy: 0.7234
Epoch 19/100
4/4 [==============================] - 0s 25ms/step - loss: -395.8590 - accuracy: 0.7016 - val_loss: -382.1789 - val_accuracy: 0.7234
Epoch 20/100
4/4 [==============================] - 0s 28ms/step - loss: -449.8561 - accuracy: 0.7016 - val_loss: -434.5131 - val_accuracy: 0.7234
Epoch 21/100
4/4 [==============================] - 0s 29ms/step - loss: -507.7209 - accuracy: 0.7016 - val_loss: -493.0289 - val_accuracy: 0.7234
Epoch 22/100
4/4 [==============================] - 0s 85ms/step - loss: -577.3880 - accuracy: 0.7016 - val_loss: -557.2144 - val_accuracy: 0.7234
Epoch 23/100
4/4 [==============================] - 0s 42ms/step - loss: -651.3359 - accuracy: 0.7016 - val_loss: -628.1809 - val_accuracy: 0.7234
Epoch 24/100
4/4 [==============================] - 0s 31ms/step - loss: -736.5449 - accuracy: 0.7016 - val_loss: -705.5308 - val_accuracy: 0.7234
Epoch 25/100
4/4 [==============================] - 0s 53ms/step - loss: -823.1683 - accuracy: 0.7016 - val_loss: -791.1340 - val_accuracy: 0.7234
Epoch 26/100
4/4 [==============================] - 0s 26ms/step - loss: -923.3344 - accuracy: 0.7016 - val_loss: -884.0878 - val_accuracy: 0.7234
Epoch 27/100
4/4 [==============================] - 0s 40ms/step - loss: -1035.2036 - accuracy: 0.7016 - val_loss: -986.5273 - val_accuracy: 0.723
Epoch 28/100
4/4 [==============================] - 0s 50ms/step - loss: -1146.5388 - accuracy: 0.7016 - val_loss: -1095.0236 - val_accuracy: 0.72
Epoch 29/100
```

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

```
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClassifier
  warnings.warn(
array([1])
```

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

```
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestClassifier
  warnings.warn(
array([1])
```

```
classifier.save("liver.h5")
```

```
y_pred = classifier.predict(X_test)
```

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

```
y_pred
```

```
array([[1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
```

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

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

```
y
```

```
0      1
1      1
2      1
3      1
4      1
      ..
578    2
579    1
580    1
581    1
582    2
Name: outcome, Length: 583, dtype: int64
```

```
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 ')
```

```
1/1 [==============================] - 0s 67ms/step
Prediction: Liver Patient
```

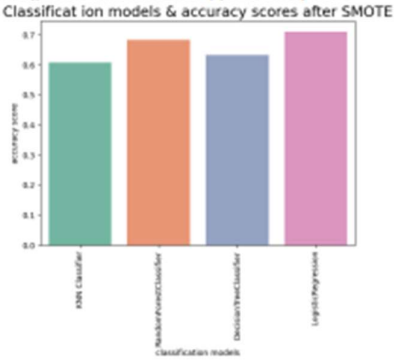
Performance Testing & Hyperparmeter tuning

```
acc_smote= [['KNN Classifier', knn1], ['RandomForestClassifier', rfc1], ['DecisionTreeClassifier', dtc1], [' LogisticRegression' , log1]]
liverpatient_pred= pd.DataFrame(acc_smote, columns = ['classification models', 'accuracy score'])
liverpatient_pred
```

	classification models	accuracy score
0	KNN Classifier	0.606838
1	RandomForestClassifier	0.683761
2	DecisionTreeClassifier	0.632479
3	LogisticRegression	0.709402

```
plt.figure(figsize=(7, 5))
plt.xticks(rotation=90)
plt.title('Classificat ion models & accuracy scores after SMOTE' , fontsize=18)
sns.barplot(x="classification models", y="accuracy score", data=liverpatient_pred, palette ="Set2")
```

```
<Axes: title={'center': 'Classificat ion models & accuracy scores after
SMOTE'}, xlabel='classification models', ylabel='accuracy score'>
```



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

```
ExtraTreesClassifier
ExtraTreesClassifier()
```

```
model.feature_importances_
```

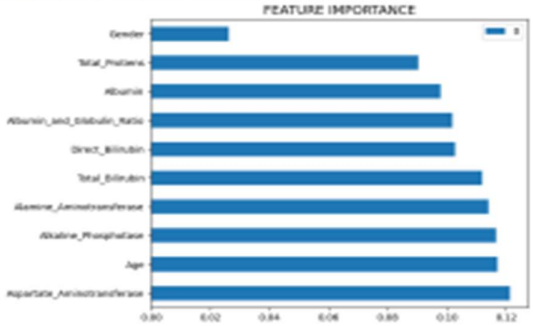
```
array([0.11702872, 0.02622561, 0.11202284, 0.10276433, 0.11650681,
       0.11404011, 0.12130022, 0.09025552, 0.09784651, 0.10195132])
```

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

	0
Aspartate_Aminotransferase	0.121300
Age	0.117029
Alkaline_Phosphotase	0.116585
Alanine_Aminotransferase	0.114040
Total_Bilirubin	0.112023
Direct_Bilirubin	0.102744
Albumin_and_Globulin_Ratio	0.101951
Albumin	0.097847
Total_Proteins	0.090256

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

```
Text(0.5, 1.0, 'FEATURE IMPORTANCE')
```



model Devlopment

```
import joblib
joblib.dump(model1, 'GTC.pkl')
```

```
['GTC.pkl']
```