

# 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 ID:** LTVIP2025TMID38326

**Team Size:** 5

- **Team Leader:** V Mounika Jaya – Project Coordination, Milestone Tracking, Model Building, Final Validation
- **Team Member:** Modhukuru Kavya – Data Collection, Cleaning, preprocessing pipeline
- **Team Member:** M Sanjana – Define Problem /Problem Understanding
- **Team Member:** G Chandana – ML Model Development & Flask Integration
- **Team Member:** Gaddam Lakshmi Bindhu – UI/UX Design, Documentation, Descriptive Statistics

## 2. Project Overview

### 2.1. Purpose

Liver cirrhosis is a critical health concern globally, leading to chronic liver failure and high mortality rates. This project aims to build a machine learning model to predict liver cirrhosis based on various patient features. By analysing these features, the model will classify patients into risk categories, aiding in early diagnosis and treatment.

### 2.2. Features

- User input form with 41 clinical features
- Prediction using a trained Random Forest model •  
Result displayed in a user-friendly web interface
- Error handling for missing or invalid data.
- Instantly returns a prediction label – "Likely Cirrhosis" or "Not Likely Cirrhosis" – after form submission.

## 3. Architecture

### 3.1. Frontend

- Developed using HTML5, CSS3, and Jinja2 templates.
- HTML form (index.html) for inputting 41 patient features
- Styled using embedded CSS and images served from the /static directory
- Features a clean, responsive UI with gradient backgrounds and conditionally rendered prediction results.

### 3.2. Backend

- Backend logic implemented using **Python and Flask**
- **Model Training Pipeline:**
- Model training, evaluation, and export handled in Google Colab.
- Scripts for data cleaning, feature engineering, and training were executed in Colab.
- Final .pkl files (model + tools) were trained and exported in Colab, then used in the Flask backend for real-time predictions.
- **Model:**
  - A Random Forest Classifier trained on a liver health dataset. Model artifacts include:
- liver\_prediction.pkl: Trained model
- label\_encoders.pkl: LabelEncoder instances
- normalizer.pkl: MinMaxScaler instance

### 3.3. Data Preprocessing:

- Normalization with MinMaxScaler
- Label Encoding for categorical variables
- Blood pressure parsing and transformation
- Encoders and scalers saved as .pkl files

### 3.4. Database

- No persistent database is used
- Prediction is done entirely in-memory based on form input

## 4. Setup Instructions

### 4.1. Prerequisites

- Python 3.10+
- Flask
- scikit-learn
- pandas, numpy
- Google Colab (or Jupyter Notebook)

### 4.2. Installation

install -r requirements.txt

```
python app.py
```

## 5. Folder Structure

### 5.1. Client (Flask Frontend)

The frontend is handled via HTML templates and static assets:

```
|— templates/
|   |— index.html      # Main HTML form for user input
|— static/
|   |— images/
|       |— liver-bg.jpg    # Background image for the form UI
|       |— healthy_liver.png
|       |— warning_liver.png
```

- The index.html file contains a full HTML form with 41 input fields styled with CSS.
- Static assets like background images and CSS files are stored in the static/ folder.

### 5.2. Server (Flask Backend)

The backend is built using Python and Flask:

```
|— app.py              # Flask application script
|— train_model.py      # Script for preprocessing data and training ML model
|— liver_prediction.pkl # Trained Random Forest model
|— normalizer.pkl      # Saved MinMaxScaler
|— label_encoders.pkl  # Dictionary of saved label encoders
|— Liver.ipynb         # Notebook for model development and training in Colab
```

- app.py handles routing, data preprocessing, and prediction logic.
- train\_model.py processes the dataset, trains the model, and saves necessary artifacts.
- .pkl files are used to load the model and preprocessing tools during prediction.

## 6. Running the Application

- Start the app:

```
cd Flask
```

```
python app.py
```

- Open in browser at: <http://127.0.0.1:5000/>

## 7. API Documentation

This is a form-based app. However, the /predict route accepts POST requests from the form and returns an HTML page with the prediction result. *POST /*

- **Route:** / (root URL)
- **Method:** POST
  - **Inputs:** 41 clinical features submitted through the HTML form, such as:
    - Age
    - Alcohol Type
    - Blood Pressure
    - AST, ALT, Bilirubin • Liver enzyme levels, etc.
  - **Processing:**
    - Input is encoded using label\_encoders.pkl
    - Numerical values are normalized using normalizer.pkl
    - The processed data is passed into the liver\_prediction.pkl model for prediction
  - **Output:**
    - Rendered HTML page (index.html) showing:
      - Prediction result: "**Likely Cirrhosis**" or "**Not Likely Cirrhosis**"
      - Visual indication (color-coded message with icon)
  - **Error Handling:**
    - Displays a friendly error message for missing/invalid inputs ○
      - Handles issues like blood pressure format (e.g., 120/80)

## 8. Authentication

Not applicable – This application does not implement user login or authentication as it's a prototype for medical prediction.

## 9. User Interface

- Form with 40+ inputs related to demographics, lifestyle, blood work, and liver panel.
- Background with a semi-transparent liver theme.
- Health vs Warning output with icons and color-coded messages.

## 10. Testing

- Model accuracy was validated using scikit-learn.
- Manual testing of form inputs and blood pressure parsing.

- The terminal shows Flask running, and incoming form data logged.

```

PC C:\Users\saia\OneDrive\Desktop\liver_prediction_cirrhosis> python train_model.py
█ Deletion values in target after cleaning: [1 'nan' 0]
█ Rows with valid target (0/1): 896
█ Rows after dropping rows with % NaNs in numeric fields: 742
█ Forced alcohol classes: ['both' 'branded liquor' 'country liquor' 'not applicable']
█ Train target distribution:
Predicted Value(Out Come-Patient suffering from liver cirrhosis or not)
0 16
1 577
Name: count, dtype: int64
█ Test target distribution:
Predicted Value(Out Come-Patient suffering from liver cirrhosis or not)
0 145
1 4
Name: count, dtype: int64
█ Model Accuracy: 100.00%
█ Model, normalizer, and encoders saved.

█ Confusion Matrix:
[[ 4  0]
 [ 0 145]]

█ Classification Report:
      precision    recall  f1-score   support

0         1.00      1.00      1.00         4
1         1.00      1.00      1.00       145

 accuracy         1.00      1.00      1.00       149
 macro avg         1.00      1.00      1.00       149
weighted avg         1.00      1.00      1.00       149

Cross-validated accuracy scores: [1. 1. 1. 1.]
Cross-validated accuracy scores: [1. 1. 1. 1.]
Mean accuracy: 1.0

```

Figure 11.4: Backend Terminal Output

- **Demo Link:**  
[https://drive.google.com/file/d/1QhDMID8jW6paeBegi\\_yCfrNAg4wDDyXC/view?usp=drive\\_link](https://drive.google.com/file/d/1QhDMID8jW6paeBegi_yCfrNAg4wDDyXC/view?usp=drive_link)

## 12. Known Issues

- Strict blood pressure format required (e.g., 120/80)
- No database or session tracking
- Limited interpretability of ML model predictions

## 13. Future Enhancements

- Add database to store patient inputs and predictions.
- Provide downloadable PDF reports of results.
- Improve model accuracy with a larger and more balanced dataset.
- Integrate with hospital systems for real-time screening.
- Add user registration, patient history tracking, and mobile responsiveness.