**Guardians of Liver Health: A Classifier Quest to Predict and Prevent Indian Liver Disease**

Introduction

Liver diseases are a significant public health concern worldwide, and India, with its diverse population and unique healthcare challenges, is no exception. Early detection of liver diseases is crucial for timely intervention and improved patient outcomes. This project aims to develop a predictive modeling framework tailored to the Indian context, utilizing a range of attributes to accurately assess the risk of liver disease in individuals.

The prevalence of liver diseases in India is influenced by several factors, including genetic predispositions, lifestyle choices, and regional variations. Leveraging machine learning and data analytics, this project seeks to harness the power of predictive modeling to analyze and interpret a comprehensive set of attributes. These attributes may include demographic information, lifestyle factors (such as diet and alcohol consumption), medical history, and relevant clinical markers.

By employing advanced statistical techniques and machine learning algorithms, the project aims to create a robust predictive model that can effectively stratify individuals based on their likelihood of developing liver diseases. This model is expected to provide valuable insights into the complex interplay of risk factors within the Indian population, aiding healthcare professionals in early identification and intervention.

The methodology involves collecting and analyzing a diverse dataset representative of the Indian population, encompassing individuals from different age groups, regions, and socio-economic backgrounds. The model development will include training, validation, and testing phases to ensure its accuracy and generalizability. Additionally, interpretability and transparency will be prioritized to enhance the model's utility in real-world clinical settings.

The outcomes of this project hold the potential to revolutionize preventive healthcare strategies in India. By offering a reliable tool for predicting liver disease risk, healthcare practitioners can implement targeted screening programs, allocate resources more efficiently, and enhance patient education regarding lifestyle modifications. Moreover, this predictive model may contribute to reducing the overall burden of liver diseases and improving the quality of healthcare delivery in the Indian context.

Table of Contents

Objective………………………………………………………………………………………………………... 4

Overview…………………………………………………………………………………………………………. 5

Data Analysis …………………………………………………………………………………………………. 8

Data Processing ……………………………………………………………………………………………… 12

Code ……………………………………………………………………………………………………………….12

Conclusion ……………………………………………………………………………………………………….42

Bibliography …………………………………………………………………………………………………… 43

Top of Form**Objectives**

The primary objectives of this project are:

1. **Model Development:** Create a robust predictive model that accurately assesses the risk of liver diseases in individuals based on a diverse set of attributes.
2. **Dataset Analysis:** Analyze a representative dataset from the Indian population, considering demographic, lifestyle, and clinical data to enhance the model's accuracy and applicability.
3. **Interpretability and Transparency:** Prioritize the development of a model that is not only accurate but also interpretable, enabling healthcare professionals to understand and trust its predictions.
4. **Contribution to Public Health:** Contribute valuable insights to the field of preventive healthcare in India by providing a tool that can aid in early identification and intervention for liver diseases.

**Scope of Documentation**

This documentation encompasses the various aspects of the project, including project planning, dataset analysis, model development, testing procedures, and potential applications. Each section is designed to provide comprehensive insights into the project's methodology, progress, and outcomes.

**Target Audience**

This documentation is intended for a diverse audience, including data scientists, healthcare professionals, project stakeholders, and anyone interested in the intersection of predictive modeling and public health in the context of liver diseases in India.

**Dataset Overview:**

The dataset appears to contain information related to liver health, with various attributes measured for each individual. Below is a detailed explanation of each column:

1. **Age:**
   * **Definition:** The age of the individual.
   * **Example:** The first row indicates an individual aged 65.
2. **Gender:**
   * **Definition:** The gender of the individual.
   * **Example:** The second row indicates a male individual.
3. **Total\_Bilirubin:**
   * **Definition:** Total bilirubin level in the blood (measured in mg/dL).
   * **Example:** The third row indicates a total bilirubin level of 7.3 mg/dL.
4. **Direct\_Bilirubin:**
   * **Definition:** Direct bilirubin level in the blood (measured in mg/dL).
   * **Example:** The third row indicates a direct bilirubin level of 4.1 mg/dL.
5. **Alkaline\_Phosphotase:**
   * **Definition:** Alkaline phosphatase enzyme level (measured in IU/L).
   * **Example:** The third row indicates an alkaline phosphatase level of 490 IU/L.
6. **Alamine\_Aminotransferase:**
   * **Definition:** Alamine aminotransferase (ALT) enzyme level (measured in IU/L).
   * **Example:** The third row indicates an ALT level of 60 IU/L.
7. **Aspartate\_Aminotransferase:**
   * **Definition:** Aspartate aminotransferase (AST) enzyme level (measured in IU/L).
   * **Example:** The third row indicates an AST level of 68 IU/L.
8. **Total\_Protiens:**
   * **Definition:** Total proteins in the blood (measured in g/dL).
   * **Example:** The third row indicates a total proteins level of 7.0 g/dL.
9. **Albumin:**
   * **Definition:** Albumin level in the blood (measured in g/dL).
   * **Example:** The third row indicates an albumin level of 3.3 g/dL.
10. **Albumin\_and\_Globulin\_Ratio:**
    * **Definition:** Ratio of albumin to globulin in the blood.
    * **Example:** The third row indicates a ratio of 0.89.
11. **Dataset:**
    * **Definition:** The target variable indicating whether the individual has liver disease (1) or not (2).
    * **Example:** The third row indicates that the individual has liver disease (Dataset value of 1).

**Dataset Characteristics**

* **Size:** The dataset contains a certain number of rows (examples) and columns (features).
* **Type of Data:** It is a structured dataset with both numerical and categorical variables.
* **Target Variable:** The "Dataset" column is the target variable for predictive modeling.

Data Analysis

A graph of different types of numerical data

Description automatically generated with medium confidence

A group of graphs showing different sizes of dots

Description automatically generated with medium confidence

A green and red bar graph

Description automatically generated

A diagram of a heatmap

Description automatically generated

A diagram of a graph

Description automatically generated with medium confidence

A graph of age distribution

Description automatically generated

Data Processing

A diagram of data processing

Description automatically generated

Code

import pandas as pd

import numpy as np

df=pd.read\_csv('indian\_liver\_patient.csv')

df

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

# Dataset column with 1 patient has liver disease and Dataset with 2 patient has no liver disease so mapping dataset column 1 as red means with liver disease and 2 as green means no disease

df['Dataset']=df['Dataset'].map({1:'red',2:'green'})

A screenshot of a computer screen

Description automatically generated

X = df.iloc[:,0:10]

y = df.iloc[:,-1]

print(X)

print(y)

A screenshot of a computer

Description automatically generated

Logistic Regression

#Applying classifier Logistic Regression

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X['Gender'] = le.fit\_transform(X['Gender'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.50)

model = LogisticRegression()

model.fit(X\_train,y\_train)

y\_predict\_LR = model.predict(X\_test)

accuracy\_LR = accuracy\_score(y\_test, y\_predict\_LR)

conf\_matrix\_LR = confusion\_matrix(y\_test, y\_predict\_LR)

class\_report\_LR = classification\_report(y\_test, y\_predict\_LR)

print(f'Accuracy: {round(accuracy\_LR\*100,2)}%')

print(f'\nConfusion Matrix:\n{conf\_matrix\_LR}')

print(f'\nClassification Report:\n{class\_report\_LR}')

A screenshot of a computer program

Description automatically generated

Decision Tree

#Applying classifier Decision Trees

from sklearn.tree import DecisionTreeClassifier

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.preprocessing import LabelEncoder

from sklearn import tree

import matplotlib.pyplot as plt

le = LabelEncoder()

X['Gender'] = le.fit\_transform(X['Gender'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.50)

model = DecisionTreeClassifier(criterion='entropy')

model.fit(X\_train,y\_train)

y\_predict\_DecisionTree = model.predict(X\_test)

plt.figure(figsize=(15,10))

tree.plot\_tree(model,filled=True)

accuracy\_DecisionTree = accuracy\_score(y\_test, y\_predict\_DecisionTree)

conf\_matrix\_DecisionTree = confusion\_matrix(y\_test, y\_predict\_DecisionTree)

class\_report\_DecisionTree = classification\_report(y\_test, y\_predict\_DecisionTree)

print(f'Accuracy: {round(accuracy\_DecisionTree\*100,2)}%')

print(f'\nConfusion Matrix:\n{conf\_matrix\_DecisionTree}')

print(f'\nClassification Report:\n{class\_report\_DecisionTree}')

A screenshot of a computer program

Description automatically generated

A diagram of a network

Description automatically generated

Random Forest

#Applying classifier Random Forest

from sklearn.ensemble import RandomForestClassifier

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.preprocessing import LabelEncoder

from sklearn import tree

import matplotlib.pyplot as plt

le = LabelEncoder()

