# A Review of Liver Patient Analysis Methods using Machine Learning

## INTRODUCTION

## OVERVIEW

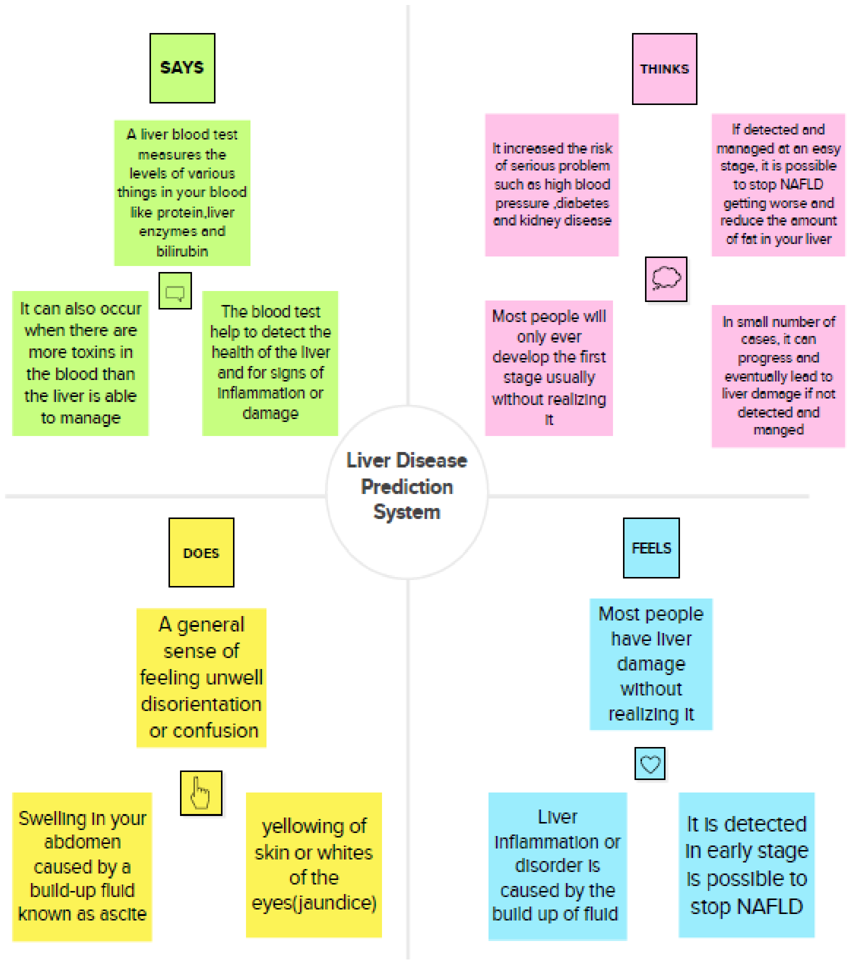
- An early diagnosis of liver problems will increase patient's survival rate
- In this project we have taken Dataset which contains 10 variables Gender, total Bilirubin, direct Bilirubin, total proteins, albumin, A/G ratio and other necessary parameters
- This project aims to identify a suitable machine learning which is capable of identifying whether a person has a liver disease or not

## PURPOSE

- In order to find the solution, the dataset is trained by various supervised learning techniques. when the new data point is arrived, it should give the better prediction and good accuracy result.
- With the help of the dataset, we can able to predict the patient has liver disease or not

## PROBLEM DEFINITION & DESIGN THINKING

## EMPATHY MAP



SAYS

A liver blood test measures the levels of various things in your blood like protein,liver enzymes and bilirubin

It can also occur when there are more toxins in the blood than the liver is able to manage

The blood test help to detect the health of the liver and for signs of Inflammation or damage

THINKS

It increased the risk of serious problem such as high blood pressure ,diabetes and kidney disease

If detected and managed at an easy stage, it is possible to stop NAFLD getting worse and reduce the amount of fat in your liver

Most people will only ever develop the first stage usually without realizing it

In small number of cases, it can progress and eventually lead to liver damage if not detected and manged

DOES

A general sense of feeling unwell disorientation or confusion

Swelling in your abdomen caused by a build-up fluid known as ascite

yellowing of skin or whites of the eyes(jaundice)

