Artificial Intelligence & Machine Learning

1. Introduction

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

1. Muddala Vasu (Data Collection,Model Evaluation)
2. Shaik Abdul Baseed (Deployment,Dser Interface Testing)
3. Rohi Jacinth (Data Loading,User Interface Testing)
4. K Gnaneswar Reddy (Data Preprocessing),
5. Vemuru Pavan Kumar (Model Building)

2. Project Overview

- **Purpose:** This project predicts the likelihood of liver cirrhosis based on patient health data using a trained machine learning model. It aims to assist with **early screening** and **medical decision support**..
- **Features:**
- 36-field input form for medical attributes
- Machine learning model (Random Forest) integrated via Flask
- Animated result display with interpretation
- Input validation and normalization (L1 norm)
- Result shown on a dedicated result page
- Future-ready for login, user tracking, and result history

3. Architecture

- **Frontend:**
- Built with **HTML, CSS, and JavaScript**
- Main form in `index.html` with 36 clinical input fields
- Animated design with custom styles and optional JavaScript enhancements
- Navigation includes: `intro.html` → `index.html` → `result.html`

- **Backend:**

 Flask (Python) handles:

- Routing (/, /predict)
- Model loading and inference
- Input collection and preprocessing

 Uses joblib to load:

- rf_acc_68.pkl (Random Forest model)
- normalizer.pkl (L1 normalizer)

4. Setup Instructions

- **Prerequisites:**

    1. Python 3.x
    2. Flask
    3. scikit-learn
    4. NumPy
    5. joblib

- **Installation:**

    Step 1: Create virtual

environment python -m

venv .venv Step 2: Activate the

environment

venv\Scripts\activate

5. Folder Structure

```
Liver_Cirrhosis_Predictor/
├── app.py                    # Flask application
├── rf_acc_68.pkl             # Trained Random Forest model
├── normalizer.pkl            # L1 normalizer
├── Training/
│   └── model_trainin.ipynb   # Jupyter notebook for model training
├── Flask/
│   ├── templates/
│   │   ├── intro.html
│   │   ├── index.html
│   │   └── result.html
│   └── static/
│       ├── style.css
```

```
│   ├── demo2.jpg          # Optional background image
│   └── JS / other assets
```

6. Running the Application

- **Frontend:** `npm start` in the client directory.
- **Backend:** `npm start` in the server directory.

Flask - Python app.py

7. API Documentation

POST /predict Request Body: {

'features': [36 clinical input values] }

Response:

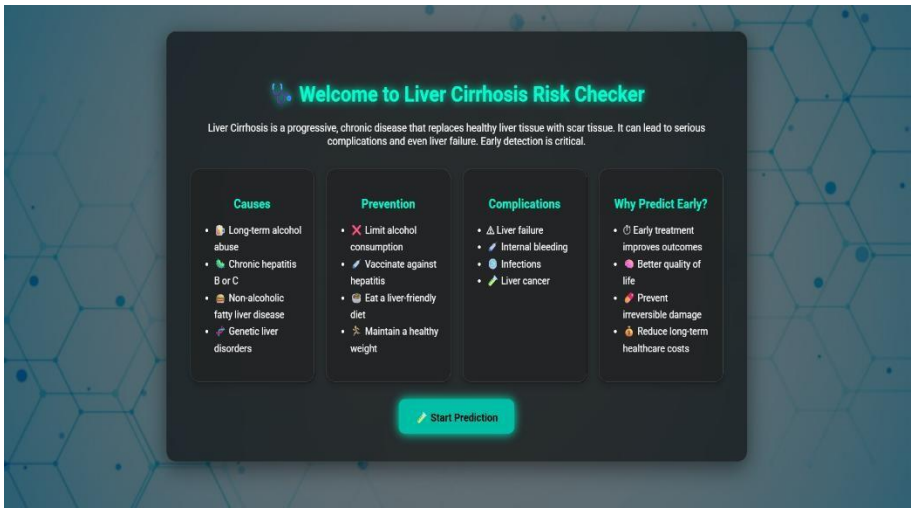
{ 'prediction': 0 or 1, 'message': '' }

8. Authentication

JWT-based auth can be optionally added for user login and saving prediction history

9. User Interface

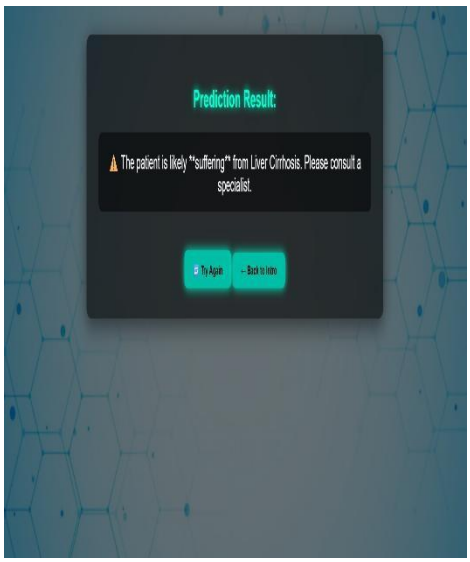
Clean form UI with animated result display and responsive design. Includes 3 pages: Intro,Index,Result



10. Testing

Manual testing via form submission and prediction checking. Postman used for backend API testing

11. Screenshots or Demo



Demo Link : https://drive.google.com/file/d/11jIoTQBYwb5ruxlaNGLmiwzTTJo7HdL/view?usp=drivesdk

12. Known Issues

Model doesn’t show probability/confidence.
Long form may feel overwhelming**.**

13. Future Enhancements

- Add login and user-based prediction history.
- Integrate SHAP or LIME to explain model decisions.
- Enable PDF download or email of results.
- Multi-language support.
- Convert form to progressive multi-step interface