X['Gender'] = le.fit\_transform(X['Gender'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.50)

model = RandomForestClassifier(criterion='entropy', n\_estimators=100, random\_state=42)

model.fit(X\_train,y\_train)

y\_predict\_RandomForest = model.predict(X\_test)

individual\_tree = model.estimators\_[0]

plt.figure(figsize=(30,20))

plot\_tree(individual\_tree, filled=True)

plt.show()

accuracy\_RandomForest = accuracy\_score(y\_test, y\_predict\_RandomForest)

conf\_matrix\_RandomForest = confusion\_matrix(y\_test, y\_predict\_RandomForest)

class\_report\_RandomForest = classification\_report(y\_test, y\_predict\_RandomForest)

print(f'Accuracy: {round(accuracy\_RandomForest\*100,2)}%')

print(f'\nConfusion Matrix:\n{conf\_matrix\_RandomForest}')

print(f'\nClassification Report:\n{class\_report\_RandomForest}')

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A diagram of a network

Description automatically generated

Naïve Bayesian

#Applying classifier Naive Bayesian

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X['Gender'] = le.fit\_transform(X['Gender'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.50)

model = GaussianNB()

model.fit(X\_train, y\_train)

y\_predict\_GaussianNB = model.predict(X\_test)

accuracy\_GaussianNB = accuracy\_score(y\_test, y\_predict\_GaussianNB)

conf\_matrix\_GaussianNB = confusion\_matrix(y\_test, y\_predict\_GaussianNB)

class\_report\_GaussianNB = classification\_report(y\_test, y\_predict\_GaussianNB)

print(f'Accuracy: {round(accuracy\_GaussianNB\*100,2)}%')

print(f'\nConfusion Matrix:\n{conf\_matrix\_GaussianNB}')

print(f'\nClassification Report:\n{class\_report\_GaussianNB}')

A screenshot of a computer program

Description automatically generated

Linear Discriminant

#Applying classifier Linear Discriminant

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X['Gender'] = le.fit\_transform(X['Gender'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.50)

model = LinearDiscriminantAnalysis()

model.fit(X\_train, y\_train)

y\_predict\_LinearDiscriminant = model.predict(X\_test)

accuracy\_LinearDiscriminant = accuracy\_score(y\_test, y\_predict\_LinearDiscriminant)

conf\_matrix\_LinearDiscriminant = confusion\_matrix(y\_test, y\_predict\_LinearDiscriminant)

class\_report\_LinearDiscriminant = classification\_report(y\_test, y\_predict\_LinearDiscriminant)

print(f'Accuracy: {round(accuracy\_LinearDiscriminant\*100,2)}%')

print(f'\nConfusion Matrix:\n{conf\_matrix\_LinearDiscriminant}')

print(f'\nClassification Report:\n{class\_report\_LinearDiscriminant}')

A screenshot of a computer program

Description automatically generated

Quadratic Discriminant

#Applying classifier Quadratic Discriminant

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.discriminant\_analysis import QuadraticDiscriminantAnalysis

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X['Gender'] = le.fit\_transform(X['Gender'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.50)

model = QuadraticDiscriminantAnalysis()

model.fit(X\_train, y\_train)

y\_predict\_QuadraticDiscriminant = model.predict(X\_test)

accuracy\_QuadraticDiscriminant = accuracy\_score(y\_test, y\_predict\_QuadraticDiscriminant)

conf\_matrix\_QuadraticDiscriminant = confusion\_matrix(y\_test, y\_predict\_QuadraticDiscriminant)

class\_report\_QuadraticDiscriminant = classification\_report(y\_test, y\_predict\_QuadraticDiscriminant)

print(f'Accuracy: {round(accuracy\_QuadraticDiscriminant\*100,2)}%')

print(f'\nConfusion Matrix:\n{conf\_matrix\_QuadraticDiscriminant}')

print(f'\nClassification Report:\n{class\_report\_QuadraticDiscriminant}')

