RAJESWARI VEDACHALAM GOVERNMENT ARTS COLLEGE

CHENGALPATTU-603001

DEPARTMENT : BCA

A REVIEW OF LIVER PATIENT ANALYSIS METHOD USING MACHINE LEARNING

PRESENTED BY :

- AJITH KUMAR S
- VENUGOPAL
- MANIGANDAN T
- CHANDRU

INDEX

ABSTRACT

The motive of our project is to provide the maximum accuracy of the prediction of Liver diseases using various machine learning techniques. Data Mining is one in Every of the foremost vital aspects of automatic malady identification and malady prediction. It involves data processing algorithms and techniques to research medical knowledge. In the recent times, the percentage of liver disease patients have been numerously increasing and this liver related diseases have been a fatal diseases now-a-days. In this project we use various machine learning techniques to predict the liver disease for a person. In this project, we have to collect a liver disease patient dataset from the UCI repository. This project is implemented in three various modules. In the first module, we have to clean the dataset which we have collected from the UCI repository. Then we have to apply the min-max algorithm to the dataset. In the next phase, we will do the classification of the dataset which we have collected from the UCI repository. The next phase is the phase three where we will have to apply various machine learning techniques to predict liver disease.

INTRODUCTION

Machine Learning:

Machine learning has become one of the most evolving technologies in the current period. Machine learning can simply explained as scientific study of algorithms and models in statistics where machines can easily understand to perform and solve specific tasks. This technique has become agile and it has been a requirement in most of the fields.

Our motive:

Our motive for this project is to predict the liver diseases for a patient with the maximum amount of accuracy in our prediction. For this we have collected a dataset named Indian patient liver disease dataset from UCI repository and used that dataset in our three modules to predict the liver disease using various machine learning techniques.

Efficient Technique:

Here in our project we have identified a best algorithm which can give a better accuracy compared to other machine learning techniques using the liver dataset. The algorithm we have identified is gradient booster.

Increased Accuracy:

We had many algorithms in the past for detecting the disease but the algorithms which we have used in our project will increase the efficiency in predicting the liver disease. The machine learning algorithm called gradient booster technique gives us higher accuracy compared to others.

Data Mining:

Data extraction is the way to find designs in expansive informant indexes including A crossing point strategies, measurements and database frames. Information Mining is an interdisciplinary field of software engineering and measuring that aims to separate data from information collection (with keen strategies) and transform data into a understandable structure to be used further.

Data Pre-processing :

Data pre-processing is an important step to solve each problem of machine learning. In order for a machine learning algorithm to be trained, most of the data sets used with machine teaching problems must be processed. The techniques used most commonly for pre-processing are very few such as imputation of lack of value, categorical coding, scaling, etc. These are easy to understand techniques. Every dataset has its own unique challenges and is different. All features, except Gender are real valued integers. The last column, Disease, is the label (with ’1’ representing presence of disease and ’2’ representing absence of disease).

Dataset:

The Indian Liver Patient Dataset contained 10 distinct qualities of 583 patients. The patients were portrayed as either 1 or 2 based on liver sickness. The nitty gritty portrayal of the dataset is appeared Table. The table give insights regarding the trait and characteristic sort. As plainly unmistakable from the table, every one of the highlights with the exception of sex are genuine esteemed numbers. The component Sex is changed over to numeric esteem (0and 1) in the information pre-preparing step.

Data Collection:

Collection of data is crucial for these kind of projects. We have collected a dataset named as Indian Patient Liver Dataset from UCI repository which consists of 10 different attributes of 583 patients.

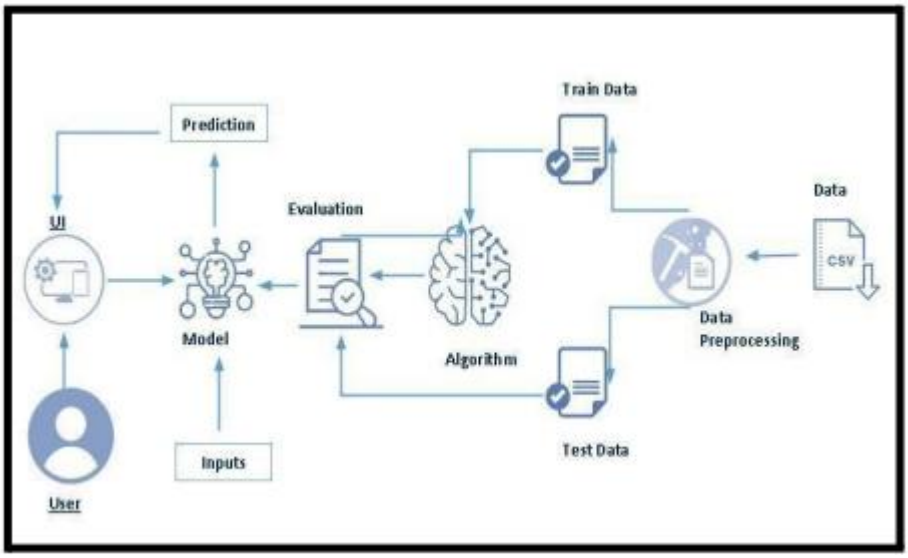
Data Cleaning:

We will have different columns which are called attributes. Some columns have null values and some values are fluctuated so we will clean those values from the datasheet and then take that dataset for classification.

A REVIEW OF LIVER PATIENT ANALYSIS METHODS USING MACHINE LEARNING

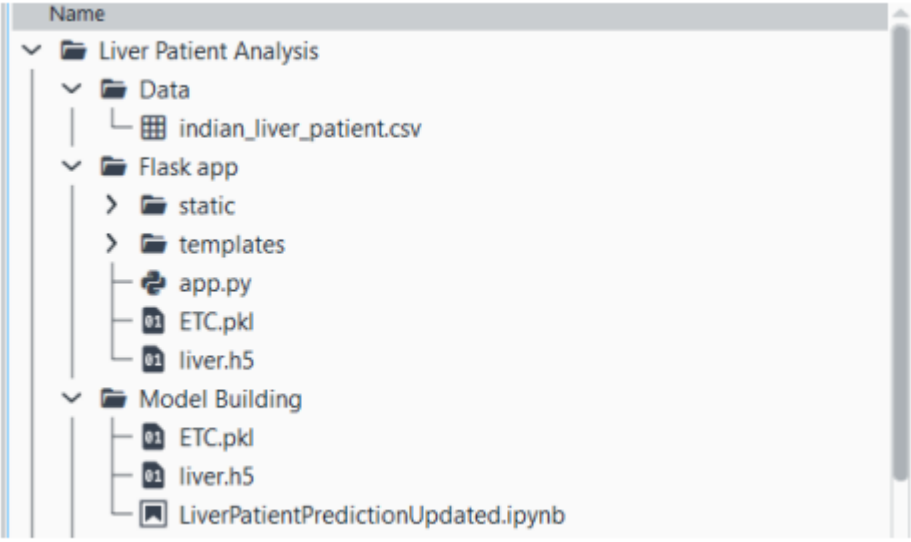
Liver diseases averts the normal function of the liver. This disease is caused by an assortment of elements that harm the liver. Diagnosis of liver infection at the preliminary stage is important for better treatment. In today's scenario devices like sensors are used for detection of infections. Accurate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

Technical Architecture:



- **Project Structure:**

Create the Project folder which contains files as shown below



● We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.

● ETC.pkl is our saved model. Further we will use this model for flask integration.

● Training folder contains a model training file.

