Final Project Report

Prediction and Analysis of Liver Patients Data Using Machine Learning

1. Introduction

1.1 Project overviews

Prediction and Analysis of Liver Patients Data Using Machine Learning

This project aims to use machine learning techniques to predict liver disease in patients based on clinical data. The dataset includes parameters such as age, gender, and various liver function tests. The goal is to develop predictive models that can identify patterns and risk factors associated with liver disease. By using algorithms like decision trees, support vector machines, or neural networks, the project seeks to provide accurate predictions to assist healthcare professionals in early diagnosis and treatment planning. The analysis will also offer insights into which features are most influential in liver disease prognosis.

1.2 Objectives

1. Develop robust machine learning models to predict liver disease based on clinical data.

2. Analyze patient data to identify key risk factors and patterns associated with liver disease.

3. Improve diagnostic accuracy by leveraging data-driven techniques over conventional methods.

4. Facilitate early detection of liver disorders, enabling timely and effective treatment.

5. Provide healthcare professionals with a decision-support tool for assessing liver disease risk.

6. Optimize patient management by prioritizing high-risk individuals for further medical evaluation.

7. Enhance understanding of liver disease progression through data insights and model interpretation.

8. Compare and evaluate the performance of different machine learning algorithms.

9. Implement feature selection techniques to identify the most influential predictors of liver disease.

10 .Improve overall healthcare efficiency by reducing misdiagnoses and minimizing invasive diagnostic tests.

2. Project Initialization and Planning Phase

2.1 Define Problem Statements (Patients Problem Statement Template):

Liver diseases are a global health concern, and early detection is critical. The goal of this project is to build a machine learning model that predicts liver diseases based on demographic, clinical, and biochemical data, as well as imaging results. The project involves cleaning and exploring the data, developing predictive models (using various algorithms), and identifying key factors influencing liver disease. The outcomes aim to help doctors detect liver disease risks early and improve treatment strategies, while also addressing challenges related to data quality and privacy concerns

Problem Statement (PS)	**I am (Customer)**	**I'm trying to**	**But**	**Because**	**Which makes me feel**
PS-1	A Person	Diagnosis liver problem	Symptoms can be vague or mistaken for other conditions, leading to late diagnosis.	No specific symptom,gradualonset, overlap with other conditions, Misdiagnosis	Progression of Disease, Delayed Treatment, Finsncial Burden, increased Anxiety and Stress.

I am	I'm trying to	But	Because	Which makes me feel
•A person	•Diagnosis liver problem.	•Symptoms can be vague or mistaken for other conditions, •leading to late diagnosis.	•No specific symptom,gradual onset,overlap with other conditions, Misdiagnosis	•Progression of Disease, Delayed Treatment, Finsncial Burden, increased Anxiety and Stress..

2.2 Project Proposal (Proposed Solution) template

The proposal report aims to transform prediction and analysis of liver disease in patients using machine learning, boosting efficiency and accuracy. It tackles system inefficiencies, promising better operations, reduced risks, and happier customers. Key features include a machine learning-based credit model and real-time decision-making.

Project Overview	
Objective	Itevaluate the overall quality of the dataset by addressing key aspects such as missing data, outliers, and anomalies, and ensure the data meets the requirements for building a machine learning model for liver disease prediction...
Scope	The scope of the liver disease prediction project includes collecting a diverse and comprehensive dataset, conducting thorough exploratory analysis, and developing machine learning models to predict liver disease outcomes. The project will provide valuable insights into the key factors influencing liver health, support early diagnosis, and help personalize treatment plans, ultimately improving patient care. Challenges related to data quality, privacy, and model accuracy will be managed throughout the process to ensure reliable outcomes.
Problem Statement	
Description	The prediction and analysis of liver patient data using machine learning involves utilizing clinical features and biochemical markers to develop predictive models that identify the presence of liver disease. By applying various algorithms, the approach aims to enhance early diagnosis and improve treatment strategies based on data-driven insights. This process includes data preprocessing, feature selection, model evaluation, and interpretation to inform healthcare professionals effectively..
Impact	The ability to make accurate predictions, identify risk factors, and improve early detection can lead to better patient outcomes, lower healthcare costs, and more effective public health strategies. It also empowers healthcare providers with data-driven tools, enhancing decision-making and patient care.

Proposed Solution	
Approach	This approach involves predicting liver disease using machine learning through data collection, preprocessing, and exploratory analysis to handle missing data, outliers, and select key features. Various models like Logistic Regression, Random Forest, and SVM are trained and evaluated using accuracy, precision, recall, F1-score, and ROC-AUC. The best model is used for predictions, with post-prediction analysis to identify important features using SHAP or LIME. Visualizations and dashboards assist in interpretation, while the model is deployed and monitored for real-time predictions in healthcare. Continuous refinement ensures improved diagnostic accuracy and risk factor identification.
Key Features	Key features for predicting liver disease include **biochemical markers** such as bilirubin levels, alanine transaminase (ALT), aspartate transaminase (AST), alkaline phosphatase (ALP), albumin, and total proteins, along with **demographic information** like age and gender. Additional features may include the **ALT/AST ratio** and other relevant clinical indicators that contribute to assessing liver health. Analyzing these features helps in identifying patterns and risk factors associated with liver disease.

Resource Requirements

Resource Type	**Description**	**Specification/Allocation**
Hardware		
Computing Resources	CPU/GPU specifications, number of cores	T4 GPU
Memory	RAM specifications	8 GB
Storage	Disk space for data, models, and logs	1 TB SSD
Software		

Frameworks	Python frameworks	Flask
Libraries	Additional libraries	scikit-learn, pandas, numpy, matplotlib, seaborn
Development Environment	IDE, version control	Google colab Notebook, vscode, Git
Data		
Data	Source, size, format	Kaggle dataset, 614, csv UCI dataset, 690csv,Meteorological departments, open weather datasets

2.3 Initial Project Planning Template

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	Data Collection and Preprocessing	USN-1	Understanding and loading data	2	High	prasanthi	2024/09/23	2024/09/26
Sprint-1	Data Collection and Preprocessing	USN-2	Data cleaning	1	High	prasanthi	2024/09/23	2024/09/26

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	Data Collection and Preprocessing	USN-3	EDA	2	Low	mahathi	2024/09/23	2024/09/26
Sprint-2	Model Development	USN-4	Training the model	2	Medium	mahathi	2024/09/27	2024/09/30
Sprint-2	Model tuning and testing	USN-5	Evaluating the model	1	High	mahathi	2024/09/27	2024/09/30
Sprint-2	Model tuning and testing	USN-6	Model tuning	2	High	arif	2024/09/27	2024/09/30
Sprint-2	Model tuning and testing	USN-7	Model testing	1	Medium	arif	2024/09/27	2024/09/30
Sprint-3	Web integration and Deployment	USN-8	Building HTML templates	2	Medium	muzaffar	2024/10/01	2024/10/05
Sprint-3	Web integration and Deployment	USN-9	Local deployment	2	High	prasanthi	2024/10/01	2024/10/05
Sprint-4	Project Report	USN-10	Report	2	Medium	muzaffar	2024/10/06	2024/10/10

3. Data Collection and Preprocessing Phase

3.1 Data Collection Plan & Raw Data Sources Identification Template

Elevate your data strategy with the Data Collection plan and the Raw Data Sources report, ensuring meticulous data curation and integrity for informed decision-making in every analysis and decision-making endeavor.

Data Collection Plan

Section	Description
Project Overview	The project focuses on leveraging machine learning techniques to predict liver disease in patients by analyzing clinical and biochemical data, aiming for early diagnosis and improved treatment strategies. Through data preprocessing, feature selection, and model evaluation, it seeks to identify significant predictors of liver health and enhance decision-making in healthcare. .
Data Collection Plan	The data collection plan involves sourcing liver patient data from medical institutions, public repositories, or health databases, focusing on key features like age, gender, and biochemical markers (bilirubin, ALT, AST). Data will be ethically gathered, anonymized, and stored securely for analysis using machine learning model
Raw Data Sources Identified	Raw data sources for predicting and Analysing liver patients include public datasets like the UCI Liver Disorder Dataset and medical records from hospitals and clinics, which provide comprehensive clinical and biochemical information. Additionally, health databases and research studies offer valuable data for training machine learning models.

Source name	Description	Location/URL	Format	Size	Access Permission
Dataset 1	Smart Internz Platform	https://www.kaggle.com/datasets/uciml/indian-liver-patient-records	CSV	13.5 MB	Public

Raw Data Sources Template

Data Collection and Preprocessing Phase

Data Quality Report Template

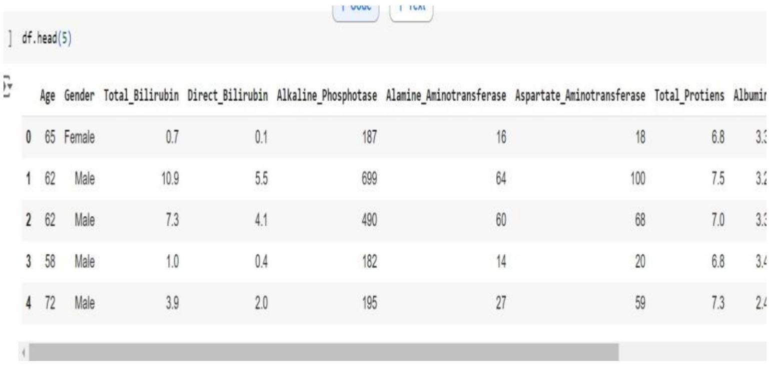
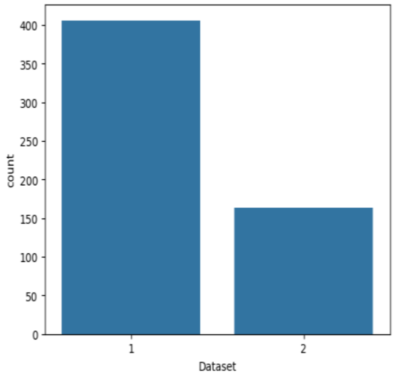
The Data Quality Report will summarize data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

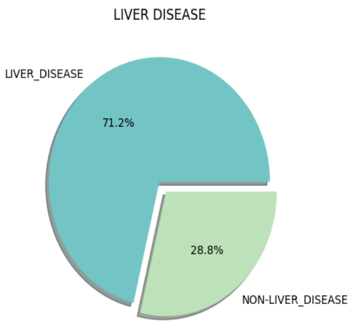
Data Source	Data Quality Issue	Severity	Resolution Plan
Smart Internz Dataset	The dataset is imbalanced, with fewer non-liver disease patients compared to liver disease cases, which can lead to model bias.	Moderate	Balancing the Dataset using Smote.
Smart InternzDataset	Categorical data in the dataset	Moderate	Encoding has to be done in the data

Data Collection and Preprocessing Phase

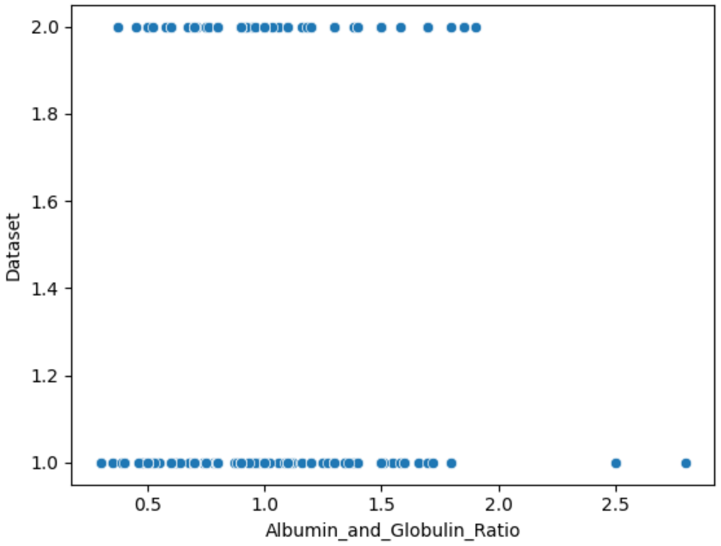
Data Exploration and Preprocessing Template

Dataset variables will be statistically analysed to identify patterns and outliers, with Python employed for preprocessing tasks like normalization and feature engineering. Data cleaning will address missing values and outliers, ensuring quality for subsequent analysis and Modeling, and forming a strong foundation for insights and predictions.

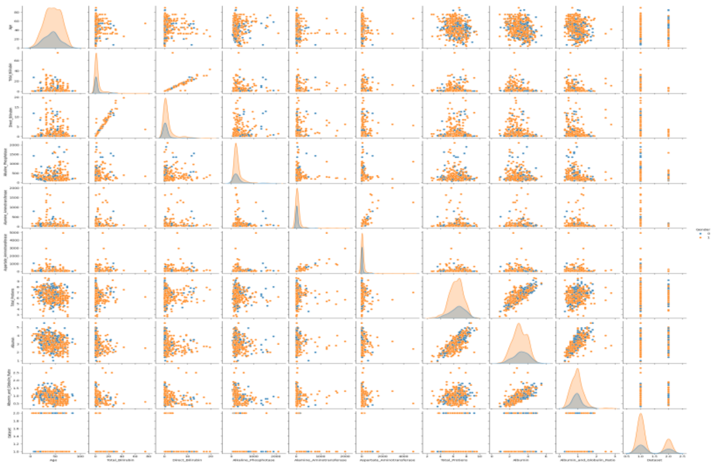
<table>
<tr><th>Section</th><th>Description</th></tr>
<tr><td>Data Overview</td><td>Dimension: 400 rows × 11 columns

] df.head(5)
<table>
<tr><th></th><th>Age</th><th>Gender</th><th>Total_Bilirubin</th><th>Direct_Bilirubin</th><th>Alkaline_Phosphotase</th><th>Alamine_Aminotransferase</th><th>Aspartate_Aminotransferase</th><th>Total_Protiens</th><th>Albumi</th></tr>
<tr><td>0</td><td>65</td><td>Female</td><td>0.7</td><td>0.1</td><td>187</td><td>16</td><td>18</td><td>6.8</td><td>3.3</td></tr>
<tr><td>1</td><td>62</td><td>Male</td><td>10.9</td><td>5.5</td><td>699</td><td>64</td><td>100</td><td>7.5</td><td>3.2</td></tr>
<tr><td>2</td><td>62</td><td>Male</td><td>7.3</td><td>4.1</td><td>490</td><td>60</td><td>68</td><td>7.0</td><td>3.3</td></tr>
<tr><td>3</td><td>58</td><td>Male</td><td>1.0</td><td>0.4</td><td>182</td><td>14</td><td>20</td><td>6.8</td><td>3.4</td></tr>
<tr><td>4</td><td>72</td><td>Male</td><td>3.9</td><td>2.0</td><td>195</td><td>27</td><td>59</td><td>7.3</td><td>2.4</td></tr>
</table>
</td></tr>
<tr><td>Univariate Analysis</td><td>


</td></tr>
</table>

Bivariate Analysis

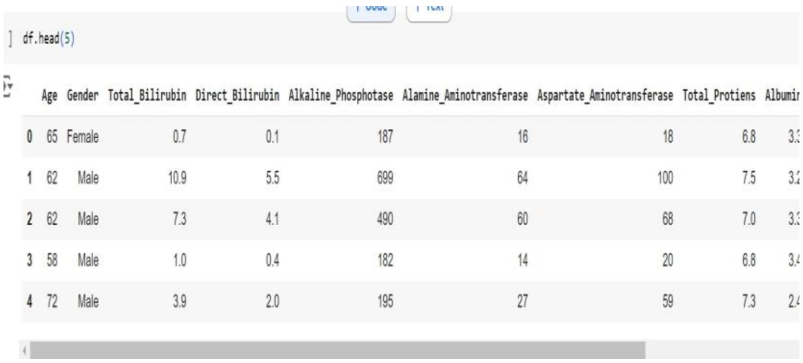


Multivariate Analysis



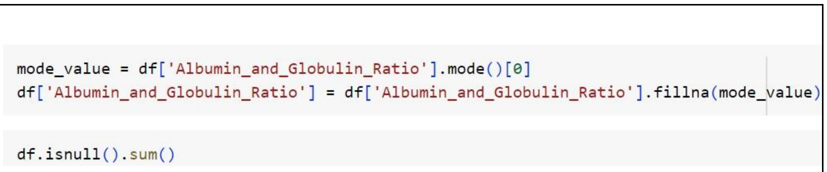
Data Preprocessing Code Screenshots

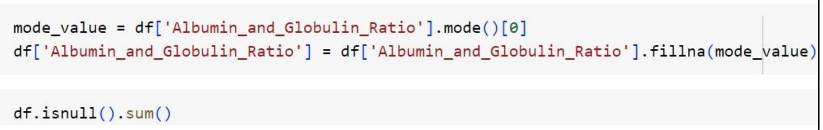
Loading Data

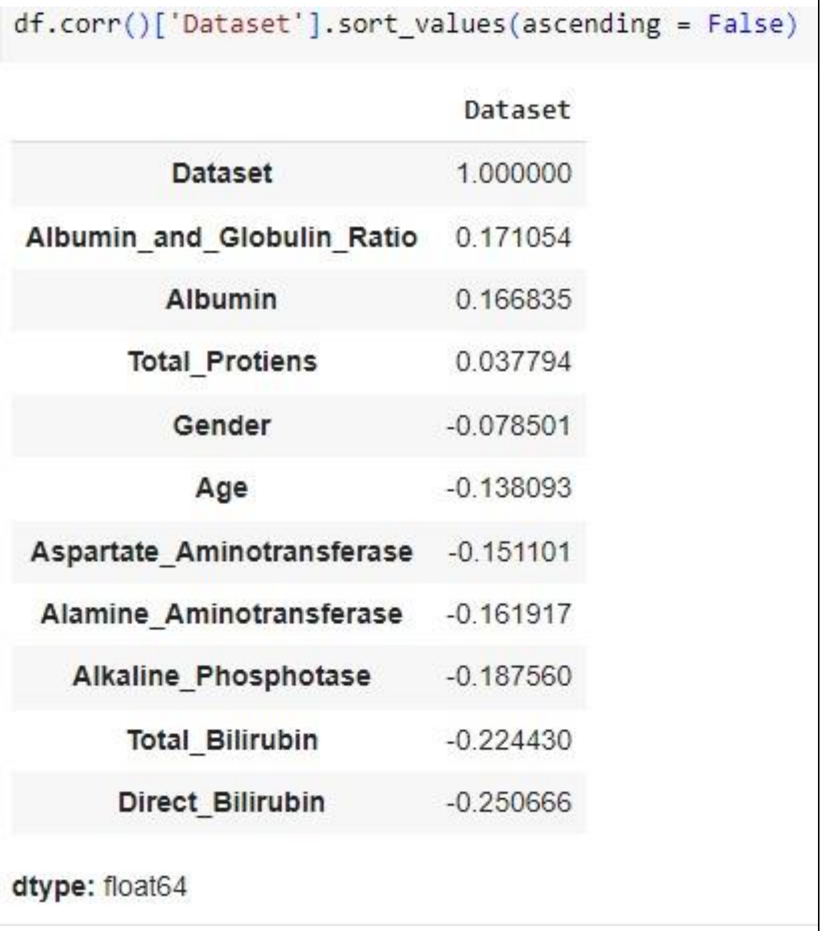


```
df.head(5)
```

```
   Age  Gender  Total_Bilirubin  Direct_Bilirubin  Alkaline_Phosphotase  Alamine_Aminotransferase  Aspartate_Aminotransferase  Total_Protiens  Albumi
0   65  Female              0.7               0.1                   187                        16                          18             6.8     3.3
1   62    Male             10.9               5.5                   699                        64                         100             7.5     3.2
2   62    Male              7.3               4.1                   490                        60                          68             7.0     3.3
3   58    Male              1.0               0.4                   182                        14                          20             6.8     3.4
4   72    Male              3.9               2.0                   195                        27                          59             7.3     2.4
```

Handling Missing Data	
	`mode_value = df['Albumin_and_Globulin_Ratio'].mode()[0]`
	`df['Albumin_and_Globulin_Ratio'] = df['Albumin_and_Globulin_Ratio'].fillna(mode_value)`
	`df.isnull().sum()`

Data Transformation	Encoding
	
	`mode_value = df['Albumin_and_Globulin_Ratio'].mode()[0]`
	`df['Albumin_and_Globulin_Ratio'] = df['Albumin_and_Globulin_Ratio'].fillna(mode_value)`
	`df.isnull().sum()`
	Scaling

Feature Engineering	
	`df.corr()['Dataset'].sort_values(ascending = False)`
	`Dataset`
	`Dataset 1.000000`
	`Albumin_and_Globulin_Ratio 0.171054`
	`Albumin 0.166835`
	`Total_Protiens 0.037794`
	`Gender -0.078501`
	`Age -0.138093`
	`Aspartate_Aminotransferase -0.151101`
	`Alamine_Aminotransferase -0.161917`
	`Alkaline_Phosphotase -0.187560`
	`Total_Bilirubin -0.224430`
	`Direct_Bilirubin -0.250666`
	`dtype: float64`
	Selecting which are highly corelated to the target column and relatable.

Save Processed Data	-

Model Development Phase Template

Feature Selection Report Template

In the forthcoming update, each feature will be accompanied by a brief description. Users will indicate whether it's selected or not, providing reasoning for their decision. This process will streamline decision-making and enhance transparency in feature selection.

Feature	Description	Select (Yes/No)	Reasoning
Age	**Age of the patient in years**	**yes**	**Age is a significant predictor for liver conditions as certain liver diseases are more prevalent in older individuals.**
Gender	**Gender of the patient (Male/Female)**	**yes**	**Gender is often a factor in liver diseases; certain conditions may be more prevalent in men or women**
Total Bilirubin	**Total bilirubin levels in the blood (mg/dL)**	**yes**	**Elevated bilirubin levels are directly linked to liver dysfunction, making this a key feature for liver disease prediction.**

DirectBilirubin	**Direct bilirubin levels in the blood (mg/dL)**	**yes**	**Levels of Bilirubin in blood can indicate that the liver is unable to excrete Bilirubin**
Alkaline Phosphatase	**Alkaline Phosphatase enzyme levels (U/L)**	**yes**	**High levels of ALP in blood may indicate liver disease**
Alamine Aminotransferase (ALT)	**An enzyme that helps the liver convert food into energy**	**Yes**	**Increased levels of alanine aminotransferase (ALT) in the blood can indicate liver damage (hepatitis,live cancer etc)**
Aspartate Aminotransferase (AST)	**An enzyme that helps the liver convert food into energy**	**yes**	**Increased levels of aspartate aminotransferase in blood can cause liver disease, liver cancer or tumors**
Total Proteins	**Measures the total amount of proteins(albumin and globulin) found in your blood**	**yes**	**Low level in total proteins may indicate liver or kidney problem. A high total protein level could indicate dehydration or cancer such as multiple myeloma**
Albumin	**Albumin is a protein produced by liver**	**yes**	**A low albumin level in patients indicates liver diseases such as cirrhosis.**

Albumin and Globulin Ratio	**The albumin-to-globulin ratio is a measure of amount of albumin proteins in blood compared to globulins**	**yes**	**A low The albumin-to-globulin ratio is often found in patients with liver disease such as cirrhosis or hepatits**
Dataset	**Disease outcome**	**yes**	**The target variable for predictive modeling–is essential for the project's goal.**

Model Development Phase Template

Model Selection Report

In the forthcoming Model Selection Report, various models will be outlined, detailing their descriptions, hyperparameters, and performance metrics, including Accuracy or F1 Score. This comprehensive report will provide insights into the chosen models and their effectiveness.

Model Selection Report:

Model	**Description**	**Hyperparameters**	**Performance Metric (e.g., Accuracy, F1 Score)**
RandomForest	Ensemble of decision trees; classifies a new liver object, the input vector is passed through each tree in the forest, and the tree that votes for most likely class wins	-	Accuracy score = 85%

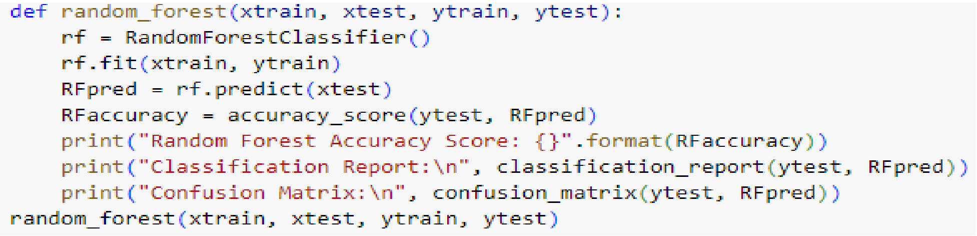
DecisionTree	Simple tree structure can be used to predict liver disease by analyzing data and generating if then rules can be used to determine if the patient has liver disease.	-	Accuracy score = 73%
KNN	Classifies based on nearest neighbors;adapts well to data patterns, effective	**-**	Accuracy score = **71%**
Support Vector machine	SVMs are supervised models that can be used to predict liver disease. SVMs analyze the data and recognize patterns to classify liver disease.	**-**	Accuracy score = **70%**
Logistic Regression	Logistic regression is a statistical method that estimates the probability of an event occurring based on the given dataset.	**-**	Accuracy score = **73%**

Model Development Phase Template

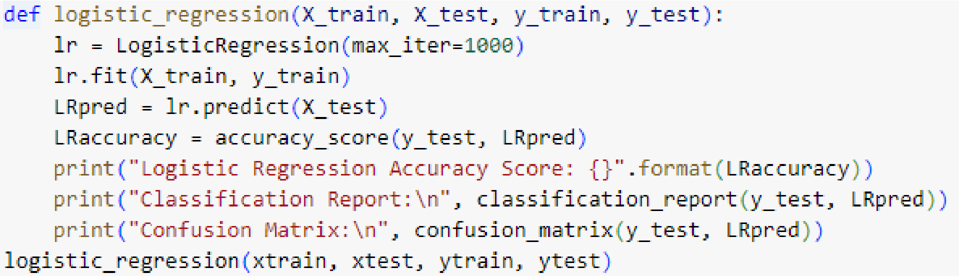
Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots

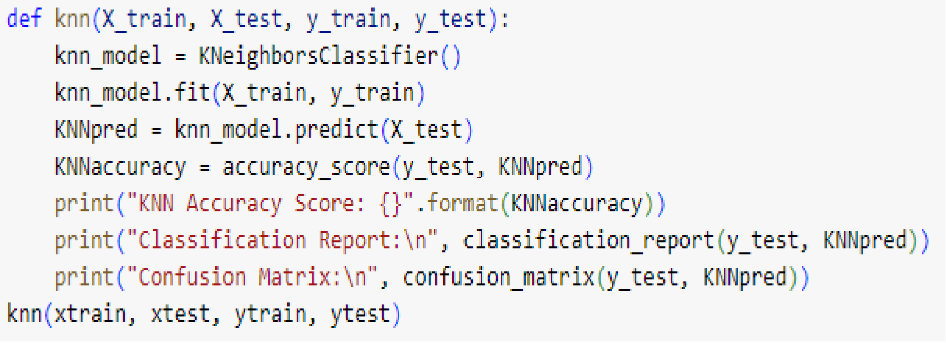
Initial Model Training Code:



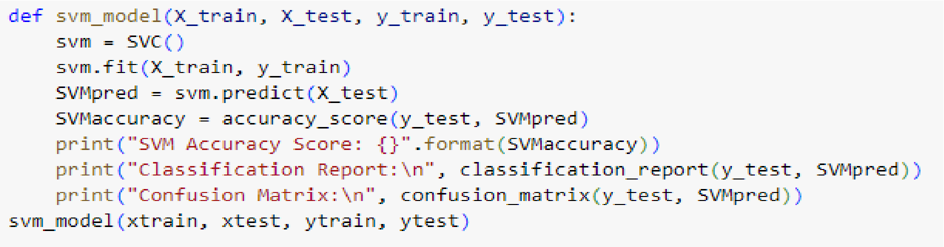
```
def random_forest(xtrain, xtest, ytrain, ytest):
    rf = RandomForestClassifier()
    rf.fit(xtrain, ytrain)
    RFpred = rf.predict(xtest)
    RFaccuracy = accuracy_score(ytest, RFpred)
    print("Random Forest Accuracy Score: {}".format(RFaccuracy))
    print("Classification Report:\n", classification_report(ytest, RFpred))
    print("Confusion Matrix:\n", confusion_matrix(ytest, RFpred))
random_forest(xtrain, xtest, ytrain, ytest)
```



```
def logistic_regression(X_train, X_test, y_train, y_test):
    lr = LogisticRegression(max_iter=1000)
    lr.fit(X_train, y_train)
    LRpred = lr.predict(X_test)
    LRaccuracy = accuracy_score(y_test, LRpred)
    print("Logistic Regression Accuracy Score: {}".format(LRaccuracy))
    print("Classification Report:\n", classification_report(y_test, LRpred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, LRpred))
logistic_regression(xtrain, xtest, ytrain, ytest)
```

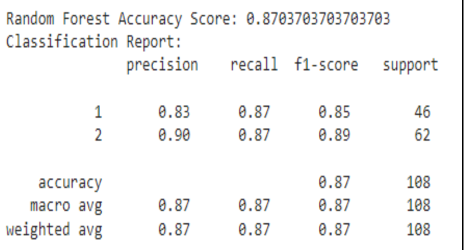
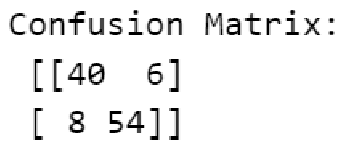
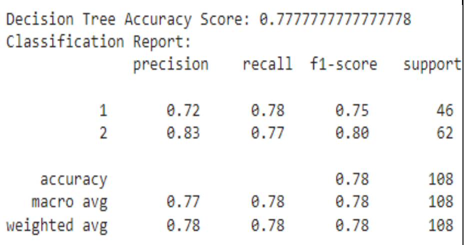
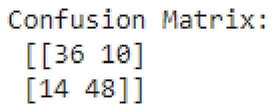
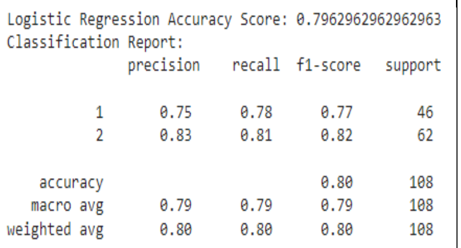
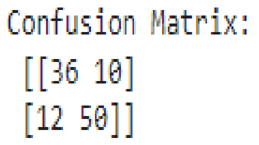
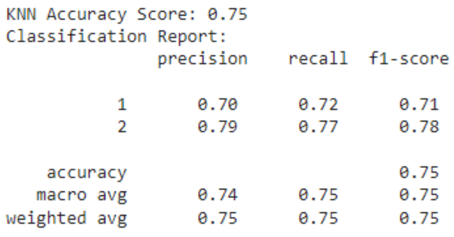
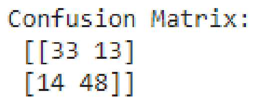
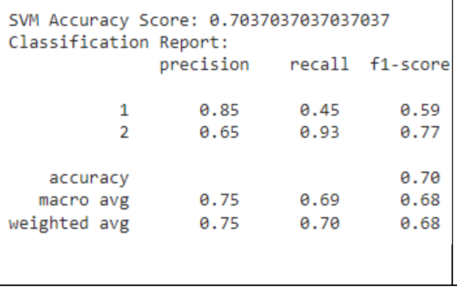
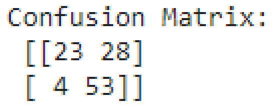


```
def knn(X_train, X_test, y_train, y_test):
    knn_model = KNeighborsClassifier()
    knn_model.fit(X_train, y_train)
    KNNpred = knn_model.predict(X_test)
    KNNaccuracy = accuracy_score(y_test, KNNpred)
    print("KNN Accuracy Score: {}".format(KNNaccuracy))
    print("Classification Report:\n", classification_report(y_test, KNNpred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, KNNpred))
knn(xtrain, xtest, ytrain, ytest)
```



```
def svm_model(X_train, X_test, y_train, y_test):
    svm = SVC()
    svm.fit(X_train, y_train)
    SVMpred = svm.predict(X_test)
    SVMaccuracy = accuracy_score(y_test, SVMpred)
    print("SVM Accuracy Score: {}".format(SVMaccuracy))
    print("Classification Report:\n", classification_report(y_test, SVMpred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, SVMpred))
svm_model(xtrain, xtest, ytrain, ytest)
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix
Random Forest	 Random Forest Accuracy Score: 0.8703703703703703 Classification Report: precision recall f1-score support 1 0.83 0.87 0.85 46 2 0.90 0.87 0.89 62 accuracy 0.87 108 macro avg 0.87 0.87 0.87 108 weighted avg 0.87 0.87 0.87 108	87%	 Confusion Matrix: [[40 6] [8 54]]
Decision Tree	 Decision Tree Accuracy Score: 0.7777777777777778 Classification Report: precision recall f1-score support 1 0.72 0.78 0.75 46 2 0.83 0.77 0.80 62 accuracy 0.78 108 macro avg 0.77 0.78 0.78 108 weighted avg 0.78 0.78 0.78 108	78%	 Confusion Matrix: [[36 10] [14 48]]
Logistic Regression	 Logistic Regression Accuracy Score: 0.7962962962962963 Classification Report: precision recall f1-score support 1 0.75 0.78 0.77 46 2 0.83 0.81 0.82 62 accuracy 0.80 108 macro avg 0.79 0.79 0.79 108 weighted avg 0.80 0.80 0.80 108	80%	 Confusion Matrix: [[36 10] [12 50]]
KNN	 KNN Accuracy Score: 0.75 Classification Report: precision recall f1-score 1 0.70 0.72 0.71 2 0.79 0.77 0.78 accuracy 0.75 macro avg 0.74 0.75 0.75 weighted avg 0.75 0.75 0.75	75%	 Confusion Matrix: [[33 13] [14 48]]
Support Vector machine	 SVM Accuracy Score: 0.7037037037037037 Classification Report: precision recall f1-score 1 0.85 0.45 0.59 2 0.65 0.93 0.77 accuracy 0.70 macro avg 0.75 0.69 0.68 weighted avg 0.75 0.70 0.68	70%	 Confusion Matrix: [[23 28] [4 53]]

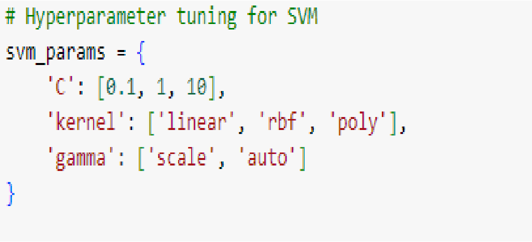
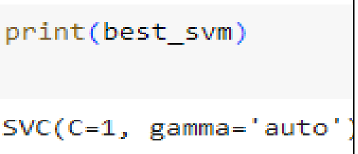
Model Optimization and Tuning Phase Template

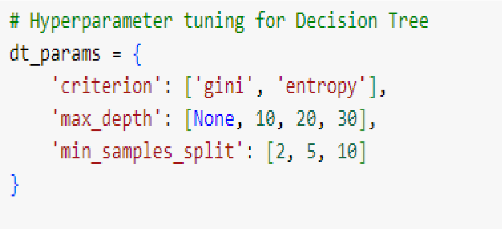
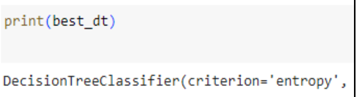
Model Optimization and Tuning Phase

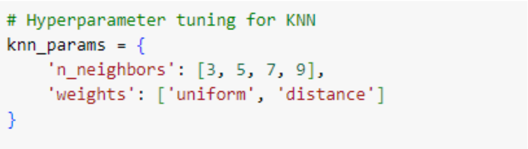
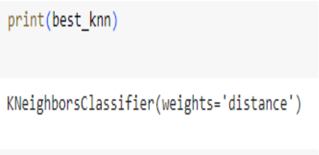
The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (6 Marks):

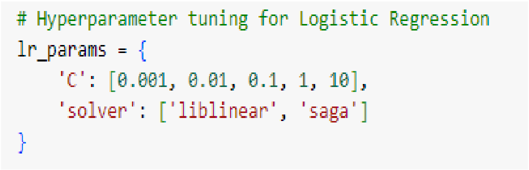
Model	Tuned Hyperparameters	Optimal Values

Support Vector Machine	
	`# Hyperparameter tuning for SVM`
	`svm_params = {`
	`    'C': [0.1, 1, 10],`
	`    'kernel': ['linear', 'rbf', 'poly'],`
	`    'gamma': ['scale', 'auto']`
	`}`
		
		`print(best_svm)`
		`SVC(C=1, gamma='auto')`

Decision Tree	
	`# Hyperparameter tuning for Decision Tree`
	`dt_params = {`
	`    'criterion': ['gini', 'entropy'],`
	`    'max_depth': [None, 10, 20, 30],`
	`    'min_samples_split': [2, 5, 10]`
	`}`
		
		`print(best_dt)`
		`DecisionTreeClassifier(criterion='entropy',`

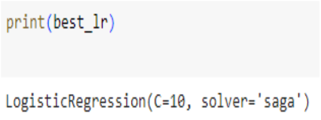
KNN	
	`# Hyperparameter tuning for KNN`
	`knn_params = {`
	`    'n_neighbors': [3, 5, 7, 9],`
	`    'weights': ['uniform', 'distance']`
	`}`
		
		`print(best_knn)`
		`KNeighborsClassifier(weights='distance')`

<table>
<tr>
<td>Logistic
Regression</td>
<td>



