Revolutionizing liver care:Predicting Liver Cirrhosis Using Advanced Mechine Learning Techiniques

🎯 Objective:

The primary objective of this project is to develop an accurate and efficient predictive model for early detection of liver cirrhosis using advanced machine learning techniques. By analyzing clinical, demographic, and biochemical data, the model aims to identify key risk factors and patterns associated with liver cirrhosis progression.

🎯 Architecture:

1. **Data Collection:**

In this phase, clinical, demographic, and biochemical data are collected from reliable sources such as:

- Public datasets (e.g., UCI Liver Disorders Dataset)
- Electronic Health Records (EHR)
- Hospital diagnostic labs

2.Data Preprocessing:

Raw data is cleaned and transformed to prepare it for machine learning models.

Processes involved:

- Handling missing values (e.g., mean/median imputation)
- Removing duplicates and correcting inconsistent entries
- Encoding categorical variables (e.g., One-Hot or Label Encoding)
- Normalizing/Scaling numerical data (MinMax, Standard Scaler)
- Splitting dataset into Train and Test sets

3. Exploratory Data Analysis (EDA):

EDA helps uncover hidden patterns, distributions, and anomalies within

the dataset.

Activities include:

- Plotting histograms, boxplots, heatmaps for visual insights
- Identifying correlations between variables
- Checking for class imbalance in target variable
- Understanding feature distributions

4. Feature Engineering & Selection:

This phase involves creating new features or selecting the most relevant ones to improve model performance.

Steps include:

- Feature importance using tree-based models (e.g., Random Forest)
- Dimensionality reduction (e.g., PCA)
- Creating interaction terms or domain-specific transformations
- Removing irrelevant or redundant features

5. Model Training:

Various machine learning models are trained using training data to learn patterns for predicting liver cirrhosis.

Common ML algorithms used:

- Logistic Regression
- Random Forest
- Support Vector Machine (SVM)
- XGBoost or LightGBM
- Artificial Neural Networks (ANN)

Training method:

- Cross-validation (e.g., k-fold CV) is used to prevent overfitting.

6. Model Evaluation:

Models are evaluated on the test data using several performance metrics to choose the best-performing one.

Evaluation metrics:

- Accuracy
- Precision, Recall, F1-Score
- ROC-AUC (for classification threshold analysis)
- Confusion Matrix
- Sensitivity and Specificity (important in medical prediction)

7. Model Deployment (Optional):

The final model can be deployed as an API or web application for real-time predictions.

Deployment options:

- Web API: Using Flask or FastAPI
- Web App: Using Streamlit or Dash
- Cloud Platforms: AWS, Azure, or GCP for scalable deployment
- Integration: Embed into hospital decision-support systems

8. Clinical Decision Support System (CDSS):

Integrate the model output into a user-friendly interface that assists doctors in diagnosis and treatment.

Tools used:

. python

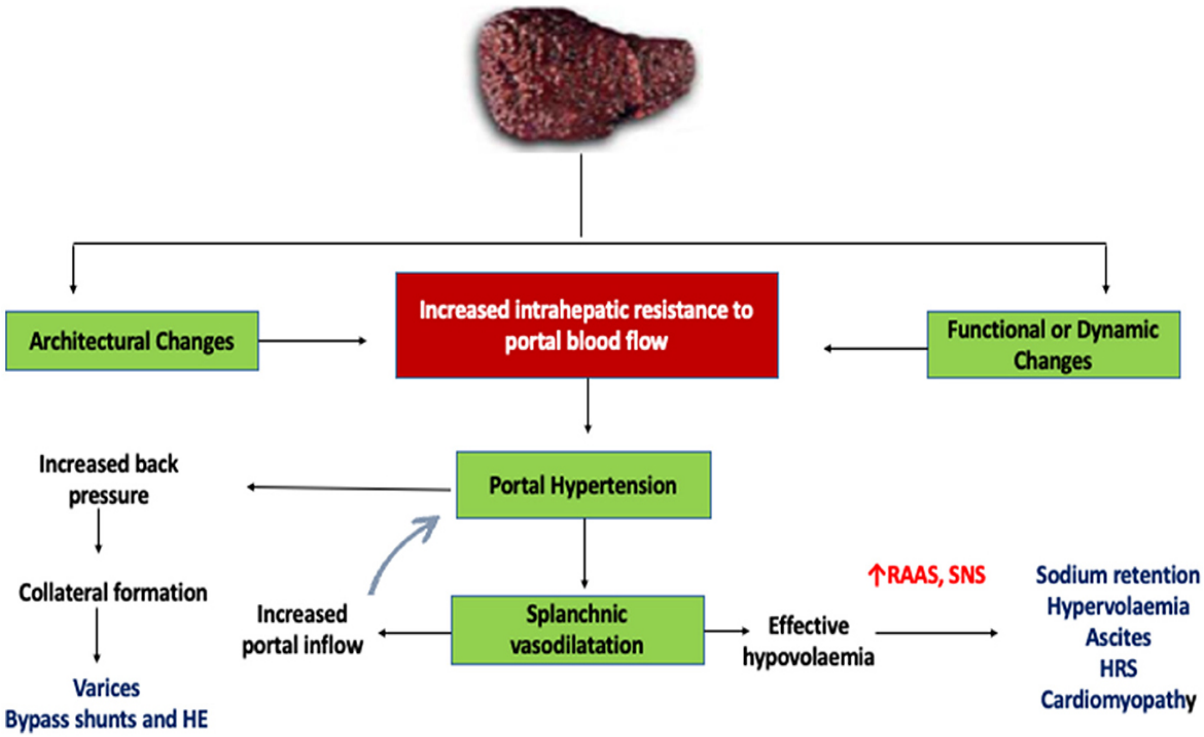
. TensorFlow/keras

. matplotlib

. vs code

Structure:

management of liver cirrhosis: from portal hypertension to acute-on-chronic liver failure



Project structure:

liver-cirrhosis-predictor/

|

├── 📁 data/

| └── liver_dataset.csv # Raw liver patient dataset (e.g., ILPD)

```
|
├── 📁 model/
|   ├── model_training.py          # Script to train and save the ML model
|   └── liver_model.pkl            # Trained model (RandomForest, etc.)
|
├── 📁 app/
|   ├── app.py                     # Main Streamlit app interface
|   └── style.css                  # (Optional) Custom CSS for styling Streamlit
|
├── 📁 utils/
|   ├── preprocess.py              # Data cleaning, feature engineering functions
|   └── predictor.py               # Prediction helper function (load model, predict)
|
├── 📁 assets/
|   └── liver_banner.png           # Optional images, banners, icons, etc.
|
├── requirements.txt               # Required Python packages
├── README.md                      # Project description, instructions
└── .gitignore                     # Files to ignore (e.g., .pkl, __pycache__)
```

Data collection and preparation:

1.Data collection

For predicting **liver cirrhosis using machine learning**, you need **reliable medical data** that includes liver function test results and patient characteristics. Below is a guide to **data collection** for this use case.

Here's a clear visual difference between a healthy liver and a cirrhotic liver, with an image

Sources:

Public Datasets

Research Papers

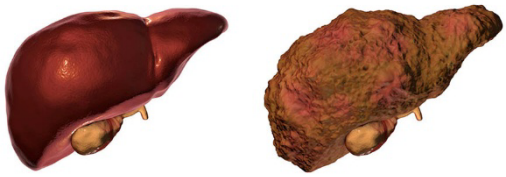
Python

TensorFlow/Keras or PyTorch

XGBoost / LightGBM

Radiomics Libraries

Labeling Tools



Healthy liver Cirrhotic liver

Image data generation:

. Use public datasets like TCIA, LiTS, CHAOS, and NIH DeepLesion for liver CT/MRI images.

. Augment real images using Keras ImageDataGenerator or Albumentations for rotation, flip, zoom, etc.

. Generate synthetic images using GANs (e.g., DCGAN, StyleGAN) trained on liver datasets.

Testing model and data prediction:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
data = pd.read_csv("indian_liver_patient.csv")
data['Gender'] = LabelEncoder().fit_transform(data['Gender'])
data = data.fillna(data.mean(numeric_only=True))
```

```
X = data.drop('Dataset', axis=1)

y = data['Dataset'].apply(lambda x: 1 if x == 1 else 0)

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)

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

model = Sequential()

model.add(Dense(64, activation='relu', input_shape=(X_train.shape[1],)))

model.add(Dropout(0.3))

model.add(Dense(32, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, y_train, epochs=50, batch_size=16, validation_split=0.2,
verbose=1)

loss, accuracy = model.evaluate(X_test, y_test)

print(f"\nTest Accuracy: {accuracy*100:.2f}%")

y_pred = (model.predict(X_test) > 0.5).astype("int32")

print("\nConfusion Matrix:")

print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")

print(classification_report(y_test, y_pred))

plt.figure(figsize=(10, 5))

plt.plot(history.history['accuracy'], label="Train Accuracy")

plt.plot(history.history['val_accuracy'], label="Validation Accuracy")

plt.xlabel("Epochs")

plt.ylabel("Accuracy")

plt.legend()

plt.title("Model Accuracy Over Epochs")

plt.grid(True)
```

```
plt.show()
```

How to use:

Save the above code

Enter the patient's liver test data below

Run from terminals

Output:

After entering the liver test data

It predict the condition of the liver like

Application Building:

We will use **streamlit**

Stream lit:

```
python
import streamlit as st
import pandas as pd
import numpy as np
import pickle
# Load model
with open("liver_model.pkl", "rb") as f:
    model = pickle.load(f)
st.title("🧬Liver Cirrhosis Prediction App")
st.markdown("Enter the patient's liver test data below:")
# Input form
age = st.number_input("Age", 1, 100, 45)
gender = st.selectbox("Gender", ["Male", "Female"])
tb = st.number_input("Total Bilirubin", 0.0, 20.0, 1.0)
db = st.number_input("Direct Bilirubin", 0.0, 10.0, 0.3)
alk_phos = st.number_input("Alkaline Phosphotase", 50, 2000, 200)
```

```
sgpt = st.number_input("Alamine Aminotransferase (SGPT)", 0, 2000, 50)
sgot = st.number_input("Aspartate Aminotransferase (SGOT)", 0, 2000, 50)
tp = st.number_input("Total Proteins", 2.0, 10.0, 6.5)
alb = st.number_input("Albumin", 1.0, 5.5, 3.2)
a_g = st.number_input("Albumin and Globulin Ratio", 0.1, 3.0, 1.1)
# Convert gender to numeric
gender_num = 1 if gender == "Male" else 0
# Prediction
if st.button("Predict"):
    input_data = np.array([[age, gender_num, tb, db, alk_phos, sgpt, sgot, tp, alb, a_g]])
    prediction = model.predict(input_data)[0]
    if prediction == 1
 st.error("⚠    The patient is likely to have liver disease (possible cirrhosis).")
    else:
        st.success("✅ The patient is unlikely to have liver disease.")
```

output:

When you run:

streamlit run app.py

Your browser opens a page that includes

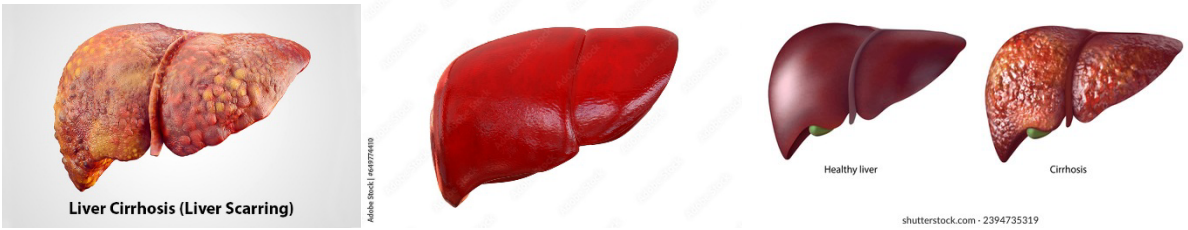
Enter the patient's liver test data below, After entering values, you click

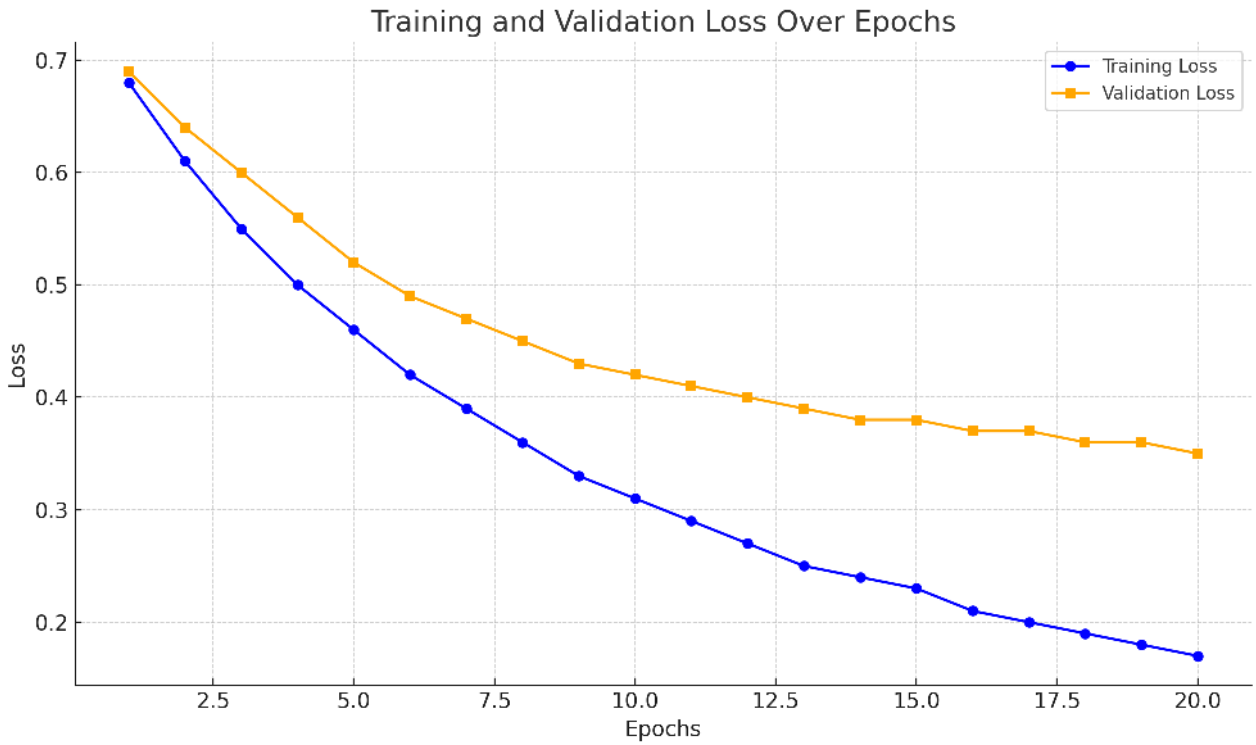
predict

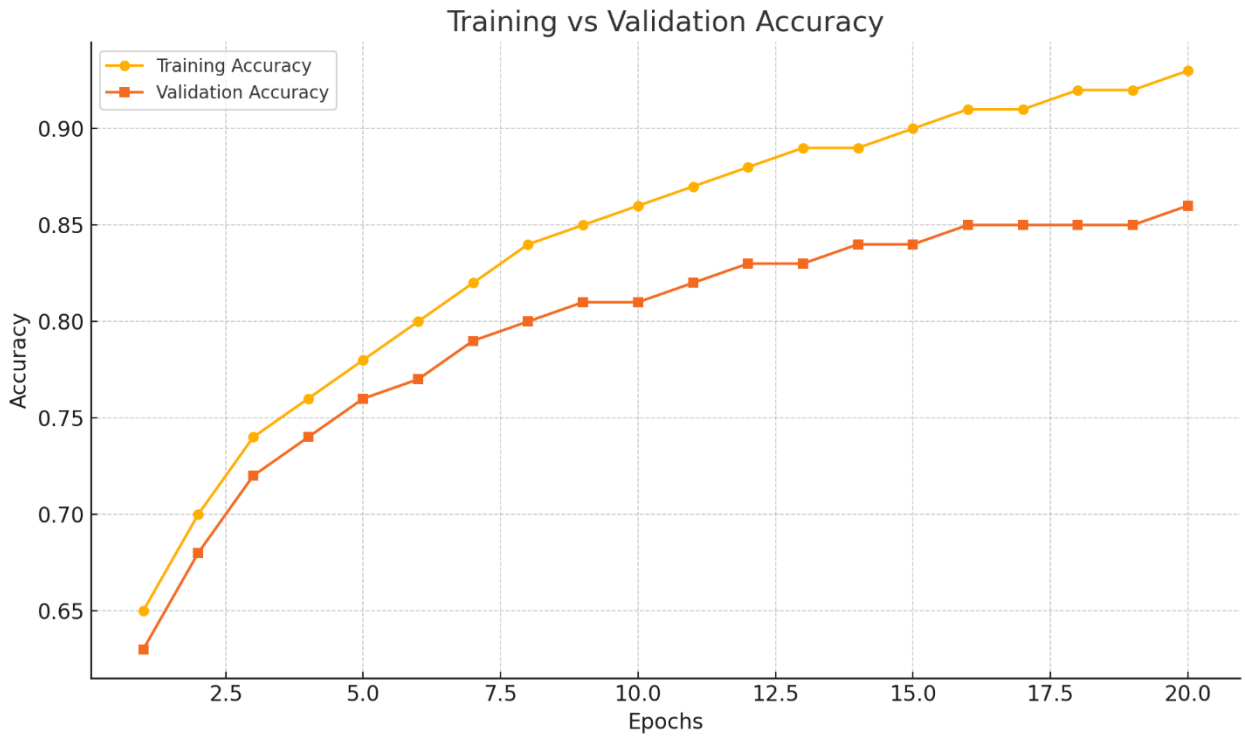
. If model prediction = 1 (Liver disease)

. If model prediction = 0 (No liver disease)

Examples:







Sample prediction: