Project Report Format

1. **INTRODUCTION**
   1.1 **Project Overview**

   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.

   1.2 **Purpose**

   The purpose of this project is to develop a machine learning–based predictive system that assists in the early detection of liver cirrhosis using a comprehensive set of clinical and lifestyle-related health indicators. Liver cirrhosis is often diagnosed at a late stage due to the lack of noticeable symptoms in the early phases. Delayed detection leads to limited treatment options, higher healthcare costs, and increased emotional distress for patients and their families.

   This project aims to:

   - Analyze patient health data to identify risk patterns indicative of earlystage liver damage.
   - Design a user-friendly prediction tool that clinicians and patients can use for proactive screening.
   - Empower healthcare providers with reliable, non-invasive decision support for early intervention.
   - Raise awareness about the importance of preventive liver care and encourage at-risk individuals to undergo timely checkups.

   By combining medical insights with modern machine learning techniques, the project seeks to bridge the gap between clinical diagnosis and preventive care, ultimately improving patient outcomes and reducing the burden of liver disease.

2. **IDEATION PHASE**
   2.1 Problem Statement
   2.2 Empathy Map Canvas
   2.3 Brainstorming
3. **REQUIREMENT ANALYSIS**
   3.1 Customer Journey map
   3.2 Solution Requirement
   3.3 Data Flow Diagram
   3.4 Technology Stack
4. **PROJECT DESIGN**
   4.1 Problem Solution Fit
   4.2 Proposed Solution

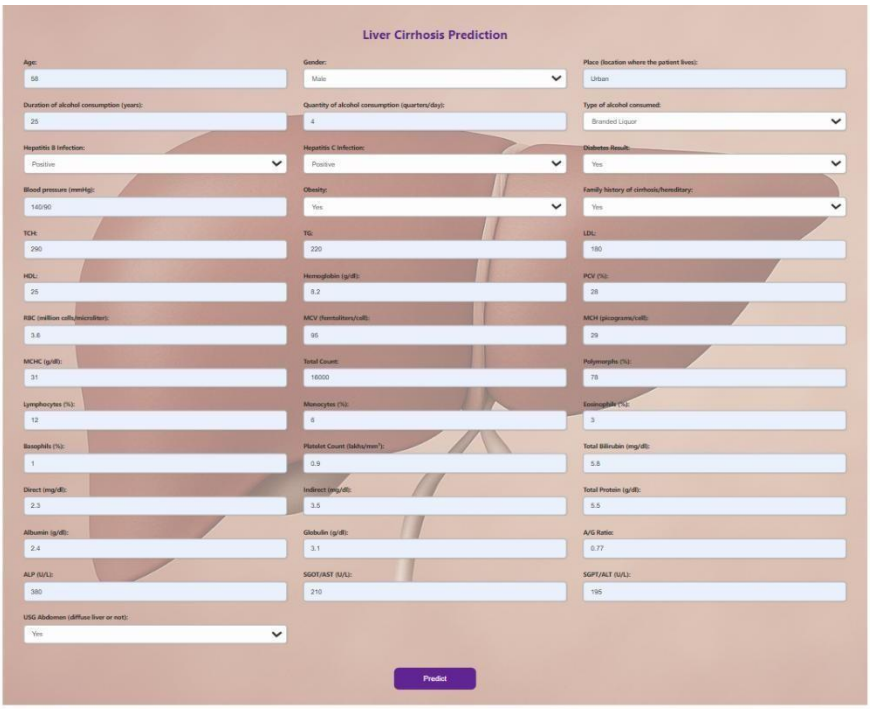

Figure 7.1: Form filled with patient data

Figure 7.2 : Prediction Result-Likely Cirrhosis

Figure 7.3: Prediction Result - Not Likely Cirrhosis

```
# Save the cleaned and processed DataFrame to a CSV file
df.to_csv('cleaned_data.csv', index=False)
df.head()
```

	Age	Gender	Place(location where the patient lives)	Duration of alcohol consumption(years)	Quantity of alcohol consumption (quarters/day)	Type of alcohol consumed	Diabetes Result	Blood pressure (mmhg)	Obesity	Family history of cirrhosis/ hereditary
0	55.0	1	1	12.0	2.0	2	1	32	1	1
1	55.0	1	1	12.0	2.0	2	1	32	1	1
2	55.0	1	1	12.0	2.0	2	1	32	0	1
3	55.0	1	1	12.0	2.0	2	0	32	0	1
4	55.0	0	1	12.0	2.0	2	1	32	0	1

Figure 7.4

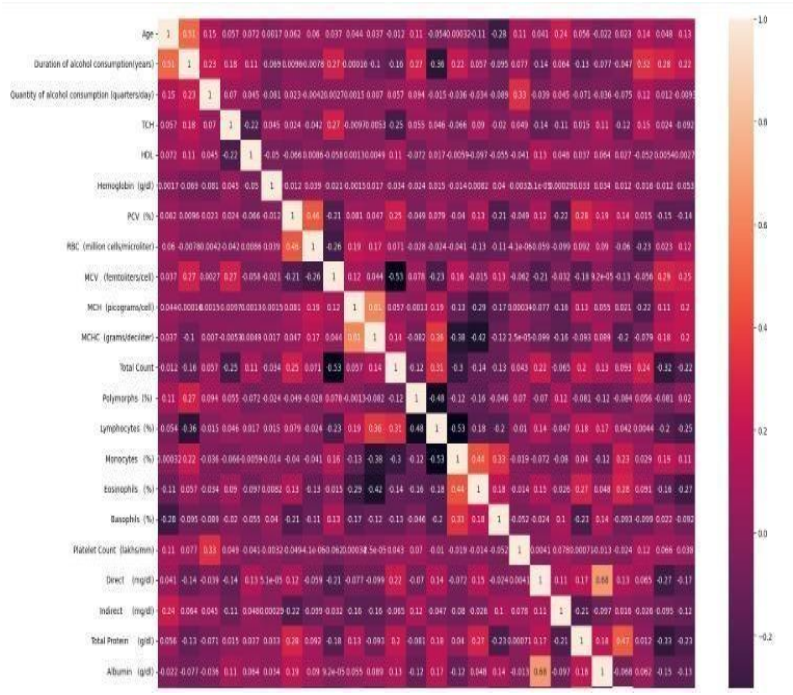

Figure 7.5

```
sns.histplot(df['Age'])
np.percentile(df['Age'],[25,75])
```

```
array([44., 57.])
```

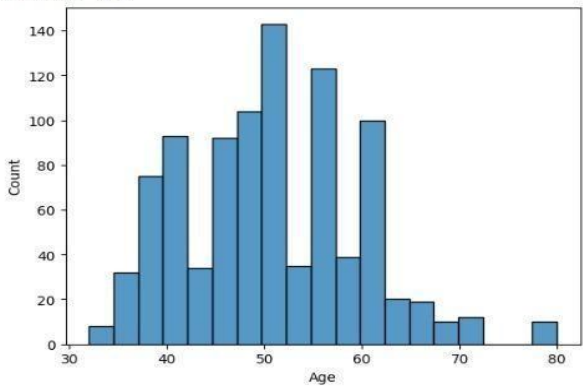

Figure 7.6

8. ADVANTAGES & DISADVANTAGES

8.1 Advantages

- **Early Detection of Liver Cirrhosis**
  The system enables identification of liver cirrhosis in its early stages, which is crucial for timely intervention and improved treatment outcomes.

- **Non-Invasive Assessment**
  The prediction model uses existing clinical and health-related parameters, eliminating the need for painful or invasive diagnostic procedures like liver biopsies.
- **Personalized Risk Evaluation**
  The machine learning model provides risk predictions tailored to individual patient profiles, making the outcomes more relevant and actionable.
- **Time and Cost Efficiency**
  The tool can generate results rapidly and reduce the financial burden associated with repeated diagnostics and hospital visits.

8.2 Disadvantages

- **Dependence on Data Quality**
  The accuracy of the prediction model is highly dependent on the quality and consistency of the input data. Poor data may lead to inaccurate predictions.
- **Potential for Misclassification**
  False positives may cause unnecessary concern, while false negatives could result in missed opportunities for early treatment.
- **Lack of Model Interpretability**
  Some machine learning algorithms may produce results that are difficult for users to interpret, which could reduce trust in the system.
- **Data Privacy and Ethical Concerns**
  Handling sensitive patient data requires strict adherence to privacy policies and ethical guidelines, including compliance with healthcare regulations.

9. CONCLUSION

The project *"Revolutionizing Liver Care: Predicting Liver Cirrhosis using Advanced Machine Learning Techniques"* demonstrates the transformative potential of artificial intelligence in healthcare, specifically in the early prediction and prevention of liver cirrhosis. By utilizing a rich set of clinical and lifestyle-related features, the machine learning model developed in this project aims to detect cirrhosis risk before symptoms become critical—offering a timely and non-invasive approach to liver health management.

In conclusion, this work bridges the gap between medical data and actionable outcomes by combining machine learning with a user-centric interface. It contributes to the vision of smarter, preventive healthcare systems and reinforces the role of technology in enabling better, faster, and more accessible liver disease diagnosis.