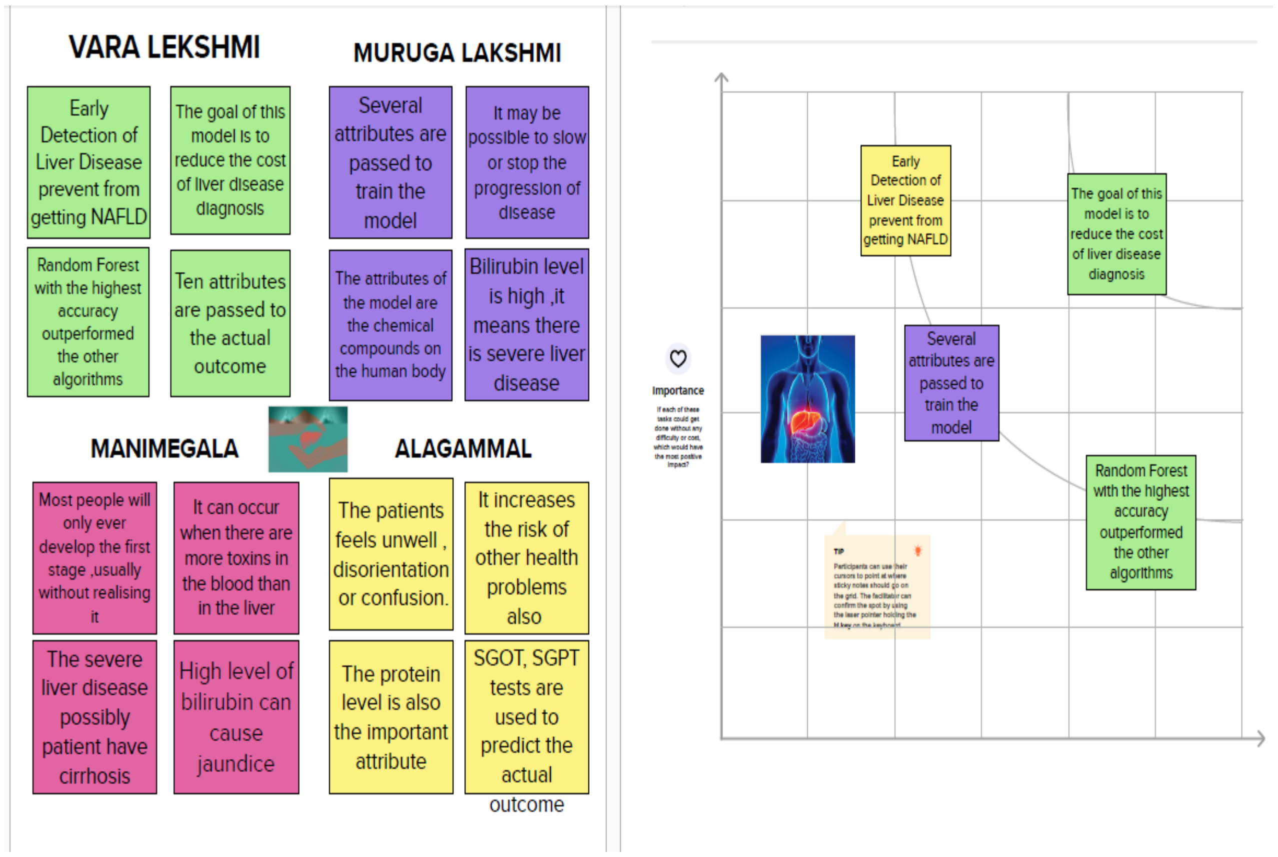
FEELS

Most people have liver damage without realizing it

Liver inflammation or disorder is caused by the build up of fluid

It is detected in early stage is possible to stop NAFLD

## IDEATION & BRAINSTORMING MAP



VARA LEKSHMI

Early Detection of Liver Disease prevent from getting NAFLD

The goal of this model is to reduce the cost of liver disease diagnosis

Random Forest with the highest accuracy outperformed the other algorithms

Ten attributes are passed to the actual outcome

MURUGA LAKSHMI

Several attributes are passed to train the model

It may be possible to slow or stop the progression of disease

The attributes of the model are the chemical compounds on the human body

Bilirubin level is high ,it means there is severe liver disease

MANIMEGALA

Most people will only ever develop the first stage ,usually without realising it

It can occur when there are more toxins in the blood than in the liver

The severe liver disease possibly patient have cirrhosis

High level of bilirubin can cause jaundice

ALAGAMMAL

The patients feels unwell , disorientation or confusion.

It increases the risk of other health problems also

The protein level is also the important attribute

SGOT, SGPT tests are used to predict the actual outcome

Early Detection of Liver Disease prevent from getting NAFLD

The goal of this model is to reduce the cost of liver disease diagnosis

Importance

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

Several attributes are passed to train the model

Random Forest with the highest accuracy outperformed the other algorithms

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the H key on the keyboard.

## RESULT

### Importing the Libraries

```
In [9]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        from matplotlib import rcParams
        import scipy
        from scipy import stats
```

## Read the Dataset

In [10]:
```python
data=pd.read_csv('C:/Users/ELCOT/anaconda3/indian_liver_patient.csv')
```

In [11]:
```python
data.head()
```

Out[11]:

| | Age | Gender | Total_Bilirubin | Direct_Bilirubin | Alkaline_Phosphotase | Alamine_Aminotransferase | Aspartate_Aminotransferase |
|---|---|---|---|---|---|---|---|
| **0** | 65 | Female | 0.7 | 0.1 | 187 | 16 | 18 |
| **1** | 62 | Male | 10.9 | 5.5 | 699 | 64 | 100 |
| **2** | 62 | Male | 7.3 | 4.1 | 490 | 60 | 68 |
| **3** | 58 | Male | 1.0 | 0.4 | 182 | 14 | 20 |
| **4** | 72 | Male | 3.9 | 2.0 | 195 | 27 | 59 |

