A Review of Liver Patient Analysis Methods Using Machine Learning

Abstract

Machine learning (ML) utilizes artificial intelligence to generate predictive models efficiently and more effectively than conventional methods through detection of hidden patterns within large data sets. With this in mind, there are several areas within hepatology where these methods can be applied. In this review, we examine the literature pertaining to machine learning in hepatology and liver transplant medicine. We provide an overview of the strengths and limitations of ML tools and their potential applications to both clinical and molecular data in hepatology. ML has been applied to various types of data in liver disease research, including clinical, demographic, molecular, radiological, and pathological data. We anticipate that use of ML tools to generate predictive algorithms will change the face of clinical practice in hepatology and transplantation. This review will provide readers with the opportunity to learn about the ML tools available and potential applications to questions of interest in hepatology.

INTRODUCTION

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

Purpose

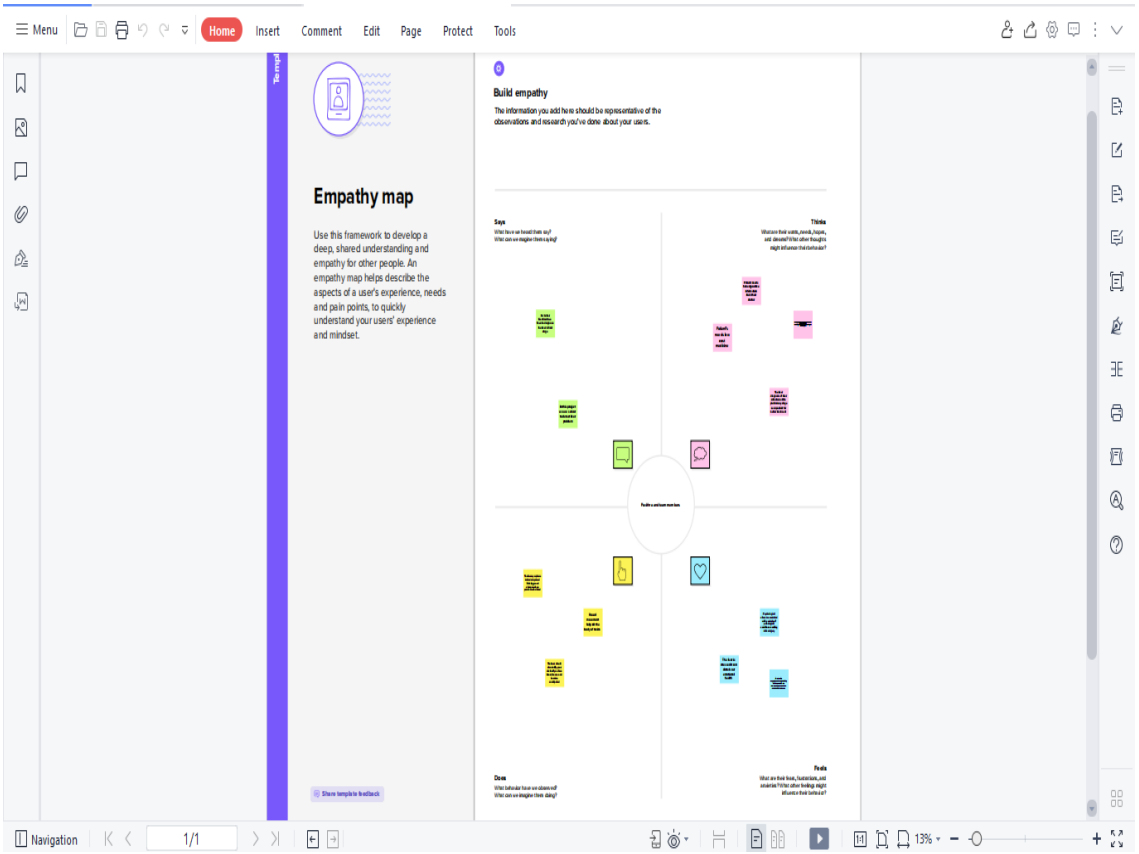
This study proposes to identify potential liver patients based on the results of a liver function test performed during a health screening to search for signs of liver disease. It is critical to detect a liver patient at an early stage in order to treat them effectively. A liver function test's level of specific enzymes and proteins in the blood is evaluated to determine if a patient has liver disease.