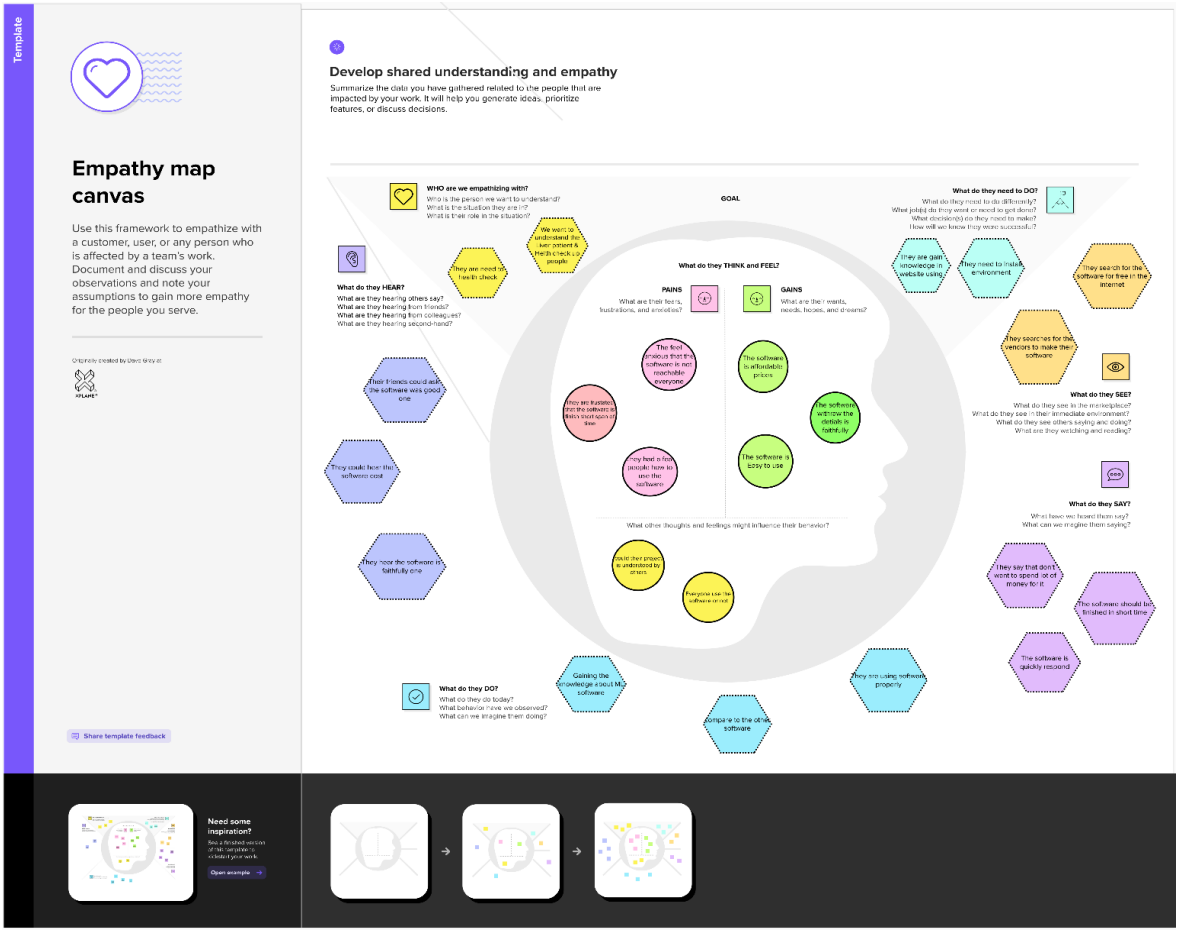
➢ Define Problem / Problem Understanding

- **Specify the business problem**

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. 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.

Empathy Map Canvas:

In the ideation phase we have empathized as our client bank and we haveacquired Details which are represented in the empathy map given below .An empathy map Is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to help teams beter understand their users.



Brainstorm & Idea Prioritization:

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

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

Share template feedback

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 →

1 Define your problem statement

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

5 minutes

PROBLEM

How might we [A Review of Liver Patient Analysis Methods using Machine Learning]

Key rules of brainstorming

To run an smooth and productive session

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

Need some inspiration?

See a finished version of this template to kickstart your work.

Open example →

2

Brainstorm

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

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Ajith

The software develop in uvigo (python programming language)

The software is Medical related one

The software provide the details in MHMU

Manigandan

The software is useful for everyone

The software is easy learn

The software is Unique

venugopal

To develop and implement the algorithm

Chandru

We can use pandas a lot numpy for data handling

Whether the customer statistics will our service.

This project provides a more comprehensive understanding of the models.

Person 5

Person 6

Person 7

Person 8

3

Group ideas

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

20 minutes

TIP

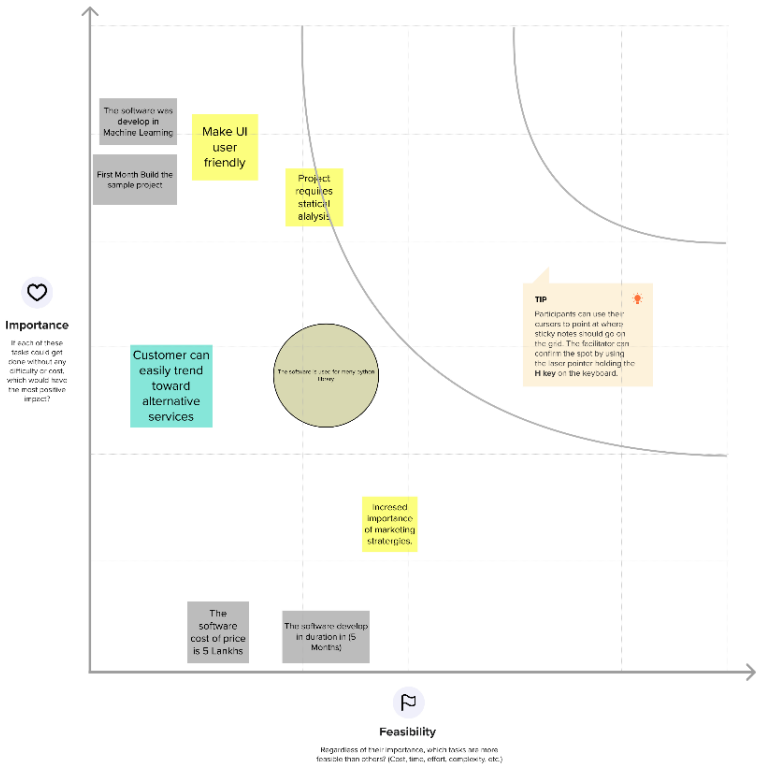
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes



After you collaborate

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

Quick add-ons

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 →

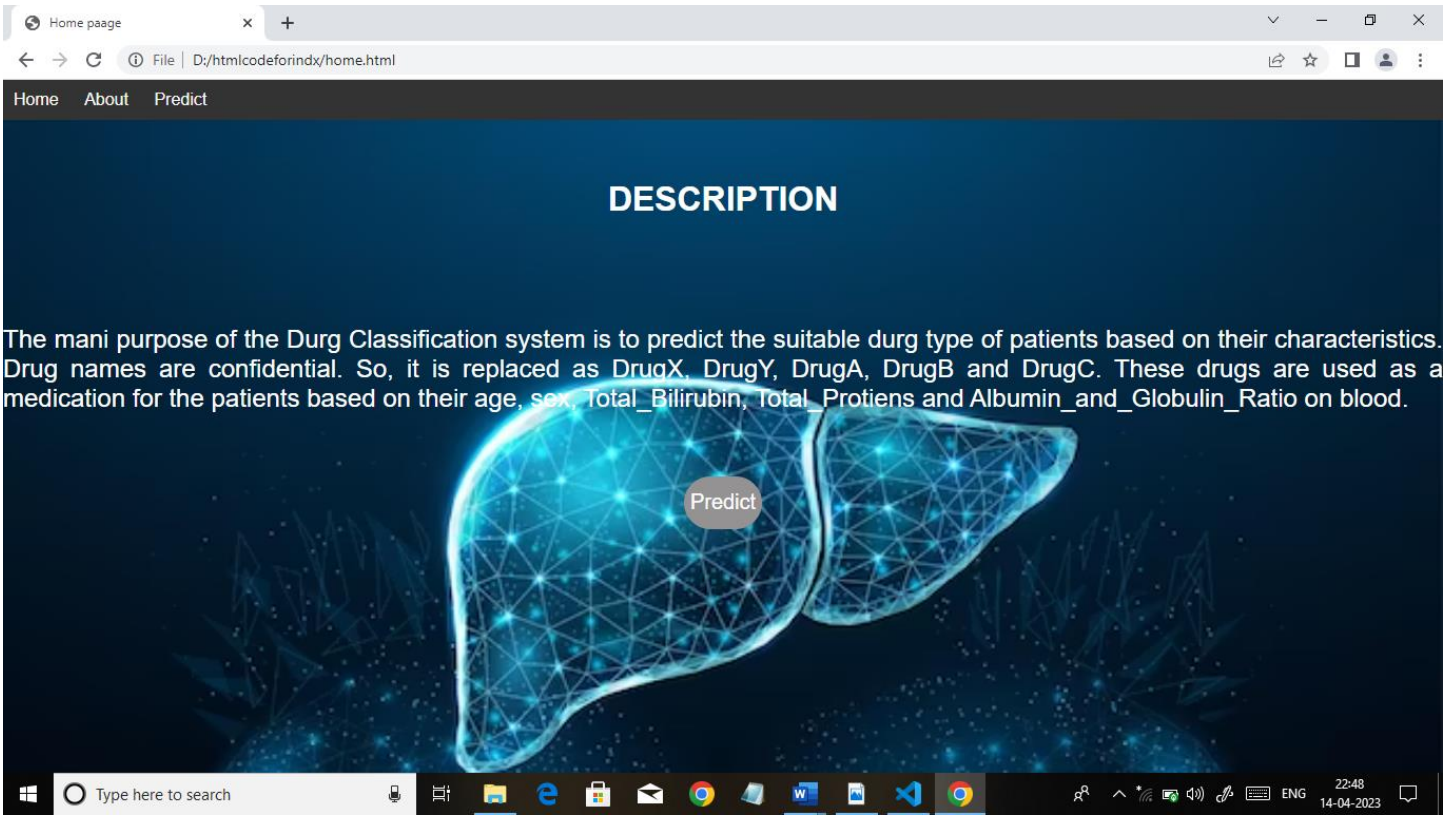
Share template feedback

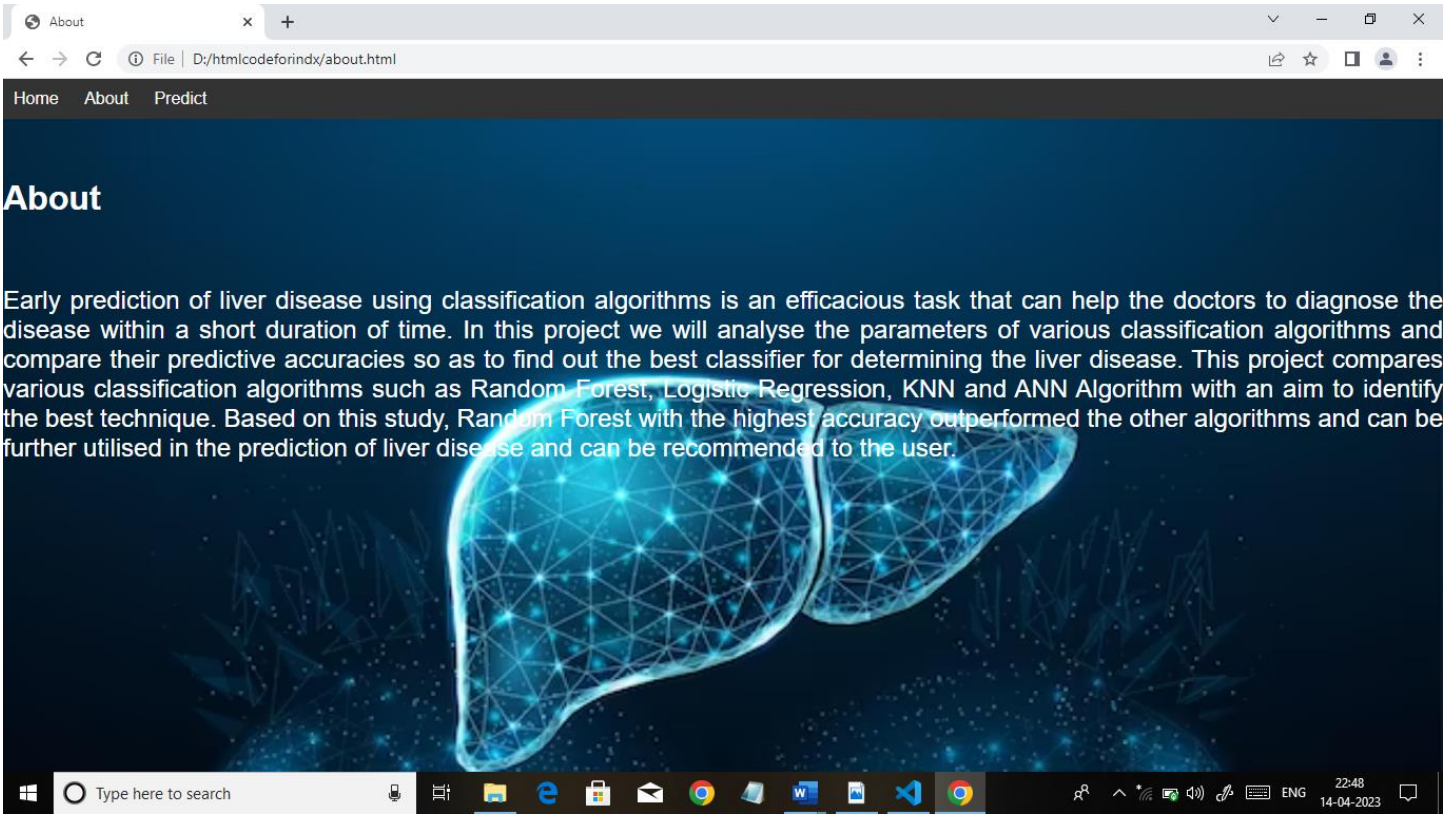
- **Business requirements**

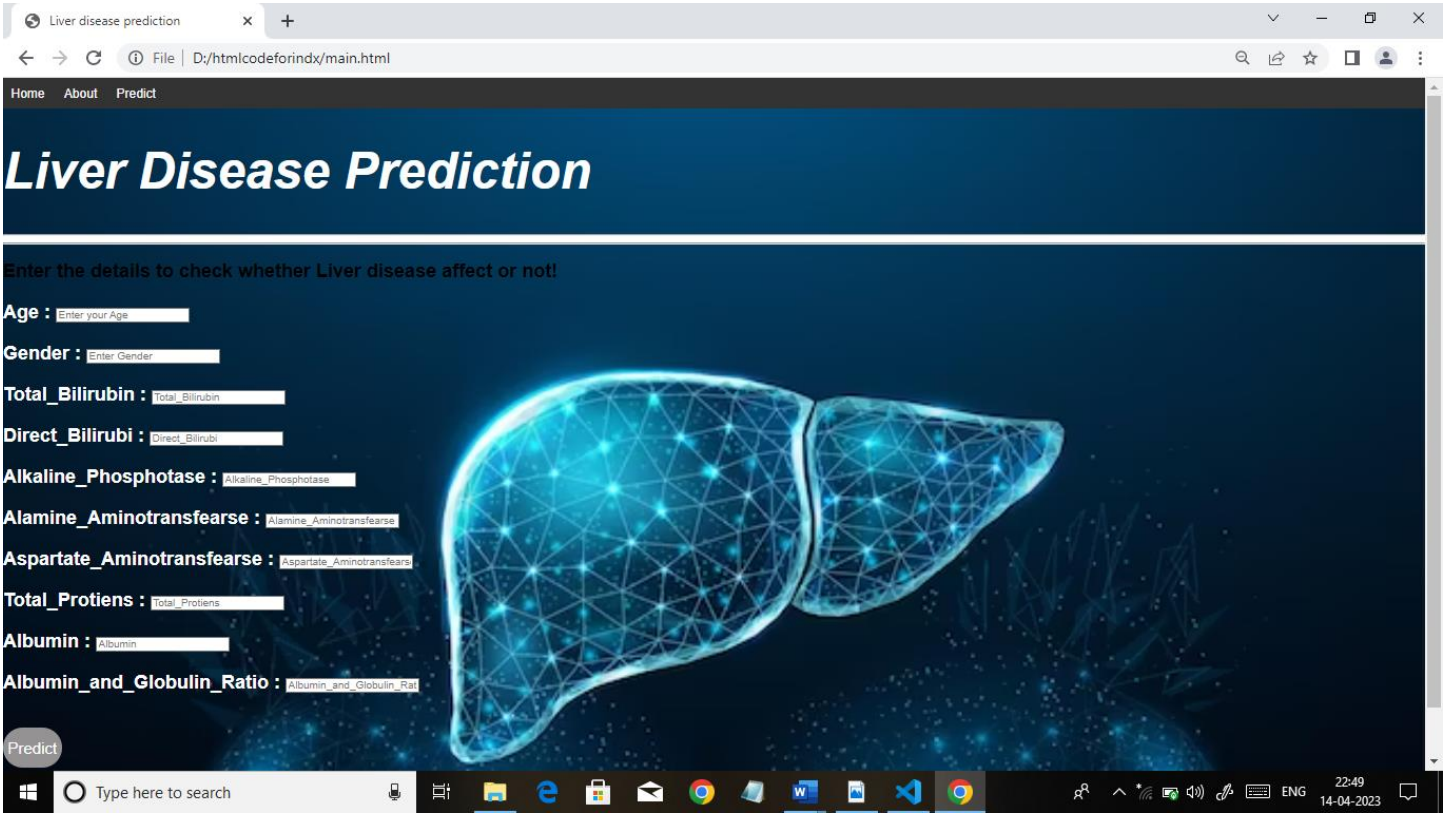
  Patients with Liver disease have been continuously increasing because of excessive consumption of alcohol, inhale of harmful gases, intake of contaminated food, pickles and drugs and other factors. This dataset was used to evaluate prediction algorithms in an effort to reduce burden on doctors. Use these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

- **Social or Business ImpactSocial Impact:-** Today almost everybody above the age of 12 years has smartphones with them, and so we can incorporate these solutions into an android app or ios app. Also it can be incorporated into a website and these app and website will be highly beneficial for a large section of society.
-
- **Business Model/Impact:-** Its now more feasible Blood test centers to give the result. As for this model user don't need to have any deep knowledge of medical science and liver diseases. User need to do pass the details being asked, which are already present in the blood test report(some like age, gender are already known) and then user will get the results of prediction.

RESULT




About
File | D:/htmlcodeforindx/about.html
Home
About
Predict
About
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.
Type here to search
ENG
22:48
14-04-2023


Liver disease prediction
File | D:/htmlcodeforindx/main.html
Home
About
Predict
Liver Disease Prediction
Enter the details to check whether Liver disease affect or not!
Age :
Enter your Age
Gender :
Enter Gender
Total_Bilirubin :
Total_Bilirubin
Direct_Bilirubi :
Direct_Bilirubi
Alkaline_Phosphotase :
Alkaline_Phosphotase
Alamine_Aminotransfearse :
Alamine_Aminotransfearse
Aspartate_Aminotransfearse :
Aspartate_Aminotransfears
Total_Protiens :
Total_Protiens
Albumin :
Albumin
Albumin_and_Globulin_Ratio :
Albumin_and_Globulin_Rat
Predict
Type here to search
ENG
22:49
14-04-2023

ADVANTAGES:

Statistically validated

- ✓ Acceptable discriminative ability
- ✓ Use of objective and widely available laboratory tests
- ✓ Inclusion of renal dysfunction
- ✓ Minimized "ceiling effect"
- ✓ Limited effect on post-LT mortality

DISADVANTAGES:

- ✓ Medical "urgency" score
- ✓ Similar discriminative ability as the CTP score
- ✓ Less convenient to use at the bedside, compared to CTP score
- ✓ Interlaboratory variations for measurement of serum creatinine and INR
- ✓ Serum creatinine provides only a rough estimation of renal function
- ✓ Systematic adverse female gender bias
- ✓ Further refinement is needed (e.g. inclusion of serum sodium, re-weighting of MELD score components)
- ✓ Exclusion of complications of cirrhosis, some of which warrant transplantation at low MELD scores (e.g. hepatic encephalopathy)
- ✓ Weak predictor of mortality after LT (exclusion of donor characteristics)
- ✓ Further statistical evaluation is needed (particularly in terms of calibration)

CONCLUSION

Diseases related to liver and heart are becoming more and 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 and are joining yoga classes, dance classes; still the sedentary lifestyle and luxuries that are continuously being introduced and enhanced; the problem is going to last long. So, in such a scenario, our project will be extremely helpful to the society. With the dataset that we used for this project, we got 100 % accuracy for SVM model, and though it might be difficult to get such accuracies with very large datasets, from this project's results, one can clearly conclude that we can predict the risk of liver diseases with accuracy of 90 % or more.

Today almost everybody above the age of 12 years has smartphones with them, and so we can incorporate these solutions into an android app or ios app. Also it can be incorporated into a website and these app and website will be highly beneficial for a large section of society

Through this project we have increased the efficiency of the prediction. We have increased the accuracy of the prediction algorithms where we have used different algorithms to predict the accuracy of the disease at different accuracy levels. We have used a specific dataset Indian liver patient dataset where we have 10 attributes and more than 500 patients data so it would be very useful and give best accuracy of the prediction.

FUTURE OF SCOPE

In the future it would help the doctors to easily detect the disease in a patient body and it could be developed further by using different datasets and also by using different algorithms. Through this project we have increased the efficiency of the prediction. We have increased the accuracy of the prediction algorithms where we have used different algorithms to predict the accuracy of the disease at different accuracy levels. This project

can also be developed to detect which kind of a liver disease a patient has and if it is conformed it can also be told at which percentage it is in the patient's body. This can be taken further by using various algorithms and also lot of datasets. This can also be done using artificial intelligence techniques using different tools.

APPENDIX

➢ Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

- **Collect the dataset**

  There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

  In this project we have used .csv data. This data is downloaded from kaggle.com.

  Please refer to the link given below to download the dataset.

  Link: https://www.kaggle.com/datasets/uciml/indian-liver-patient-records

  As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques

- **Importing the libraries**

  Import the necessary libraries as shown in the image. (optional) Here we have used visualisation style as fivethirtyeight.

```
1
2 import pandas as pd
3 import numpy as np
4 import seaborn as sns
5 import matplotlib.pyplot as plt
6 from matplotlib import rcParams
7 from scipy import stats
```

- **Read the Dataset**

  Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

  In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

```
In [32]: df=pd.read_csv('indian_liver_disease.csv')

In [33]: df.shape

Out[33]: (583, 11)

In [34]: df.columns

Out[34]: Index(['Age', 'Gender', 'Total_Bilirubin', 'Direct_Bilirubin',
       'Alkaline_Phosphotase', 'Alamine_Aminotransferase',
       'Aspartate_Aminotransferase', 'Total_Protiens', 'Albumin',
       'Albumin_and_Globulin_Ratio', 'Dataset'],
      dtype='object')

In [35]: df.head()
```

Out[35]:

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

.

- **Data Preparation**

As we have understood how the data is, let's pre-process the collected dataThe download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data

- **Handling missing values**

Let’s find the shape of our dataset first. To find the shape of our data, the df.shape method is used. To find the data type, df.info() function is used.

```
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
```

- For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function.

```
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
```

We can see that there are null values in the Albumin_and_Globulin_Ration Column.

Let us check how many numbers of null records present in the Closing Value column using sum() function.

```
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 the above code of analysis, we can infer that columns such as Albumin and Globulin Ratio is having the missing values, we need to treat them in a required way.

```
#checking for the missing data after cleaning data
data['Albumin_and_Globulin_Ratio'] = data.fillna(data['Albumin_and_Globulin_Ratio'].mode()[0])
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
```

- We will fill in the missing values in the numeric data type using the mean value of that particular column and categorical data type using the most repeated value.

- **Handling Categorical Values**

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using manual encoding with the help of list comprehension.

- In our project, for Gender, encoding is done.

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

➢ Exploratory Data Analysis

- **Descriptive statistical**

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```
data.describe()
```

	age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alanine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin
count	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000
mean	44.746141	3.298799	1.486106	290.576329	80.713551	109.910806	6.483190	3.141852
std	16.189833	6.209522	2.808498	242.937989	182.620356	288.918529	1.085451	0.795519
min	4.000000	0.400000	0.100000	63.000000	10.000000	10.000000	2.700000	0.900000
25%	33.000000	0.800000	0.200000	175.500000	23.000000	25.000000	5.800000	2.600000
50%	45.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000	3.100000
75%	58.000000	2.600000	1.300000	298.000000	60.500000	87.000000	7.200000	3.800000
max	90.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.600000	5.500000

- **Visual analysis**

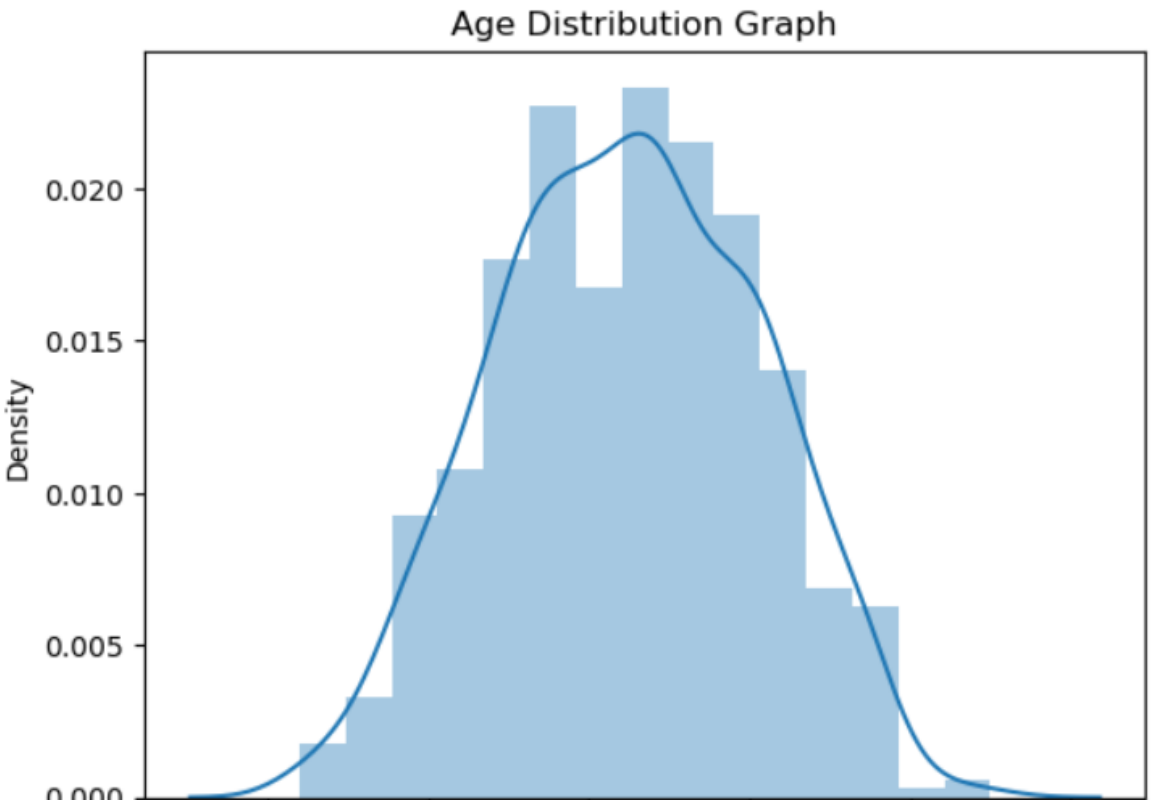
Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

- **Univariate analysis**

In simple words, univariate analysis is understanding the data with a single feature. Here we have displayed two different graphs such as distplot and countplot.The Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