## Data Preparation

## Handling Missing Values

In [12]:
```python
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 [13]:
```python
data.isnull().any()
```

Out[13]:
```
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 [14]:
```python
data.isnull().sum()
```

Out[14]:
```
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 [15]:
```python
data['Albumin_and_Globulin_Ratio'].mean()
```

Out[15]:
```
0.9470639032815197
```

```
In [16]: data=data.fillna(0.94)
```

```
In [17]: data.isnull().sum()
```

```
Out[17]: 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
```

## Handling Categorical Values

```
In [18]: from sklearn.preprocessing import LabelEncoder
         lc=LabelEncoder()
         data['Gender']=lc.fit_transform(data['Gender'])
```

## Exploratory Data Analysis

## Descriptive Statistical

```
In [19]: data.describe()
```

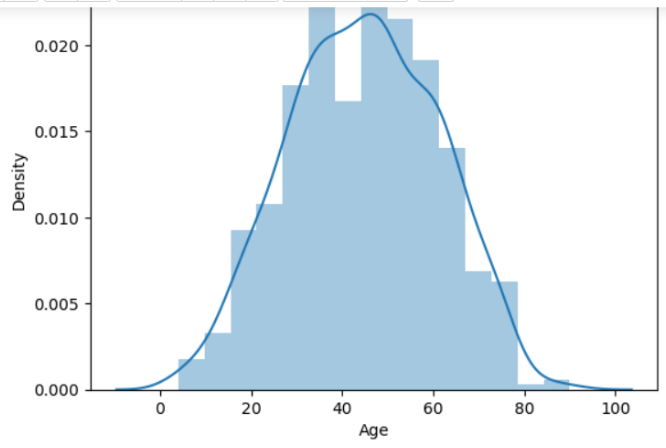
Out[19]:

| | Age | Gender | Total_Bilirubin | Direct_Bilirubin | Alkaline_Phosphotase | Alamine_Aminotransferase | Aspartate_Ami |
|---|---|---|---|---|---|---|---|
| **count** | 583.000000 | 583.000000 | 583.000000 | 583.000000 | 583.000000 | 583.000000 | |
| **mean** | 44.746141 | 0.756432 | 3.298799 | 1.486106 | 290.576329 | 80.713551 | |
| **std** | 16.189833 | 0.429603 | 6.209522 | 2.808498 | 242.937989 | 182.620356 | |
| **min** | 4.000000 | 0.000000 | 0.400000 | 0.100000 | 63.000000 | 10.000000 | |
| **25%** | 33.000000 | 1.000000 | 0.800000 | 0.200000 | 175.500000 | 23.000000 | |
| **50%** | 45.000000 | 1.000000 | 1.000000 | 0.300000 | 208.000000 | 35.000000 | |
| **75%** | 58.000000 | 1.000000 | 2.600000 | 1.300000 | 298.000000 | 60.500000 | |
| **max** | 90.000000 | 1.000000 | 75.000000 | 19.700000 | 2110.000000 | 2000.000000 | |

## Visual Analysis

## Univariate Analysis

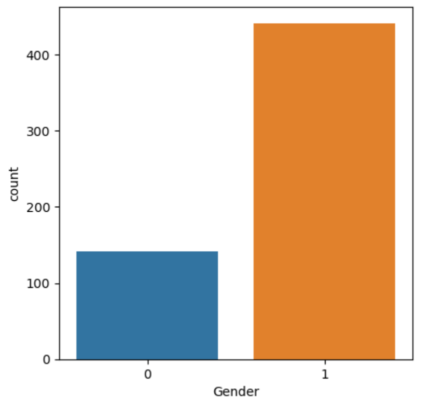
```
In [20]: sns.distplot(data['Age'])
         plt.title('Age Distribution graph')
         plt.show()
```



## Bivariate Analysis

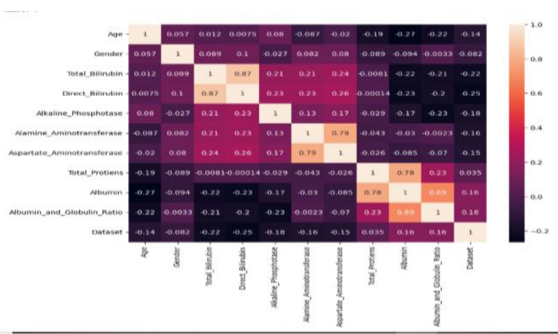
```
In [21]: plt.figure(figsize=(5,5))
         sns.countplot(x='Gender', data=data)
```

```
Out[21]: <Axes: xlabel='Gender', ylabel='count'>
```

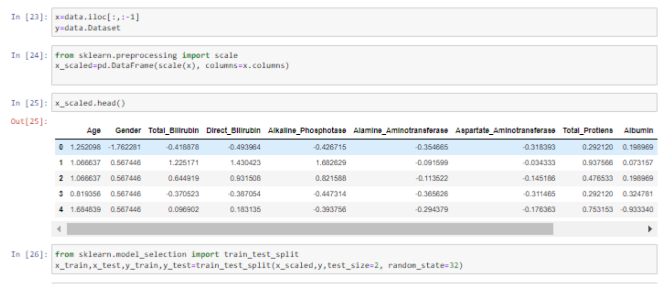


## Multivariate Analysis

```
In [22]: plt.figure(figsize=(10,7))
         sns.heatmap(data.corr(),annot=True)
```



## Scaling the Data

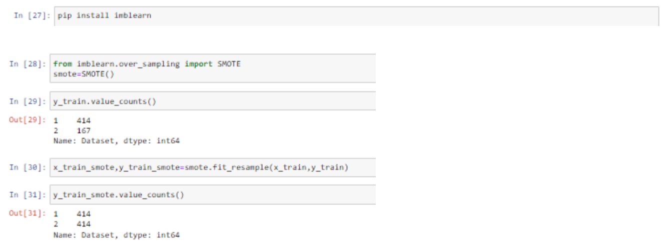


```
In [23]: x=data.iloc[:,:-1]
         y=data.Dataset

In [24]: from sklearn.preprocessing import scale
         x_scaled=pd.DataFrame(scale(x), columns=x.columns)

In [25]: x_scaled.head()

Out[25]:
```

| | 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 [26]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=2, random_state=32)
```

## Handling Imbalanced Data

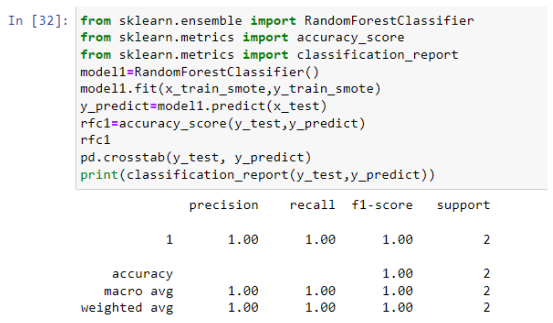


```
In [27]: pip install imblearn

In [28]: from imblearn.over_sampling import SMOTE
         smote=SMOTE()

In [29]: y_train.value_counts()

Out[29]: 1    414
         2    167
         Name: Dataset, dtype: int64

In [30]: x_train_smote,y_train_smote=smote.fit_resample(x_train,y_train)

In [31]: y_train_smote.value_counts()

Out[31]: 1    414
         2    414
         Name: Dataset, dtype: int64
```

## Model Building

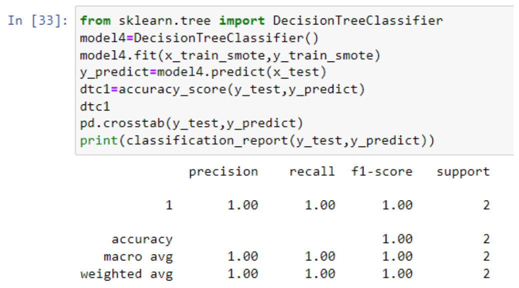
## Training the model with multiple algorithm

## Random Forest model



```
In [32]: 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       1.00      1.00      1.00         2

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```

## Decision Tree model



```
In [33]: 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       1.00      1.00      1.00         2

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```

## KNN model

```
In [34]: 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       1.00      1.00      1.00         2

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```

## Logistic Regression model

```
In [35]: 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       1.00      1.00      1.00         2

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```

## ANN model

```
In [36]: import tensorflow.keras
         from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Dense
```

```
In [37]: classifier=Sequential()
```

```
In [38]: classifier.add(Dense(units=100, activation='relu', input_dim=10))
```

```
In [39]: classifier.add(Dense(units=50, activation='relu'))
```

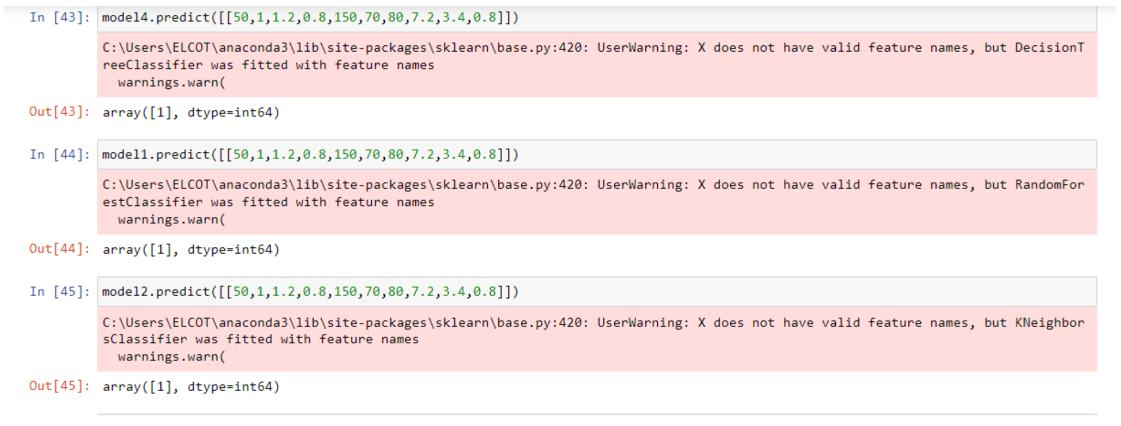
```
In [40]: classifier.add(Dense(units=1, activation='sigmoid'))
```

```
In [41]: classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

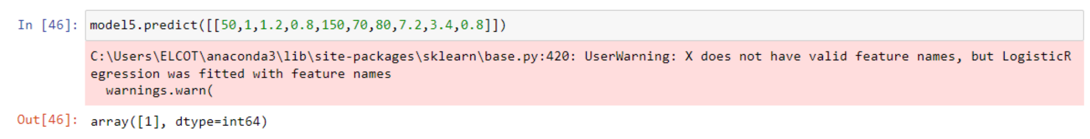
```
In [42]: model_history=classifier.fit(x_train,y_train, batch_size=100, validation_split=0.2, epochs=100)
```