Liver cirrhosis is the biggest health problem posed by alcohol use, with 1.4 lakh deaths every year.

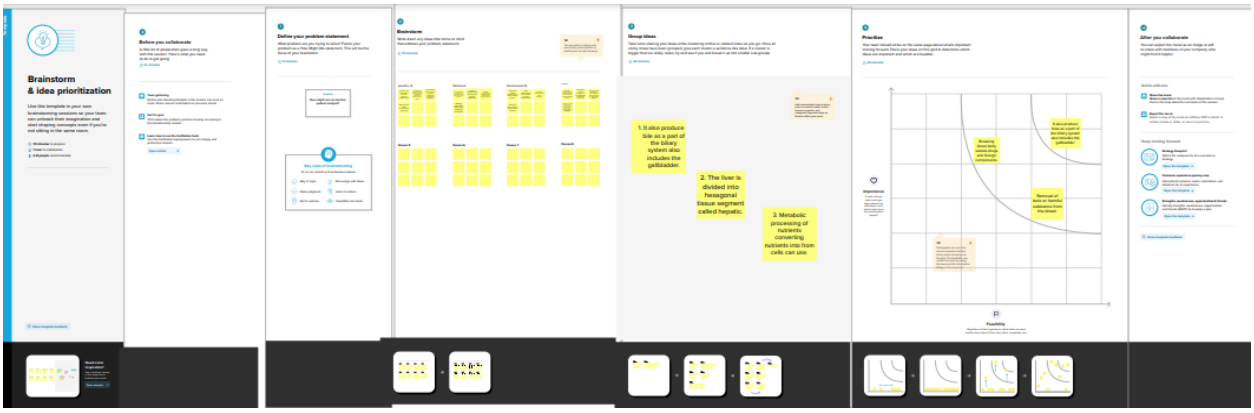
Sadly, no. In fact, it is getting more common in younger people than ever before. Dr. Amrish said that liver disease can set in childhood too as it can pass through genes.

Problem definition & design thinking

Empathy Map



Ideation & Brainstorming



Results

A total of 577 patients were included in this study; of those 377 patients had fatty liver. The area under the receiver operating characteristic (AUROC) of RF, NB, ANN, and LR with 10 fold-cross validation was 0.925, 0.888, 0.895, and 0.854 respectively. Additionally, The accuracy of RF, NB, ANN, and LR 87.48, 82.65, 81.85, and 76.96%.

Advantages

- Cirrhosis is a late-stage liver disease in which healthy liver tissue is replaced with scar tissue and the liver is permanently damaged. Scar tissue keeps your liver from working properly.

- Many types of liver diseases and conditions injure healthy liver cells, causing cell death and inflammation.01

Disadvantages

- It's possible, and dangerous, to get too much vitamin A.

- Eating large amounts of liver can lead to symptoms of vitamin A toxicity, which happens when your own liver can't process the excess vitamin A quickly enough.

- Most doctors recommend that people without vitamin deficiencies eat just one serving of liver per week.

Application

- It's used to help find the cause of health conditions like jaundice, anemia, and liver disease.

Conclusion

- In this study, we developed and compared the four classification models to predict fatty liver disease accurately.

- However, the random forest model showed higher performance than other classification models. Implementation of a random forest model in the clinical setting could help physicians to stratify fatty liver patients for primary prevention, surveillance, early treatment, and management.

Future scope

- Endoscopic retrograde cholangiopancreatography (ERCP).

- This is a procedure that helps diagnose and treat problems in the liver, gallbladder, bile ducts, and pancreas. It uses X-rays and a long, flexible, lighted tube.

Appendix

```
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.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  outcome                     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
outcome                      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
outcome                      0
dtype: int64
```

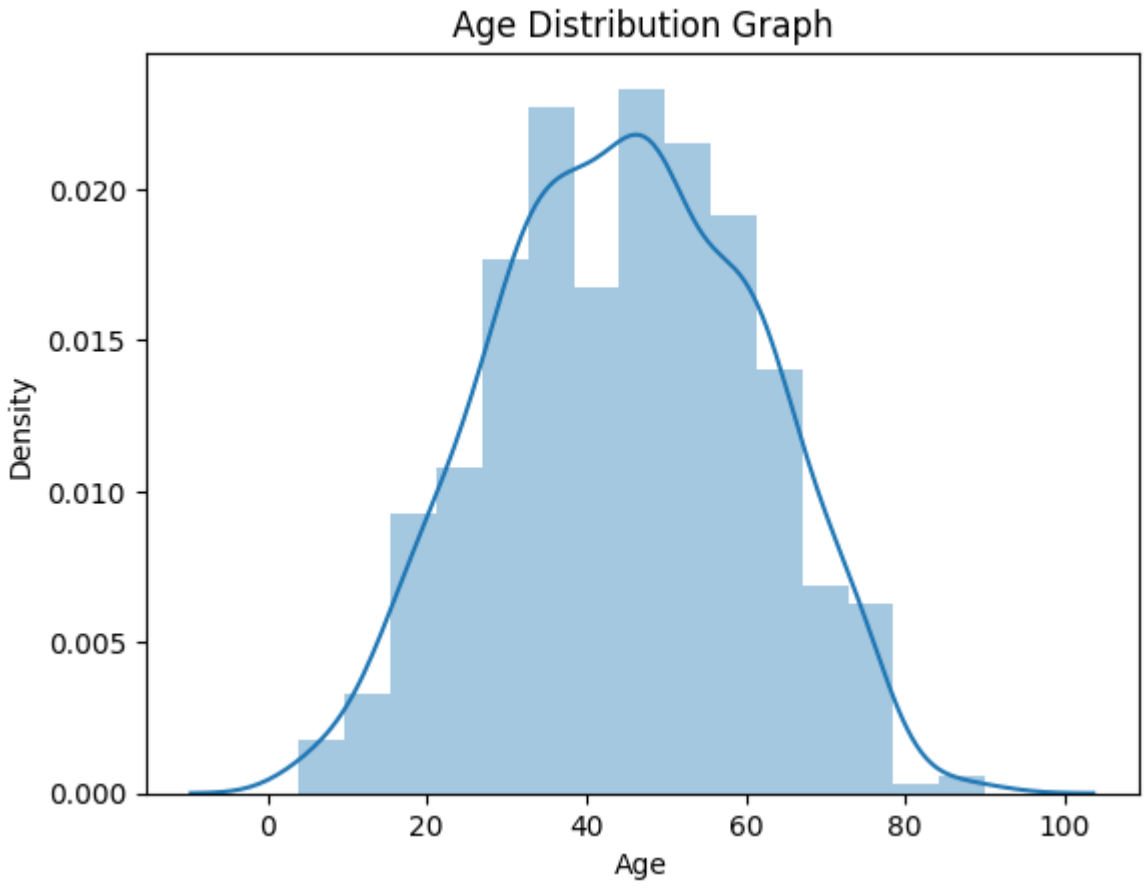
```
data['Albumin_and_Globulin_Ratio'] =data.fillna(data['Album
in_and_Globulin_Ratio'].mean(),inplace=True)
data.isnull().sum()
```

```
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   583
outcome                      0
```

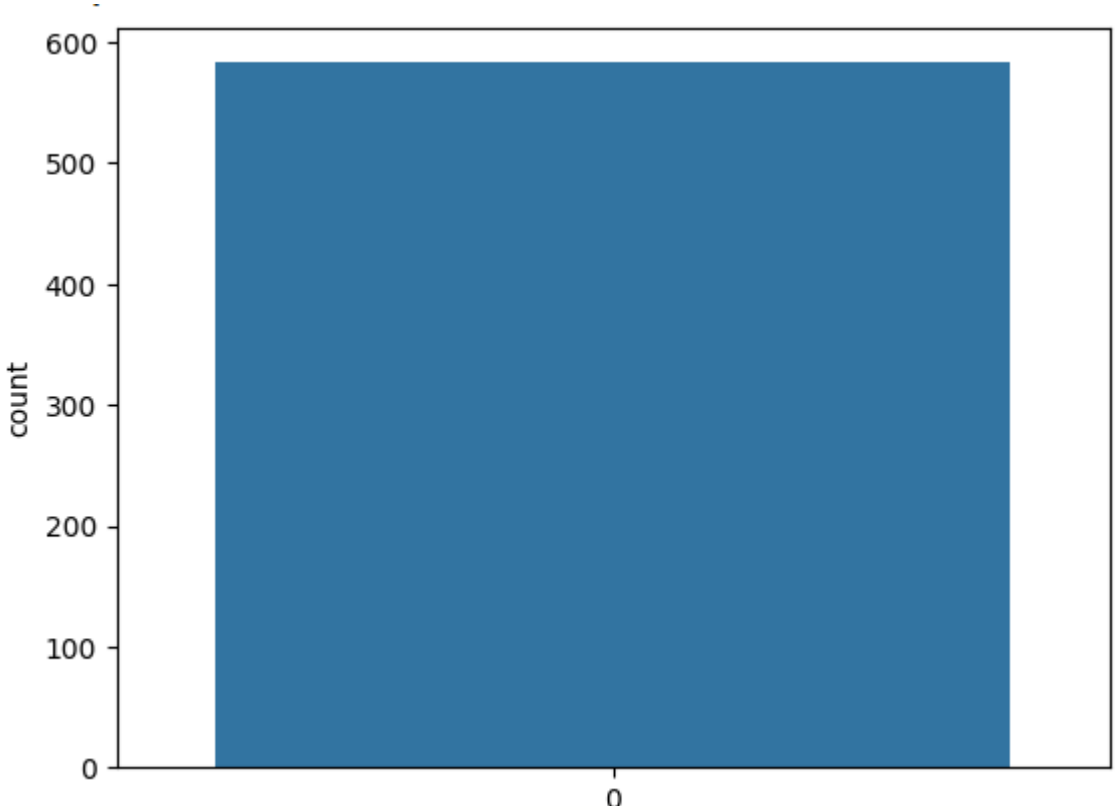
```
from sklearn.preprocessing import LabelEncoder
lc = LabelEncoder()
data['Gender']= lc.fit_transform(data['Gender']

data.describe()
```

	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
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
50%	45.000000	1.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000
75%	58.000000	1.000000	2.600000	1.300000	298.000000	60.500000	87.000000	7.200000
max	90.000000	1.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.600000