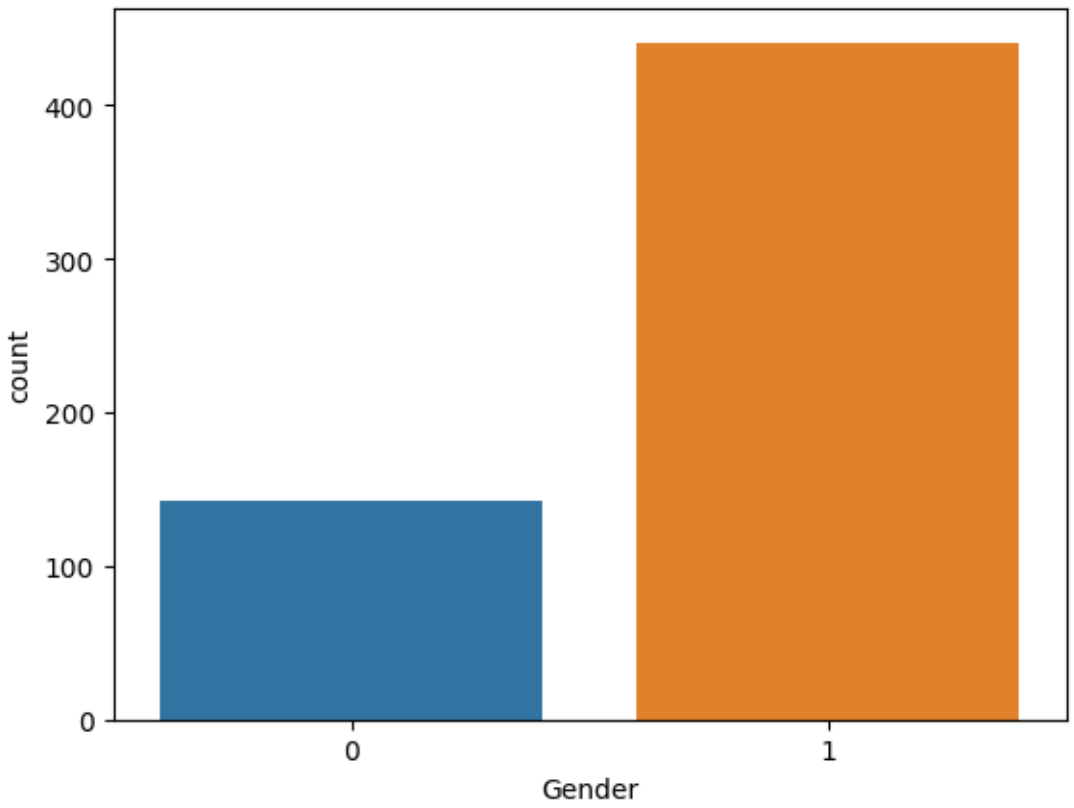
```
1 sns.distplot(data['age'])
2 plt.title('Age Distribution Graph')
3 plt.show()
4
```

```
D:\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a
oved in a future version. Please adapt your code to use either `displot` (a figure-level fun
`histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



- **Bivariate analysis**

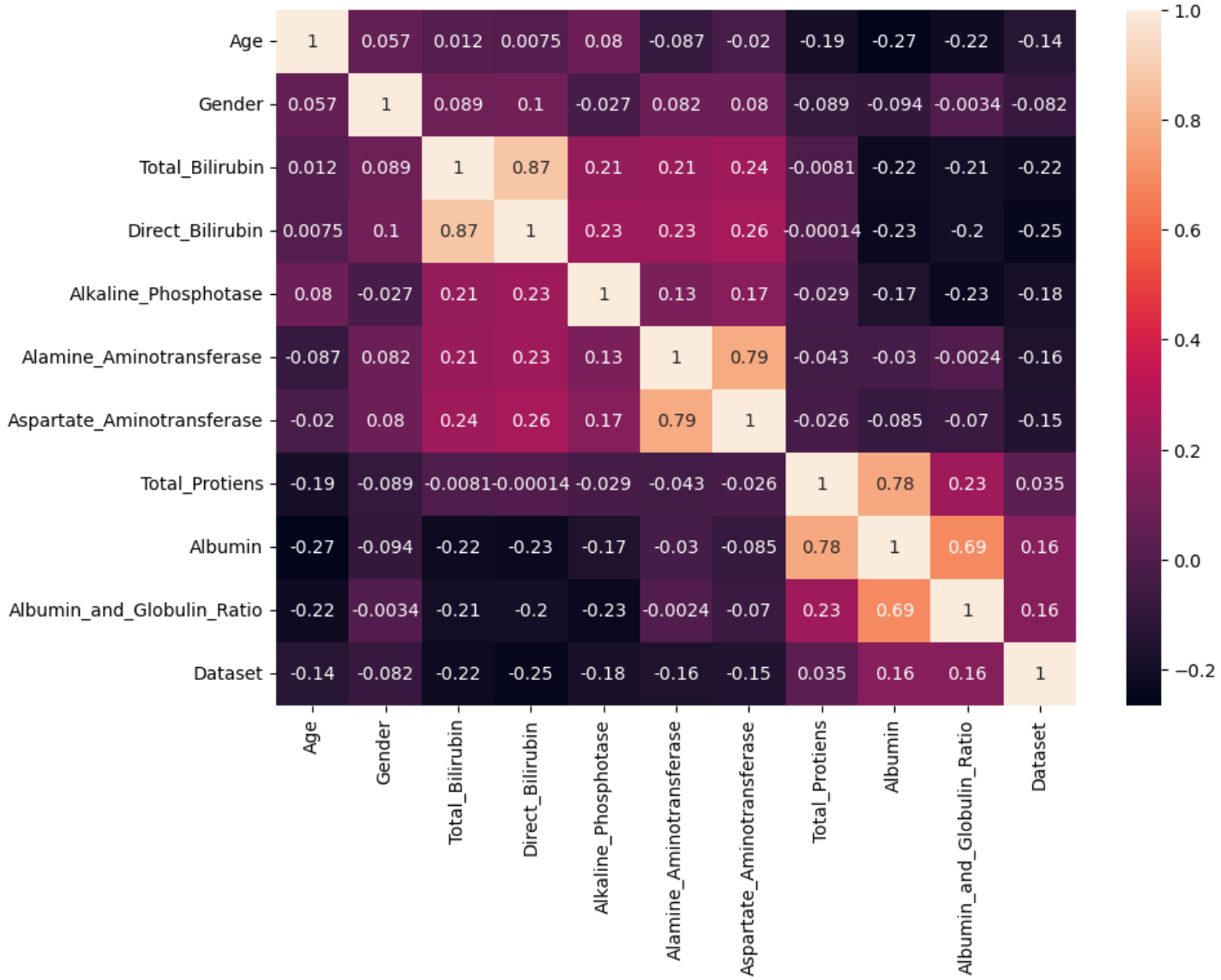
```
#df = sns.load_dataset('indian_liver_patient.csv')
sns.countplot(x=df['Gender'])
plt.show()
```



- **Multivariate analysis**

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used a heat plot from the seaborn package.

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



Now, the code would be normalising the data by scaling it to have a similar range of values, and then splitting that data into a training set and a test set for training the model and testing its performance, respectively.

- **Scaling the Data**

  Scaling is one the important process, we have to perform on the dataset, because of data measures in different ranges can leads to mislead in prediction Models such as KNN, Logistic regression need scaled data, as they follow distance based method and Gradient Descent concept.

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

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protie
0	1.252098	-1.762281	-0.418878	-0.493964	-0.426715	-0.354665	-0.318393	0.2921
1	1.066637	0.567446	1.225171	1.430423	1.682629	-0.091599	-0.034333	0.9375
2	1.066637	0.567446	0.644919	0.931508	0.821588	-0.113522	-0.145186	0.4765
3	0.819356	0.567446	-0.370523	-0.387054	-0.447314	-0.365626	-0.311465	0.2921
4	1.684839	0.567446	0.096902	0.183135	-0.393756	-0.294379	-0.176363	0.7531

We will perform scaling only on the input values.Once the dataset is scaled, it will be converted into an array and we need to convert it back to a dataframe.

- **Splitting data into train and test**

  Now let's split the Dataset into train and test sets

  Changes: first split the dataset into x and y and then split the data set

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

Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using the train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

```
from sklearn.model_selection import train_test_split

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

- **Handling Imbalance Data**

Data Balancing is one of the most important step, which need to be performed for classification models, because when we train our model on imbalanced dataset, we will get biased results, which means our model is able to predict only one class element. For balancing the data we are using the SMOTE Method.

SMOTE: Synthetic minority over sampling technique, which will create new synthetic data points for under class as per the requirements given by us using KNN method.

```
from imblearn.over_sampling import SMOTE

smote = SMOTE()

y_train.value_counts()

 1 316

 2 121
```

```
Name: Dataset, dtype: int64
x_train_smote, y_train_smote = smote.fit_resample(X_train,y_train)
y_train_smote.value_counts()
  1 316
  2 316
Name: Dataset, dtype: int64
```

From the above code we can infer that,previously our dataset had 316 class 1, and 121 class items, after applying smote technique on the dataset the size has become equal.

➢ Model Building

- **Training the model in multiple algorithms**

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

- **Random Forest model**

A function named RandomForestClassifier is imported and train and test data are passed as the parameters. Inside the function, RandomForestClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
# Random forest model
from sklearn.metrics import accuracy_score,classification_report
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(classification_report(y_test, y_predict))
```

- **Decision tree model**

A function named DecisionTreeClassifier is imported and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with.predict()

function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
# Decision Tree model
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))
```

- **KNN model**

A function named K KNeighborsClassifier is imported and train and test data are passed as the parameters. Inside the function, KNeighborsClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```
# KNN Model
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))
```

- **Logistic Regression model**

A function named Logistic Regression is imported and train and test data are passed as the parameters. Inside the function, Logistic Regression algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```
# Logistic Regression Model
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))
```

- **ANN model**

Building and training an Artificial Neural Network (ANN) using the Keras library with TensorFlow as the backend. The ANN is initialised as an instance of the Sequential class, which is a linear stack of layers. Then, the input layer and two hidden layers are added to the model using the Dense class, where the number of units and activation function are specified. The output layer is also added using the Dense class with a sigmoid activation function. The model is then compiled with the Adam optimizer, binary cross-entropy loss function, and accuracy metric. Finally, the model is fit to the training data with a batch size of 100, 20% validation split, and 100 epochs.

```
# ANN Model

import tensorflow as tf
from tensorflow import keras
from tensorflow.python import keras
from tensorflow.keras.models import Sequential
from tensorflow.python.keras.layers import Dense
classifier = Sequential()
classifier.add(Dense(units=100, activation='relu', input_dim=11))
classifier.add(Dense (units=50, activation='relu'))
classifier.add(Dense(units=1, activation='sigmoid'))
classifier.compile(optimizer='adam',loss='binary_crossentropy', metrics=['accuracy'])

# Fitting the ANN to the Training set
model_history = classifier.fit(X_train, y_train, batch_size=100, validation_split=0.2
, epochs=100)
```

```
Epoch 1/100
4/4 [==============================] - 2s 133ms/step - loss: 0.6497 - accuracy: 0.6532 - val_loss: 0.6394 - val_accuracy: 0.7
234
Epoch 2/100
4/4 [==============================] - 0s 21ms/step - loss: 0.6112 - accuracy: 0.7070 - val_loss: 0.6062 - val_accuracy: 0.72
34
Epoch 3/100
4/4 [==============================] - 0s 20ms/step - loss: 0.5901 - accuracy: 0.7016 - val_loss: 0.5800 - val_accuracy: 0.72
34
Epoch 4/100
4/4 [==============================] - 0s 20ms/step - loss: 0.5743 - accuracy: 0.7016 - val_loss: 0.5592 - val_accuracy: 0.72
34
Epoch 5/100
4/4 [==============================] - 0s 21ms/step - loss: 0.5619 - accuracy: 0.7016 - val_loss: 0.5437 - val_accuracy: 0.72
34
```

- **Testing the model**

```
#Age    Gender  Total_Bilrubin  Direct_Bilrubin Alkaline_Phosphotase    Alanin_Aminotransferase     Asparate_Aminotrans
model4.predict([[50,1,1.2,0.8,150,70,80,7.2,3.4,0.8]])
```

```
D:\Anaconda\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but DecisionTreeClass
ifier was fitted with feature names
  warnings.warn(
```

```
array([1], dtype=int64)
```

```
#Age    Gender  Total_Bilrubin  Direct_Bilrubin Alkaline_Phosphotase    Alanin_Aminotransferase     Asparate_Aminotrans
model1.predict([[50,1,1.2,0.8,150,70,80,7.2,3.4,0.8]])
```

```
D:\Anaconda\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but RandomForestClass
ifier was fitted with feature names
  warnings.warn(
```

```
array([1], dtype=int64)
```

```
#Age    Gender  Total_Bilrubin  Direct_Bilrubin Alkaline_Phosphotase    Alanin_Aminotransferase     Asparate_Aminotrans
model2.predict([[50,1,1.2,0.8,150,70,80,7.2,3.4,0.8]])
```

```
D:\Anaconda\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but KNeighborsClassif
ier was fitted with feature names
  warnings.warn(
D:\Anaconda\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction functions
(e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, thi
s behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is taken wi
ll be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

```
array([1], dtype=int64)
```

```
#Age    Gender  Total_Bilrubin  Direct_Bilrubin Alkaline_Phosphotase    Alanin_Aminotransferase     Asparate_Aminotrans
model5.predict([[42,0,1.2,0.8,240,70,80,7.2,3.4,0.8]])
```

```
D:\Anaconda\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but LogisticRegressio
n was fitted with feature names
  warnings.warn(
```

```
array([1], dtype=int64)
```

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

```
y([[ True],
   [ True],
   [ True],
   [ True],
```

This code defines a function named "predict_exit" which takes in a sample_value as an input. The function then converts the input sample_value from a list to a numpy array. It reshapes the sample_value array as it contains only one record. Then, it applies feature scaling to the reshaped sample_value array using a scaler object 'scale' that should have been previously defined and fitted. Finally, the function returns the prediction of the classifier on the scaled sample_value.

```python
def predict_exit(sample_value):

  # Convert list to numpy array
    sample_value = np.array(sample_value)

  # Reshape because sample_value contains only 1 record
    sample_value = sample_value.reshape(1, -1)

  # Feature Scaling
    sample_value = scale(sample_value)

    return classifier.predict(sample_value)
```

```python
#Age    Gender    Total_Bilrubin    Direct_Bilrubin    Alkaline_Phosphotase    A
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
```

➢ Performance Testing & Hyperparameter Tuning

- **Testing model with multiple evaluation metrics**

  Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using accuracy, score to compare between models

- **Compare the model**

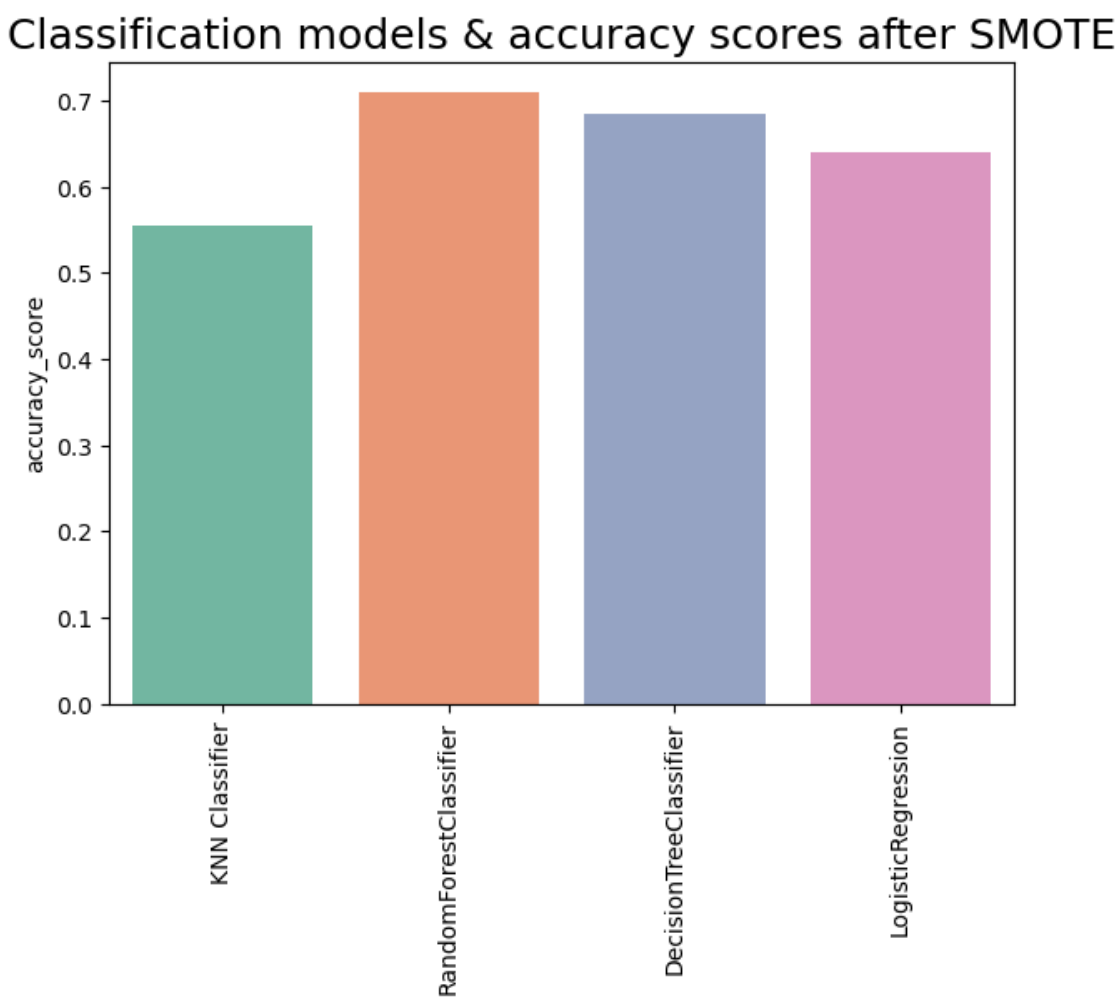
  For comparing the above four models, the Accuracy function is defined.

```
acc_smote= [['KNN Classifier', knn1],  ['RandomForestClassifier', rfc1],
              ['DecisionTreeClassifier', dtc1],['LogisticRegression',logi1]]
Liverpatient_pred= pd.DataFrame(acc_smote, columns = ['classification 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 SMOTE',fontsize=18)
sns.barplot(x="classification models", y="accuracy_score",  data=Liverpatient_pred,palette ="Set2")
```

```
<AxesSubplot:title={'center':'Classification models & accuracy scores after SMOTE'}, xlabel='classification
l='accuracy_score'>
```



After calling the function, the results of models are displayed as output. From the five models Random Forest Classifier is performing well. From the above image, We can see the

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

```
ExtraTreesClassifier()
```

```
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.columns).sort_values(0,ascending=False)
dd
```

	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

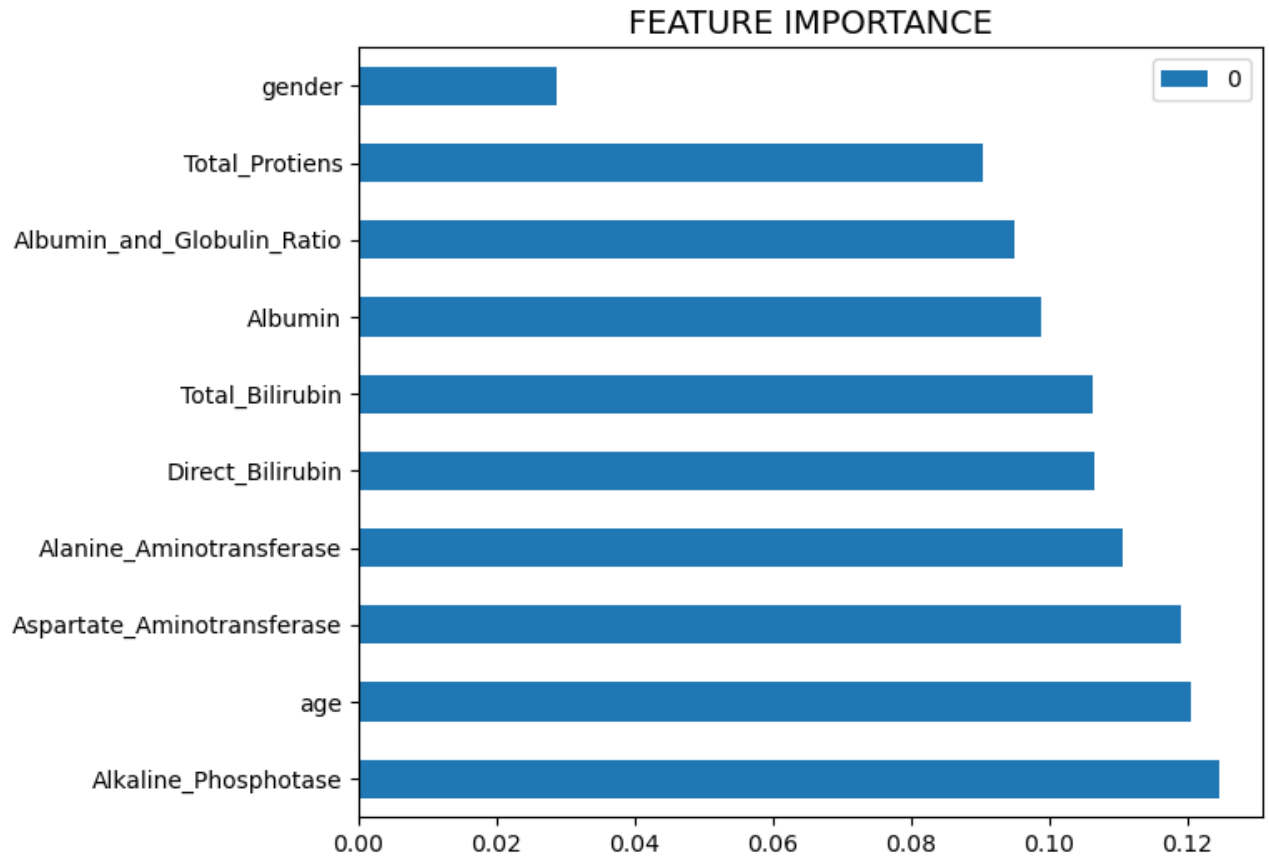
accuracy of the model. Random Forest Classifer is giving the accuracy of 70 percent.

- **Identifying Important Features**

10 attributes are passed to predict the actucal outcome, Its necessary to identify the l important feature to determin the output. Here we are using function called feature_importance to identify the important features among the available attributes and understand with a visualization.

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

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



Direct_Bilirubin & Total_Bilirubin are the most important features to predict the outcome

➢ Model Deployment

- **Save the best model**

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```
1 import joblib
2 joblib.dump(model1, 'ETC.pkl')
```

```
['ETC.pkl']
```