```
Epoch 1/100
5/5 [==============================] - 5s 277ms/step - loss: 0.7306 - accuracy: 0.2888 - val_loss: 0.5561 - val_accuracy: 0.5
214
Epoch 2/100
5/5 [==============================] - 0s 40ms/step - loss: 0.4512 - accuracy: 0.6509 - val_loss: 0.3043 - val_accuracy: 0.66
67
Epoch 3/100
5/5 [==============================] - 0s 43ms/step - loss: 0.2090 - accuracy: 0.7241 - val_loss: 0.0759 - val_accuracy: 0.66
67
Epoch 4/100
5/5 [==============================] - 0s 41ms/step - loss: -0.0013 - accuracy: 0.7241 - val_loss: -0.1465 - val_accuracy: 0.
6667
Epoch 5/100
5/5 [==============================] - 0s 36ms/step - loss: -0.2034 - accuracy: 0.7241 - val_loss: -0.3672 - val_accuracy: 0.
6667
Epoch 6/100
5/5 [==============================] - 0s 35ms/step - loss: -0.4022 - accuracy: 0.7241 - val_loss: -0.5902 - val_accuracy: 0.
6667
Epoch 7/100
```

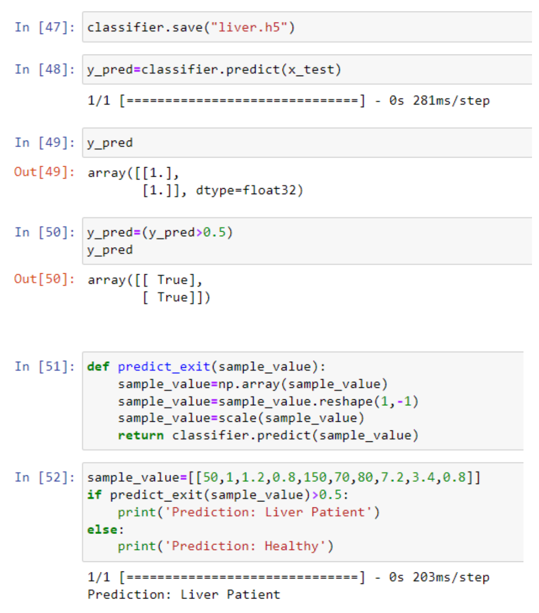
## Testing the model



```
In [43]: model4.predict([[50,1,1.2,0.8,150,70,80,7.2,3.4,0.8]])

C:\Users\ELCOT\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X does not have valid feature names, but DecisionT
reeClassifier was fitted with feature names
  warnings.warn(

Out[43]: array([1], dtype=int64)

In [44]: model1.predict([[50,1,1.2,0.8,150,70,80,7.2,3.4,0.8]])

C:\Users\ELCOT\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X does not have valid feature names, but RandomFor
estClassifier was fitted with feature names
  warnings.warn(

Out[44]: array([1], dtype=int64)

In [45]: model2.predict([[50,1,1.2,0.8,150,70,80,7.2,3.4,0.8]])

C:\Users\ELCOT\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X does not have valid feature names, but KNeighbor
sClassifier was fitted with feature names
  warnings.warn(

Out[45]: array([1], dtype=int64)
```



```
In [46]: model5.predict([[50,1,1.2,0.8,150,70,80,7.2,3.4,0.8]])

C:\Users\ELCOT\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X does not have valid feature names, but LogisticR
egression was fitted with feature names
  warnings.warn(

Out[46]: array([1], dtype=int64)
```

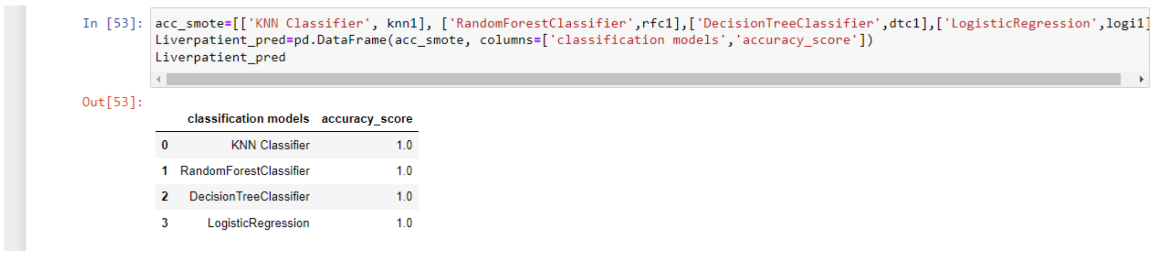


```
In [47]: classifier.save("liver.h5")

In [48]: y_pred=classifier.predict(x_test)

1/1 [==============================] - 0s 281ms/step

In [49]: y_pred

Out[49]: array([[1.],
                [1.]], dtype=float32)

In [50]: y_pred=(y_pred>0.5)
         y_pred

Out[50]: array([[ True],
                [ True]])

In [51]: 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 [52]: 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 203ms/step
Prediction: Liver Patient
```

## Performance Testing & Hyperparameter Tuning

## Testing model with multiple evaluation metrics

## Compare the model

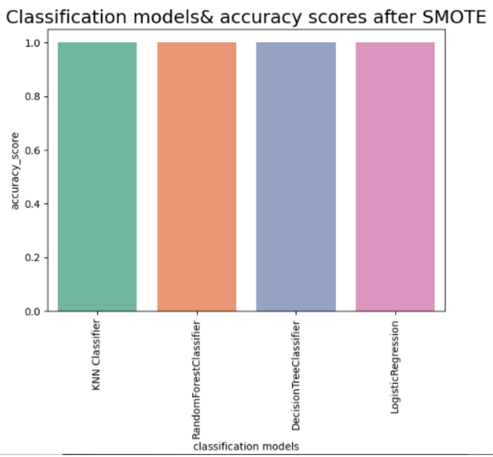


```
In [53]: acc_smote=[['KNN Classifier', knn1], ['RandomForestClassifier',rfc1],['DecisionTreeClassifier',dtc1],['LogisticRegression',logi1]
         Liverpatient_pred=pd.DataFrame(acc_smote, columns=['classification models','accuracy_score'])
         Liverpatient_pred

Out[53]:
```

| | classification models | accuracy_score |
|---|---|---|
| 0 | KNN Classifier | 1.0 |
| 1 | RandomForestClassifier | 1.0 |
| 2 | DecisionTreeClassifier | 1.0 |
| 3 | LogisticRegression | 1.0 |

```
In [54]: 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[54]: <Axes: title={'center': 'Classification models& accuracy scores after SMOTE'}, xlabel='classification models', ylabel='accuracy
         _score'>
```

```
Out[54]: <Axes: title={'center': 'Classification models& accuracy scores after SMOTE'}, xlabel='classification models', ylabel='accuracy
         _score'>
```



```
In [55]: from sklearn.ensemble import ExtraTreesClassifier
         model=ExtraTreesClassifier()
         model.fit(x,y)

Out[55]: ▾ ExtraTreesClassifier
         ExtraTreesClassifier()

In [56]: model.feature_importances_

Out[56]: array([0.12024273, 0.02435313, 0.11146545, 0.10537009, 0.11516687,
                0.11688296, 0.11713285, 0.0913383 , 0.10015124, 0.09789637])
```

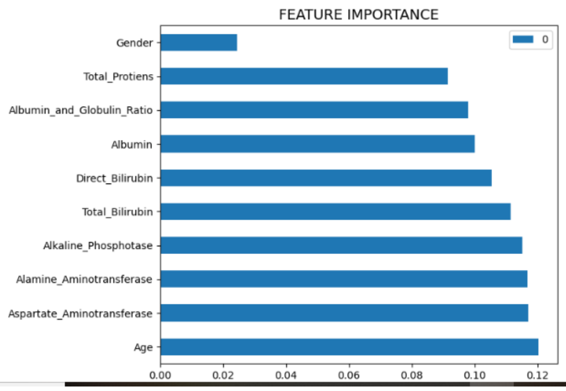
model.feature_importances_

```
In [57]: dd=pd.DataFrame(model.feature_importances_,index=x.columns).sort_values(0,ascending=False)
         dd

Out[57]:
```

| | 0 |
|---|---|
| **Age** | 0.120243 |
| **Aspartate_Aminotransferase** | 0.117133 |
| **Alamine_Aminotransferase** | 0.116883 |
| **Alkaline_Phosphotase** | 0.115167 |
| **Total_Bilirubin** | 0.111465 |
| **Direct_Bilirubin** | 0.105370 |
| **Albumin** | 0.100151 |
| **Albumin_and_Globulin_Ratio** | 0.097896 |
| **Total_Protiens** | 0.091338 |
| **Gender** | 0.024353 |

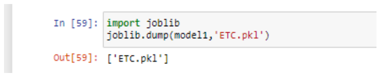
## Identifying Important Features

```
In [58]: dd.plot(kind='barh', figsize=(7,6))
         plt.title("FEATURE IMPORTANCE",fontsize=14)

Out[58]: Text(0.5, 1.0, 'FEATURE IMPORTANCE')
```



## Model Deployment

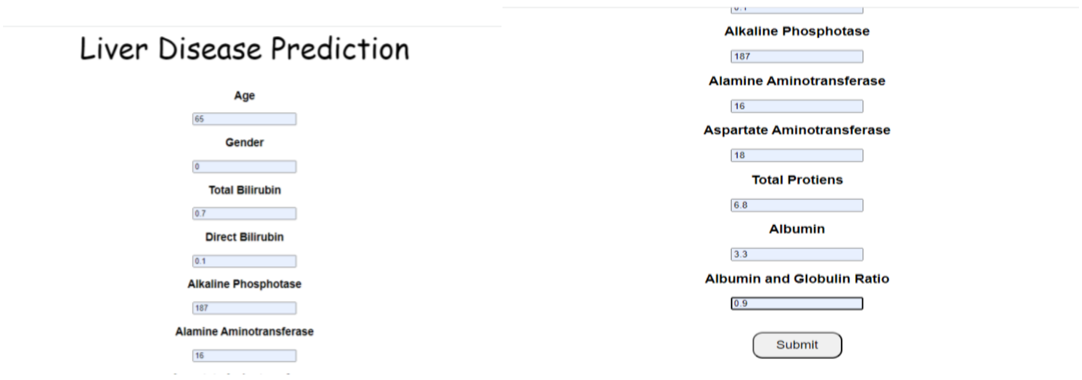
### Save the model



```
In [59]: import joblib
         joblib.dump(model1,'ETC.pkl')

Out[59]: ['ETC.pkl']
```

## Build a HTML Web Pages

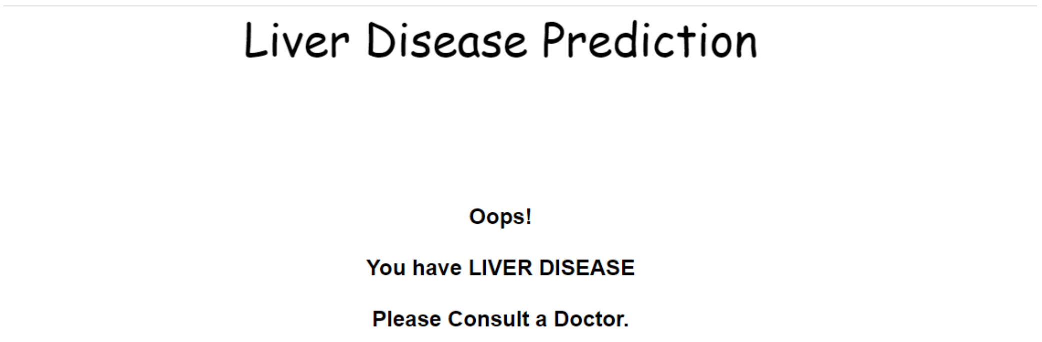
### INDEX.html



Liver Disease Prediction

| | |
|---|---|
| Age | 65 |
| Gender | 0 |
| Total Bilirubin | 0.7 |
| Direct Bilirubin | 0.1 |
| Alkaline Phosphotase | 187 |
| Alamine Aminotransferase | 16 |
| Aspartate Aminotransferase | 18 |
| Total Protiens | 6.8 |
| Albumin | 3.3 |
| Albumin and Globulin Ratio | 0.9 |

### RESULT.html



Liver Disease Prediction

Oops!

You have LIVER DISEASE

Please Consult a Doctor.

## ADVANTAGES & DISADVANTAGES

### ADVANTAGES

- Diagnostic criterion standard
- Confirmed diagnostic value
- Etiologic suggestion
- Differential diagnosis
- Grade and stage evaluation
- Therapeutic decision
- Follow-up comparison of treated and untreated patients

### DISADVANTAGES

- Highly invasive test
- The potential complications include death
- Significant sampling
- Error
- High cost
- Inter-observer variation

## APPLICATION

- Building and training the system:
  - The phase is totally worked upon by developer of the system, and end user has nothing to do with it. In this phase, we split the dataset into training dataset and test dataset, and then trained the models using training datasets
- Testing the models:
  - In this phase we tested the accuracy of the models with the test dataset that was formed in previous phase and the most accurate model is figured out.
- Entering details and Prediction:
  - In this phase, the end user comes into picture. He/she enters the details of blood test report using GUI of the application. The application then matches the details

with the training dataset of the most accurate model, and then predicts final result displaying, ‘Liver Disease’ or ‘No Liver Disease’ on the screen

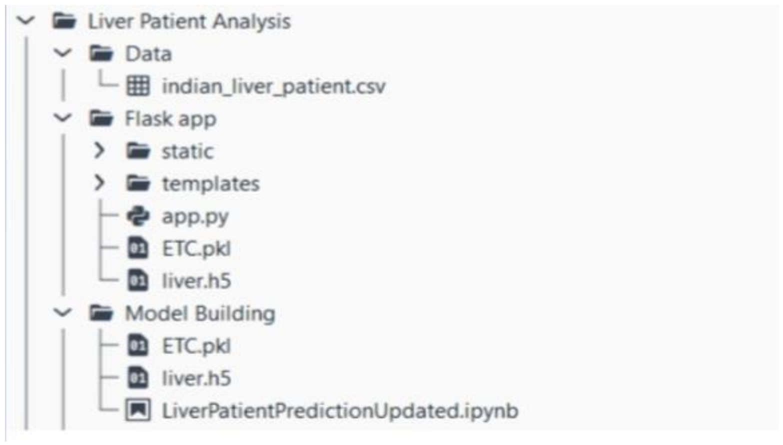
## CONCLUSION AND FUTURE WORKS

- Diseases related to the liver are becoming more common with time. With continuous technological advancements, these are only going to increase in the future. Although people are becoming more conscious of health nowadays
- Our project will be extremely helpful to the society. With the dataset used for this project, we got 100% accuracy for Random Forest Model, and though it might be difficult to get such accuracies with very large dataset, from this project results, one can clearly conclude that we can predict the risk of liver Disease with accuracy 90% or more

- Today almost everybody above the age of 12years has smartphones with them, and so we can incorporate these solutions into a website and these app and website will be highly beneficial for a large section of society

# APPENDIX

## Source Code