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

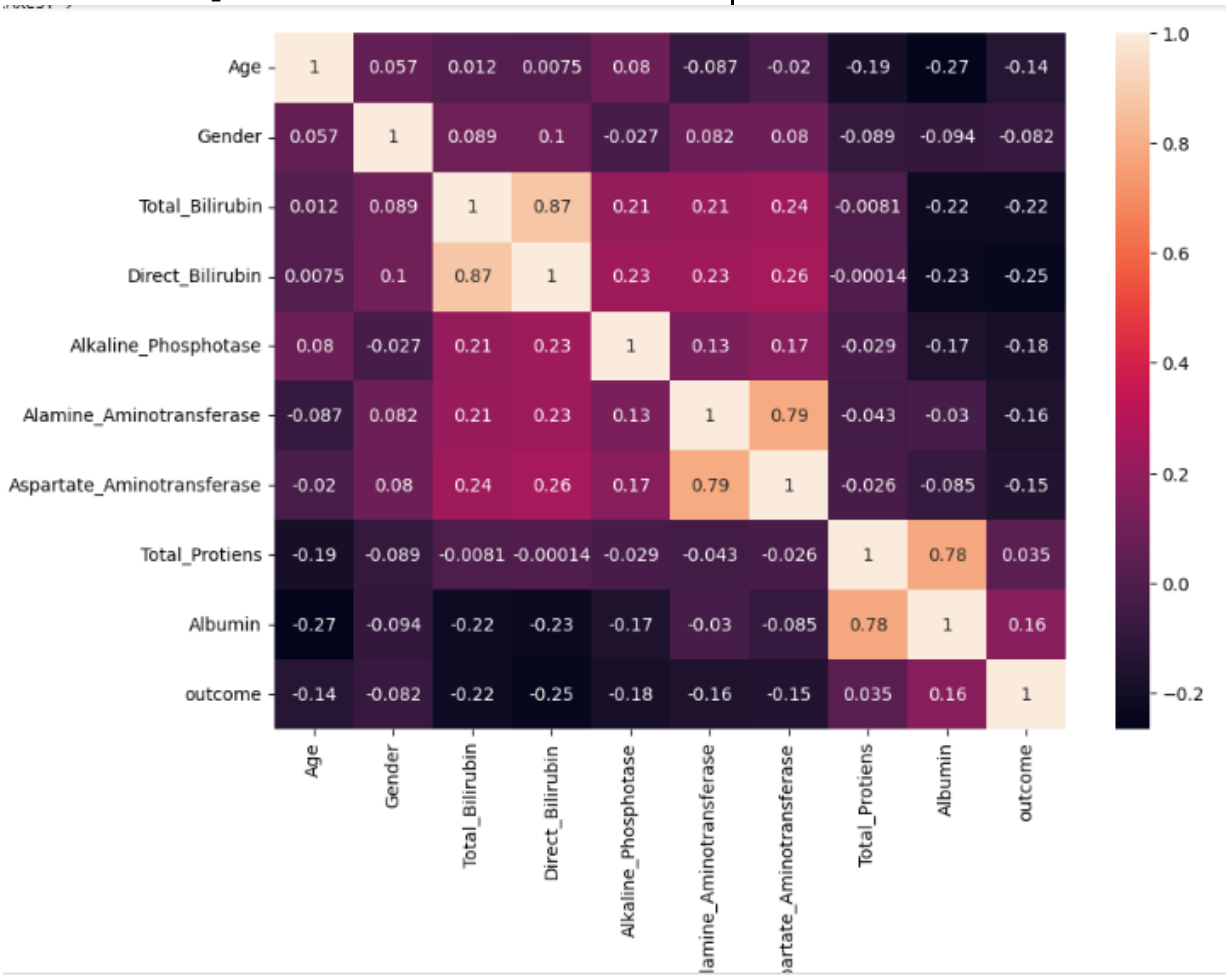


```
sns.countplot(data['outcome'])
```




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

```
sns.heatmap(data.corr(),annot=True)
```



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

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

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

```
from pandas.core.common import random_state
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_s
ize=0.2,random_state=42)
pip install imblearn
```

```
Downloading imblearn-0.0-py2.py3-none-any.whl (1.9 kB)
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.9/dist-packages (from imblearn) (0.10.1)
Requirement already satisfied: 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)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from imbalanced-learn->imblearn) (3.1.0)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.9/dist-packages (from imbalanced-learn->imblearn) (1.22.4)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.9/dist-packages (from imbalanced-learn->imblearn) (1.10.1)
Installing collected packages: imblearn
Successfully installed imblearn-0.0
```