10. FUTURE SCOPE

While the current project successfully demonstrates the feasibility and effectiveness of using machine learning for early prediction of liver cirrhosis, there are several areas where the system can be further expanded and improved to maximize clinical impact and real-world usability:

1. **Integration with Electronic Health Records (EHRs)**
   The model can be integrated with hospital EHR systems to automate predictions during routine check-ups, allowing real-time risk assessments without manual data input.
2. **Incorporation of Real-Time Monitoring**
   Future versions can include live health tracking data (e.g., wearable devices, smart sensors) for more dynamic and personalized predictions.
3. **Explainable AI (XAI)**
   To increase clinical trust and transparency, integrating explainable AI techniques would help healthcare professionals understand how the model makes predictions.
4. **Mobile Application Deployment**
   Building a mobile app version of the tool will improve accessibility for patients, allowing them to self-screen and track their liver health regularly.
5. **Support for Multi-Language and Regional Adaptation**
   Adapting the interface and instructions to multiple languages and cultural contexts can improve usability in rural and diverse population settings.

11. APPENDIX

Source Code

This appendix contains key parts of the source code used in the project "Revolutionizing Liver Care: Predicting Liver Cirrhosis using Advanced Machine Learning Techniques". It includes the backend logic, model training script, and frontend form design.

A. app.py – Flask Web Application

This script handles the core backend logic including:

- Route definitions and handling form input
- Loading the trained model, scaler, and label encoders
- Handling form input
- Preprocessing and prediction

```python
from flask import Flask, request, render_template
import pickle
import numpy as np
# import pandas as pd
app = Flask(__name__)
# Load model, normalizer, and encoders
model = pickle.load(open('liver_prediction.pkl', 'rb'))
normalizer = pickle.load(open('normalizer.pkl', 'rb'))
label_encoders = pickle.load(open('label_encoders.pkl', 'rb'))
alcohol_encoder = label_encoders['Type of alcohol consumed']
print("🔍 Classes for 'Type of alcohol consumed':", alcohol_encoder.classes_)
print("✅ Model loaded:", model)
# Exact 41 field names as per your dataset
fields = [
    'Age','Gender','Place(location where the patient lives)','Duration of alcohol consumption(years)','Quantity of alcohol consumption (quarters/day)','Type of alcohol consumed',
    'Hepatitis B infection','Hepatitis C infection','Diabetes Result','Blood pressure (mmhg)','Obesity','Family history of cirrhosis/ hereditary',
    'ICH','TG','LDL','HDL','Hemoglobin  (g/dl)','PCV  (%)','RBC  (million cells/microliter)','MCV  (femtoliters/cell)','MCH  (picograms/cell)',
    'MCHC  (grams/deciliter)','Total Count','Polymorphs  (%)','Lymphocytes  (%)','Monocytes  (%)','Eosinophils  (%)','Basophils  (%)',
    'Platelet Count  (lakhs/mm)','Total Billirubin    (mg/dl)','Direct    (mg/dl)','Indirect     (mg/dl)','Total Protein     (g/dl)','Albumin   (g/dl)',
    'Globulin  (g/dl)','A/G Ratio','AL.Phosphatase      (U/L)','SGOT/AST      (U/L)','SGPT/ALT (U/L)','USG Abdomen (diffuse liver or  not)'
]
@app.route('/')
def home():
    form_fields = []
    for field in fields:
        if field in label_encoders:
            options = label_encoders[field].classes_
            select_html = f'<select name="{field}">' + ''.join([
                f'<option value="{opt}">{opt}</option>' for opt in options
            ]) + '</select>'
            form_fields.append({'label': field, 'input': select_html})
        else:
            input_html = f'<input type="text" name="{field}"'
            if field == 'Blood pressure (mmhg)':
                input_html += ' placeholder="e.g. 120/80" pattern="\\d{2,3}/\\d{2,3}" title="Enter like 120/80"'
            input_html += '>'
            form_fields.append({'label': field, 'input': input_html})
    return render_template('index.html', form_fields=form_fields)

@app.route('/predict', methods=['POST'])
def predict():
    input_data = []
    try:
        for field in fields:
            value = request.form.get(field, '').strip()

            if field == 'Blood pressure (mmhg)':
                # Expect value like "120/80" and convert to 120.80 float
                if '/' not in value:
                    return "❌ Please enter blood pressure in the format 120/80"
                try:
                    systolic, diastolic = map(int, value.split('/'))
                    combined_bp = float(f"{systolic}.{diastolic}")  # e.g. 120.80
                    input_data.append(combined_bp)
                except:
                    return "❌ Invalid blood pressure value. Use format like 120/80"
                continue


            if field in label_encoders:
                encoder = label_encoders[field]
                encoder_classes = [cls.lower().strip() for cls in encoder.classes_]

                if value.lower() not in encoder_classes:
                    return f"❌ Invalid categorical input for '{field}': '{value}'. Expected: {list(encoder.classes_)}"

                matched_value = encoder.classes_[encoder_classes.index(value.lower())]
                encoded = encoder.transform([matched_value])[0]
                input_data.append(encoded)
            else:
                try:
                    input_data.append(float(value))
                except ValueError:
                    return f"❌ Invalid numeric input for '{field}': '{value}'"
        if len(input_data) != len(fields):
            return f"❌ Expected {len(fields)} inputs, got {len(input_data)}."

        input_array = np.array([input_data])
        normalized = normalizer.transform(input_array)
        prediction = model.predict(normalized)[0]
        print("📝 Encoded input:", input_data)
        print("🔮 Prediction output:", prediction)

        result_text = "🟢 Result: Not Likely Cirrhosis"
        result_class = ""
        if str(prediction).strip().lower() in ['1', 'yes']:
            result_text = "🔴 Result: Likely Cirrhosis"
            result_class = "danger"

        return render_template('index.html', form_fields=[
            {'label': field, 'input': f'<input type="text" name="{field}" value="{request.form.get(field,"")}>' if field not in label_encoders else ''}
            for field in fields
        ], prediction_text=result_text, result_class=result_class)

    except Exception as e:
        return f"⚠️ Error: {str(e)}"
if __name__ == '__main__':
    app.run(debug=True)
```

