

Revolutionizing Liver Care: Predicting Liver Cirrhosis Using Advanced Machine Learning Techniques

1. Introduction

- **Project Title:**
Revolutionizing Liver Care: Predicting Liver Cirrhosis Using Advanced Machine Learning Techniques
- **Team Members:**

Name	Role
Teki Satya Durga	Team Lead / Domain Expert
Sharun Pushpa Vardhineedi	Machine Learning Engineer
Surla Vagesh Kumar	Frontend Developer (React)
Vara Satya Ramya Bora	Backend Developer (Node.js)

2. Project Overview

- **Purpose:**
This project aims to harness the power of machine learning to **predict liver cirrhosis at an early stage** using patient health records and diagnostic parameters. It seeks to aid clinicians by providing a decision support system that improves diagnostic accuracy and reduces the burden of late-stage detection.
- **Features:**
 - User-friendly web interface for data entry and prediction.
 - Secure user authentication and role-based access.
 - Integration with ML model for cirrhosis prediction.
 - Admin dashboard to manage patient records.
 - Visual analytics and prediction results interpretation.

3. Architecture

- **Frontend (React):**
 - Built using React with React Router for navigation.
 - Axios used for RESTful API requests.
 - Styled using Tailwind CSS.

- Component-based structure (e.g., Nav bar, Prediction Form, Result Card).
 - **Backend (Node.js & Express.js):**
 - REST API to serve front end requests.
 - Routes for user management, prediction, and data analytics.
 - ML model integration using Python shell or API endpoint via Flask.
 - Middleware for error handling and token verification.
 - **Database (MongoDB):**
 - User schema: name, email, password (hashed), role.
 - Prediction schema: patient info, lab test results, prediction outcome.
 - Mongoose ODM used for schema modeling and queries.
-

4. Setup Instructions

- **Prerequisites:**
 - Node.js (v18+)
 - MongoDB (local or MongoDB Atlas)
 - Git
 - Python 3.8+ (for ML model)
 - npm or yarn
- **Installation Steps:**
- # Clone the repository
- `git clone https://github.com/your-org/liver-cirrhosis-prediction.git`
- `cd liver-cirrhosis-prediction`
-
- # Setup backend
- `cd server`
- `npm install`
-
- # Setup frontend
- `cd ../client`
- `npm install`
-

- # Setup Python ML model (if applicable)
 - `cd ../ml-model`
 - `pip install -r requirements.txt`
 - **Environment Variables (.env in server folder):**
 - `PORT=5000`
 - `MONGO_URI=mongodb+srv://<username>:<password>@cluster.mongodb.net/db`
 - `JWT_SECRET=your_jwt_secret_key`
 - `PYTHON_SCRIPT_PATH=../ml-model/predict.py`
-

5. Folder Structure

- **Client (React):**
- `client/`
- `├── src/`
- `| ├── components/`
- `| ├── pages/`
- `| ├── services/`
- `| └── App.js`
- `| └─ index.js`
- `└── public/`
- `└─ package.json`
- **Server (Node.js):**
- `server/`
- `├── controllers/`
- `├── routes/`
- `├── models/`
- `├── middleware/`
- `├── utils/`
- `└── server.js`
- `└─ .env`
- **ML Model (Python):**

- ml-model/
 - └─ predict.py
 - └─ model.pkl
 - └─ requirements.txt
-

6. Running the Application

- **Front end:**
 - cd client
 - npm start
 - **Back end:**
 - cd server
 - npm start
 - **ML Model Server (optional if standalone Flask app):**
 - cd ml-model
 - python app.py
-

7. API Documentation

Endpoint	Method	Description	Request Body / Params	Sample Response
/api/auth/register	POST	Register a new user	{ name, email, password }	{ token, user }
/api/auth/login	POST	Login user	{ email, password }	{ token, user }
/api/predict	POST	Submit data for prediction	{ age, bilirubin, ... }	{ prediction: 'Yes' }
/api/records	GET	Get all predictions (admin)	JWT Token	[{id, patient, result}]

8. Authentication.

- Tokens stored in **local Storage**.
 - Protected routes with middleware validation.
 - Roles: admin, user – used to control access to certain features like user management or analytics.
-

9. User Interface


(Add images in actual README or documentation PDF)

- **Login/Register Screens**
 - **Prediction Form** – input patient data
 - **Prediction Result View**
 - **Admin Dashboard** – view all patient records and results
 - **Visual Analytics** – charts and trends
-

10. Testing

- **Tools Used:**
 - Jest (unit testing for back end logic)
 - React Testing Library (component testing)
 - Postman (manual API testing)
 - PyTest (for Python model testing)
 - **Strategy:**
 - Unit tests for validation, utility functions.
 - Integration tests for REST APIs.
 - Snapshot/UI testing for React components.
-

11. Screenshots or Demo

-  *Screenshots of Key Pages:*
 - Login Page
 - Prediction Form
 - Prediction Results
 - Admin Dashboard
-

12. Known Issues

- Response time may increase with large datasets.
- Limited cross-browser testing.
- Mobile view not fully optimized.
- No support for bulk upload of patient records.

13. Future Enhancements

- Add support for **bulk CSV upload** for mass predictions.
- Improve model accuracy with more training data.
- Implement **multi-language support**.
- Create a mobile app version using React Native.
- Add visual explanation of predictions using **SHAP values** or **LIME**.