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

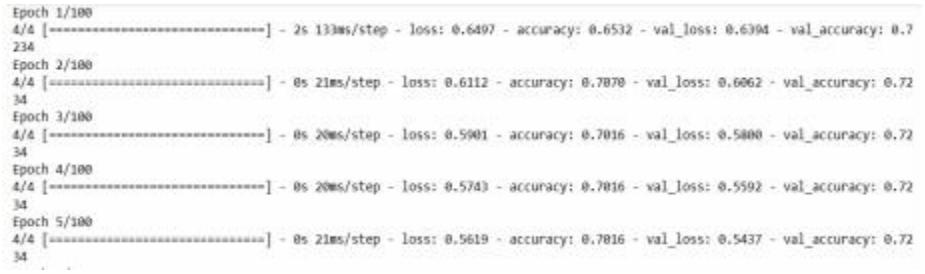
```
1 329
2 137
Name: outcome, dtype: int64
```

```
X_train_smote, Y_train_smote = smote.fit_resample(X_train,
Y_train)
Y_train_smote.value_counts()
```

```
  1  329
  0  329
  Name: outcome, dtype: int64
```

```
from sklearn.ensemble import RandomForestClassifier
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(clasification_report(Y_test,Y_predict))

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))

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))

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))
```

```
import tensorflow as Keras
from tensorflow.Keras.module 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='s='binary_crossentropy',metri
cs=['accuracy'])adam',los

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

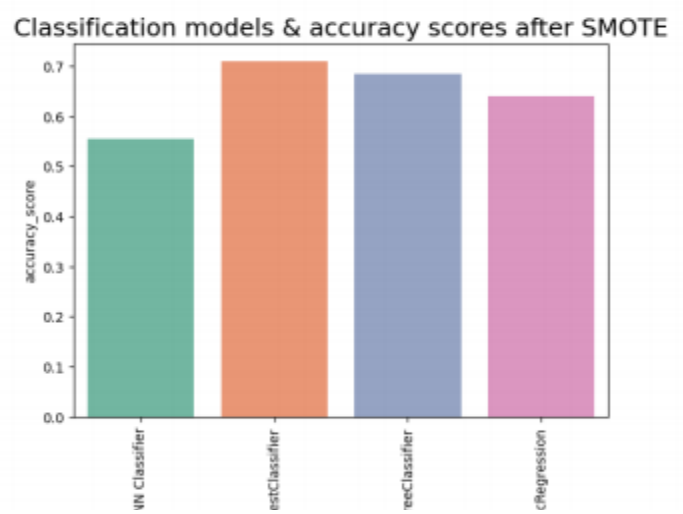



```
model4.predict([[50,1,1.2,0.8,150,70,80,7.2,3.4,0.8]])

array([1], dtype=int64)


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

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

```
Y_pred = classifier.predict(X_test)

4/4 [==============================] - 0s 2ms/step

Y_pred

Y_pred = (Y_pred > 0.5)
Y_pred

Y([[True],
  [True],
  [True],
  [True]. 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 105ms/step
Prediction: Liver Patient

acc_smote= [['KNN classifier',knn1],  ['RandomForestClassif
ier',rfc1],['DecisionTreeClassifier', dtc1],['LogisticRegre
ssion',logi1]]
Liverpatient_pred= pd.DataFarme(acc_smote, colums = ['Class
ification models', 'accuracy_score'])
Liverpatient_pred
```

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

```
plt.figure(figsize=(7,5)
plt.xticks(rotation=90)
plt.title('Classification models & accuracy scores after SM
OTE',fontsize=18 )
sns.barplot(x="Classification models", y="accuracy_score",
 data=Liverpatient_pred,palette ="set2")
```