A screenshot of a computer program

Description automatically generated

Linear SVM

#Applying classifier Linear SVM

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn import svm

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X['Gender'] = le.fit\_transform(X['Gender'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.50)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

model = svm.SVC(kernel='linear')

model.fit(X\_train\_scaled, y\_train)

y\_predict\_LinearSVM = model.predict(X\_test\_scaled)

accuracy\_LinearSVM = accuracy\_score(y\_test, y\_predict\_LinearSVM)

conf\_matrix\_LinearSVM = confusion\_matrix(y\_test, y\_predict\_LinearSVM)

class\_report\_LinearSVM = classification\_report(y\_test, y\_predict\_LinearSVM)

print(f'Accuracy: {round(accuracy\_LinearSVM\*100,2)}%')

print(f'\nConfusion Matrix:\n{conf\_matrix\_LinearSVM}')

print(f'\nClassification Report:\n{class\_report\_LinearSVM}')

A screenshot of a computer program

Description automatically generated

Gaussian SVM

#Applying classifier Gaussian SVM

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn import svm

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X['Gender'] = le.fit\_transform(X['Gender'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.50)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

model = svm.SVC(kernel='rbf')

model.fit(X\_train\_scaled, y\_train)

y\_predict\_GaussianSVM = model.predict(X\_test\_scaled)

accuracy\_GaussianSVM = accuracy\_score(y\_test, y\_predict\_GaussianSVM)

conf\_matrix\_GaussianSVM = confusion\_matrix(y\_test, y\_predict\_GaussianSVM)

class\_report\_GaussianSVM = classification\_report(y\_test, y\_predict\_GaussianSVM)

print(f'Accuracy: {round(accuracy\_GaussianSVM\*100,2)}%')

print(f'\nConfusion Matrix:\n{conf\_matrix\_GaussianSVM}')

print(f'\nClassification Report:\n{class\_report\_GaussianSVM}')

A screenshot of a computer program

Description automatically generated

SVM Degree 2

#Applying classifier SVM with degree 2

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn import svm

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X['Gender'] = le.fit\_transform(X['Gender'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.50)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

model = svm.SVC( kernel ='poly', degree =2)

model.fit(X\_train\_scaled, y\_train)

y\_predict\_SVMdegree2 = model.predict(X\_test\_scaled)

accuracy\_SVMdegree2 = accuracy\_score(y\_test, y\_predict\_SVMdegree2)

conf\_matrix\_SVMdegree2 = confusion\_matrix(y\_test, y\_predict\_SVMdegree2)

class\_report\_SVMdegree2 = classification\_report(y\_test, y\_predict\_SVMdegree2)

print(f'Accuracy: {round(accuracy\_SVMdegree2\*100,2)}%')

print(f'\nConfusion Matrix:\n{conf\_matrix\_SVMdegree2}')

print(f'\nClassification Report:\n{class\_report\_SVMdegree2}')

AdaBoost

#Applying classifier AdaBoost

from sklearn.ensemble import AdaBoostClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X['Gender'] = le.fit\_transform(X['Gender'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.50, random\_state=42)

base\_classifier = DecisionTreeClassifier(max\_depth=1)

model = AdaBoostClassifier(base\_classifier, n\_estimators=50, random\_state=42)

model.fit(X\_train, y\_train)

y\_predict\_adaboost = model.predict(X\_test)

accuracy\_adaboost = accuracy\_score(y\_test, y\_predict\_adaboost)

conf\_matrix\_adaboost = confusion\_matrix(y\_test, y\_predict\_adaboost)

class\_report\_adaboost = classification\_report(y\_test, y\_predict\_adaboost)

print(f'Accuracy: {round(accuracy\_adaboost \* 100, 2)}%')

print(f'\nConfusion Matrix:\n{conf\_matrix\_adaboost}')

print(f'\nClassification Report:\n{class\_report\_adaboost}')

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

Conclusion

In this project, we undertook an in-depth analysis of liver disease prediction using a diverse set of classifiers. The dataset, encompassing various demographic and clinical attributes, allowed us to explore the capabilities of different algorithms in predicting the presence or absence of liver disease.

A white paper with black text

Description automatically generated

**Key Findings**

* **Random Forest Outperformed:** Among the algorithms considered, Random Forest exhibited the highest accuracy, suggesting its effectiveness in managing the complexity and nuances of the dataset.
* **SVM Variants:** Both Gaussian SVM and Linear SVM performed well, highlighting the versatility of SVMs in capturing non-linear relationships.
* **Logistic Regression Stability:** Logistic Regression and SVM with Degree 2 provided consistent accuracy, emphasizing the stability of these models across different scenarios.

Bibliography

1. Kaggle Indian *Liver Disease Dataset.*
2. I conducted in-depth studies on various classifiers to enhance my project.