```
Liver Patient Analysis
  Data
    indian_liver_patient.csv
  Flask app
    static
    templates
    app.py
    ETC.pkl
    liver.h5
  Model Building
    ETC.pkl
    liver.h5
    LiverPatientPredictionUpdated.ipynb
```

app.py

```
from flask import Flask,render_template,request,url_for

import numpy as np

import pickle



app=Flask(__name__)


@app.route('/')

def index():

 return render_template('index.html')
```

```
@app.route('/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(tb),float(db),float(ap),float(aa1),float(aa2),float(tp),float(a)
,float(agr)]]

 model=pickle.load(open('ETC.pkl','rb'))

 prediction=model.predict(data)[0]

 if (prediction==1):

     return render_template('result.html',prediction=prediction)

 else:

     return render_template('result.html',prediction=prediction)


if __name__=='__main__':

  app.run(debug=True)
```

INDEX.html

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

<head>
    <meta charset="UTF-8">
    <title>Liver Prediction Model</title>

</head>
<body>
<section id="about" class="about">
   <div class="container">
        <h2 class='container-heading'><span class="heading_font">Liver Disease Prediction</span></h2>
    </div>

    <div class="ml-container">
        <div class="first">
        <form action="/predict" method="POST">
            <br>
            <h3>Age</h3>
            <input type id="age" name="age" placeholder="Age" required="required">
            <br>
            <h3>Gender</h3>
            <input type id="gender" name="gender" placeholder="Male = 1, Female=0" required="required">
            <br>
            <h3>Total Bilirubin</h3>
            <input type id="tb" name="tb" placeholder="Total Bilirubin" required="required">
            <br>
            <h3>Direct Bilirubin</h3>
```

```
          <h3>Total Bilirubin</h3>
          <input type id="tb" name="tb" placeholder="Total Bilirubin" required="required">
          <br>
          <h3>Direct Bilirubin</h3>
          <input type id="db" name="db" placeholder="Direct Bilirubin" required="required">
          <br>
          <h3>Alkaline Phosphotase</h3>
          <input type  id="ap" name="ap" placeholder="Alkaline Phosphotase" required="required">
          <br>
          <h3>Alamine Aminotransferase</h3>
          <input type id="aa1" name="aa1" placeholder="Alamine Aminotransferase" required="required">
          <br>
          <h3>Aspartate Aminotransferase</h3>
          <input type  id="aa2" name="aa2" placeholder="Aspartate Aminotransferase" required="required">
          <br>
          <h3>Total Protiens</h3>
          <input type id="tp" name="tp" placeholder="Total Protiens" required="required">
          <br>
          <h3>Albumin</h3>
          <input type id="a" name="a" placeholder="Albumin" required="required">
          <br>
          <h3>Albumin and Globulin Ratio</h3>
          <input type id="agr" name="agr" placeholder="Albumin and Globulin Ratio" required="required">
          <br>
          <br>
          <br>
          <button id="sub" type="submit " onclick="action">Submit</button>
          <br>
```

```
            <h3>Aspartate Aminotransferase</h3>
            <input type  id="aa2" name="aa2" placeholder="Aspartate Aminotransferase" required="required">
            <br>
            <h3>Total Protiens</h3>
            <input type id="tp" name="tp" placeholder="Total Protiens" required="required">
            <br>
            <h3>Albumin</h3>
            <input type id="a" name="a" placeholder="Albumin" required="required">
            <br>
            <h3>Albumin and Globulin Ratio</h3>
            <input type id="agr" name="agr" placeholder="Albumin and Globulin Ratio" required="required">
            <br>
            <br>
            <br>
            <button id="sub" type="submit " onclick="action">Submit</button>
            <br>
            <br>
            <br>
            <br>


        </form>
       </div>
   </div>
</section>


<style>
```

RESULT.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>
<section id="hero" class="d-flex flex-column justify-content-center">
    <div class="container">
        <form action="/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">
{% if prediction==1 %}
<h1><span class='danger'>Oops! <br><br>You have LIVER DISEASE <br><br>Please Consult a Doctor.</span></h1>
{% elif prediction==0 %}
<h1><span class='safe'>Congratulation! <br><br>You DON'T have LIVER DISEASE.</span></h1>
{% endif %}
</div>
        </form>

    </div>
```