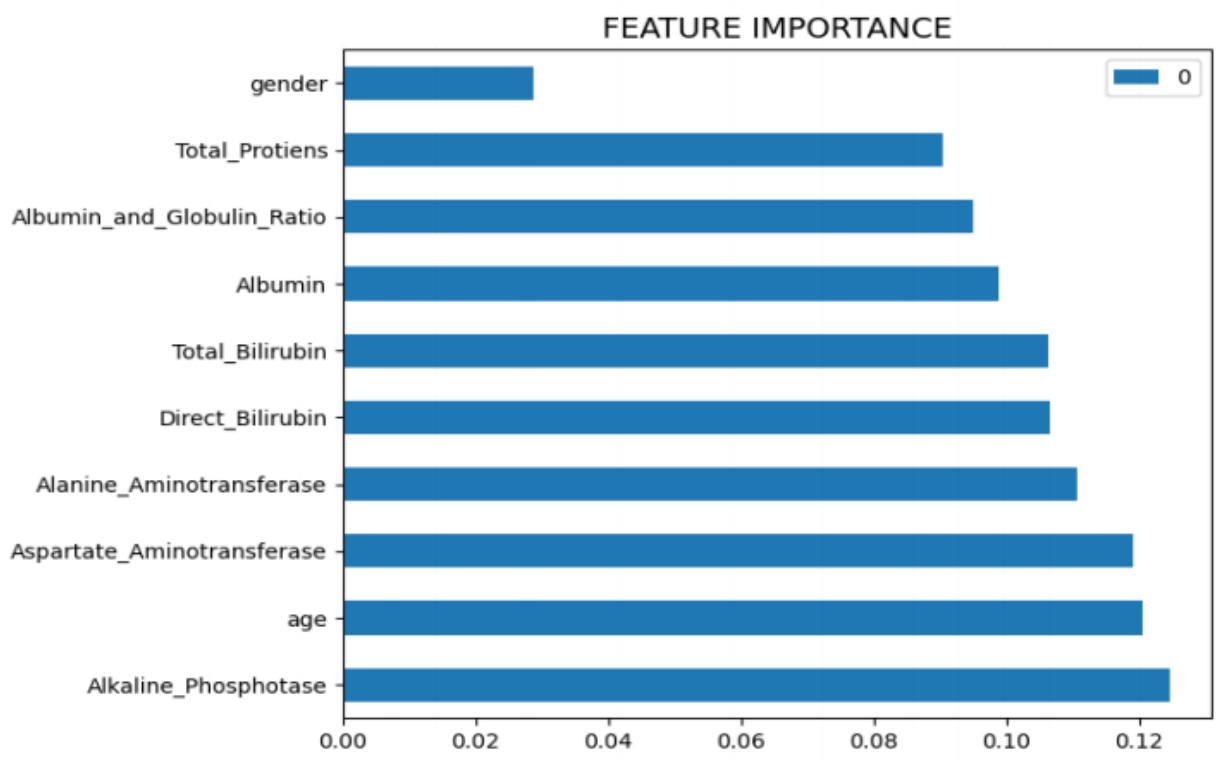



```
from sklearn.ensemble import ExtraTreesClassifier
model=ExtraTreesClassifier()
model.fit(x,y)

ExtraTreeClassifier()

model.feature_importances_

array([0.1205029 , 0.02863187 , 0.10625368 , 0.10648548 ,
0.1245292 , 0.11049943 , 0.118963 , 0.09033392 , 0.09882431
, 0.09497621])
```

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

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

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



```
import joblib
joblib.dumb(model1, 'ETC.pkl')

['ETC.pkl']

app=flash(_name_)
@app.route('/')
def home():
  return render_template('home.html')
@app.route('/predict')
def index()
  return render_template("index.html")

@app.route('/date_predict') , methods['POST']
def predict():
    age=request.form['age']
    age=request.form['gender']
    tb=request.form['tb']
    db=request.form['db']
    aa1=request.form['aa1']
    aa2=request.form['aa2']
    tp=request.form['tb']
    a=request.form['a']
    agr=request.form['agr']
    data=[[float(age),float(gender), float(tb), float(db),
float(ab), float(aa1), float(aa2), float(tp),
           model=pickle.load(open('liver_analysis.pkl', 'rb
'))
           prediction=model.prediction(date)[0]
           if(prediction==1):
              return render_template('nochance.html', predi
ction='you havea liver desease problem, you must and:
           else:
             return render_template('chance.html', predicti
on='you dont have a liver desease problem')
```

```
if_name_=='_main':
  app.run()
from flask import flash, render_template, request
import numpy as np
import pickle
```