B. train_model.py – Model Training Scrip: This script handles dataset cleaning, label encoding, feature normalization, model training (using Random Forest), and saving the model and tools.

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pickle

# Load the dataset
df = pd.read_excel("HealthCareData.xlsx")
df.columns = df.columns.str.strip()

# Define target
target = 'Predicted Value(Out Come-Patient suffering from liver  cirrosis or not)'
df[target] = df[target].astype(str).str.strip().str.lower()
df[target] = df[target].replace({'yes': 1, 'no': 0})
print("🔍 Unique values in target after cleaning:", df[target].unique())

# Filter out only rows with target 0 or 1
df = df[df[target].isin([0, 1])]

print("📊 Rows with valid target (0/1):", len(df))

# Numeric columns that may contain dirty data
numeric_columns = [
    'TCH', 'TG', 'LDL', 'HDL',
    'Hemoglobin  (g/dl)', 'PCV  (%)', 'RBC  (million cells/microliter)',
    'MCV  (femtoliters/cell)', 'MCH  (picograms/cell)', 'MCHC  (grams/deciliter)',
    'Total Count', 'Polymorphs  (%)', 'Lymphocytes  (%)', 'Monocytes  (%)',
    'Eosinophils  (%)', 'Basophils  (%)', 'Platelet Count  (lakhs/mm)',
    'Total Bilirubin  (mg/dl)', 'Direct  (mg/dl)', 'Indirect  (mg/dl)',
    'Total Protein  (g/dl)', 'Albumin  (g/dl)', 'Globulin  (g/dl)',
    'A/G Ratio', 'AL.Phosphatase  (U/L)', 'SGOT/AST  (U/L)', 'SGPT/ALT (U/L)'
]

# Convert dirty numerics to float, coercing errors to NaN
df[numeric_columns] = df[numeric_columns].apply(pd.to_numeric, errors='coerce')

# Remove rows with too many NaNs in these columns
df = df[df[numeric_columns].isnull().sum(axis=1) <= 5]
print("🧹 Rows after dropping rows with >5 NaNs in numeric fields:", df.shape[0])

# Fill remaining NaNs with column means
df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].mean())

# Define features (drop S.NO and target)
features = [col for col in df.columns if col not in ['S.NO', target]]
X = df[features].copy()
y = df[target].astype(int)

# Label encode categorical columns
label_encoders = {}

for col in X.columns:
    if X[col].dtype == 'object':
        X[col] = X[col].astype(str).str.strip().str.lower()
        le = LabelEncoder()

        if col == 'Type of alcohol consumed':
            custom_classes = ['country liquor', 'branded liquor', 'both', 'not applicable']
            le.fit(custom_classes)
            print(f"🔧 Forced alcohol classes: {le.classes_}")
            X[col] = X[col].apply(lambda x: x if x in custom_classes else 'not applicable')
        else:
            le.fit(X[col].unique())

        X[col] = le.transform(X[col])
        label_encoders[col] = le

# Normalize
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42,stratify=y)


# Train the model
model = RandomForestClassifier(class_weight='balanced', random_state=42)
model.fit(X_train, y_train)

# Evaluate
y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print(f"✅ Model Accuracy: {acc * 100:.2f}%")

# Save model and tools
pickle.dump(model, open("liver_prediction.pkl", "wb"))
pickle.dump(scaler, open("normalizer.pkl", "wb"))
pickle.dump(label_encoders, open("label_encoders.pkl", "wb"))
print("📦 Model, normalizer, and encoders saved.")
```

Dataset Link: https://www.kaggle.com/datasets/bhavanipriya222/liver-cirrhosis-prediction

Project Demo Link:
https://drive.google.com/file/d/1QhDMlD8jW6paeBegi_yCfrNAg4wDDyXC/view?usp=drive_link