```python
# Hyperparameter tuning for Logistic Regression
lr_params = {
    'C': [0.001, 0.01, 0.1, 1, 10],
    'solver': ['liblinear', 'saga']
}
```

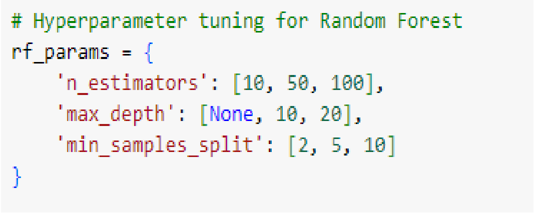
</td>
<td>



```python
print(best_lr)
```

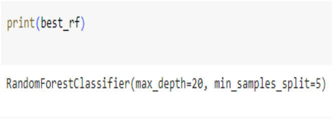
```
LogisticRegression(C=10, solver='saga')
```

</td>
</tr>
<tr>
<td>RandomForest</td>
<td>



```python
# Hyperparameter tuning for Random Forest
rf_params = {
    'n_estimators': [10, 50, 100],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10]
}
```

</td>
<td>



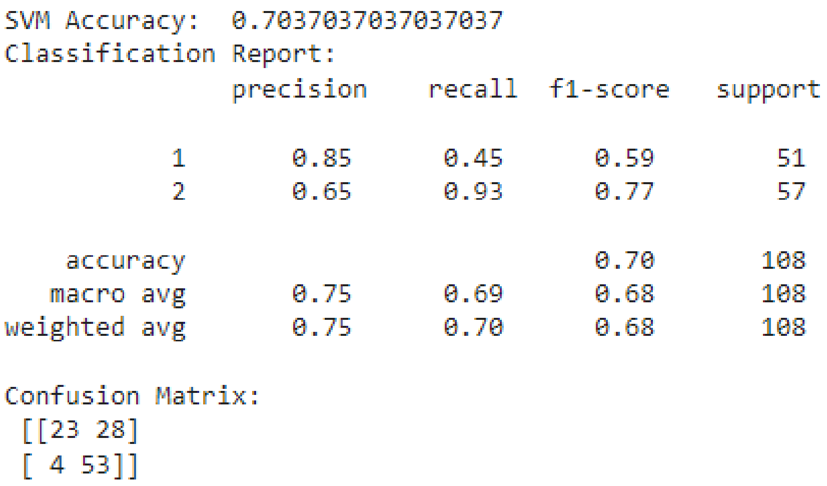
```python
print(best_rf)
```

```
RandomForestClassifier(max_depth=20, min_samples_split=5)
```

</td>
</tr>
</table>

Performance Metrics Comparison Report (2 Marks):

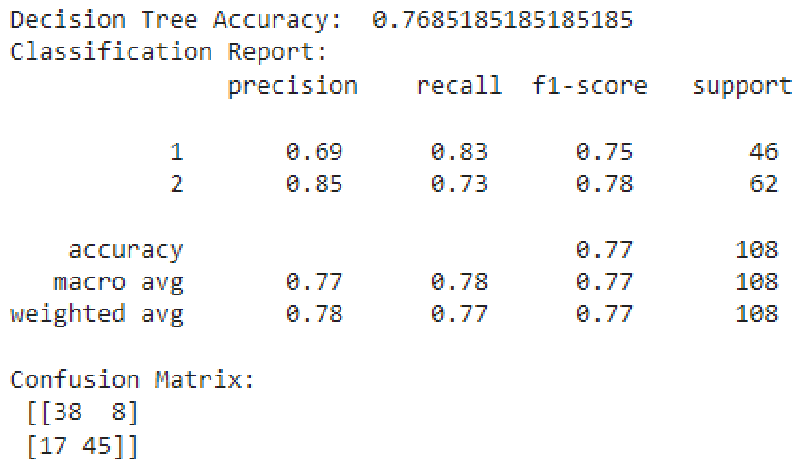
<table>
<tr>
<th>Model</th>
<th>Optimized Metric</th>
</tr>
<tr>
<td>Support Vector
Machine</td>
<td>



```
SVM Accuracy:  0.7037037037037037
Classification Report:
              precision    recall  f1-score   support

           1       0.85      0.45      0.59        51
           2       0.65      0.93      0.77        57

    accuracy                           0.70       108
   macro avg       0.75      0.69      0.68       108
weighted avg       0.75      0.70      0.68       108

Confusion Matrix:
 [[23 28]
 [ 4 53]]
```

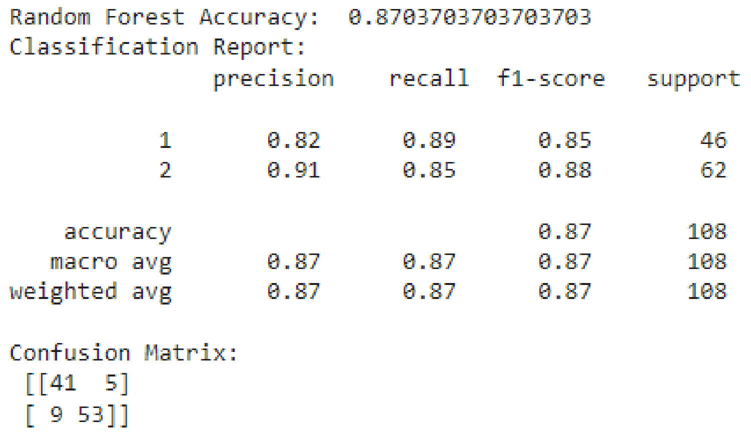
</td>
</tr>
</table>

<table>
<tr>
<td>Decision Tree</td>
<td>



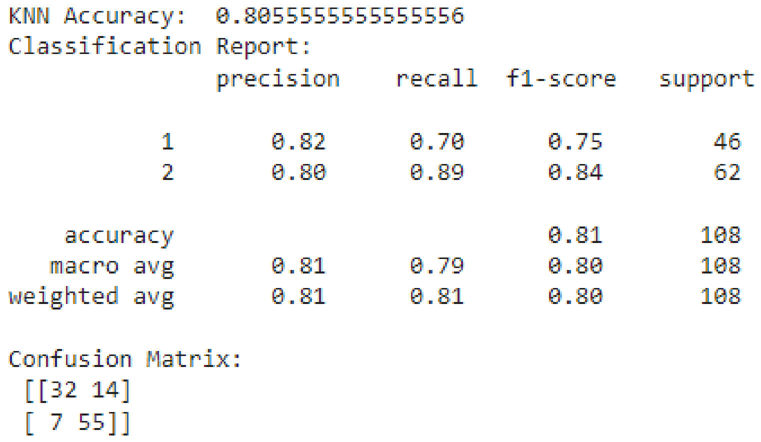
```
Decision Tree Accuracy:  0.7685185185185185
Classification Report:
               precision    recall  f1-score   support

           1       0.69      0.83      0.75        46
           2       0.85      0.73      0.78        62

    accuracy                           0.77       108
   macro avg       0.77      0.78      0.77       108
weighted avg       0.78      0.77      0.77       108

Confusion Matrix:
 [[38  8]
 [17 45]]
```

</td>
</tr>
<tr>
<td>Random Forest</td>
<td>



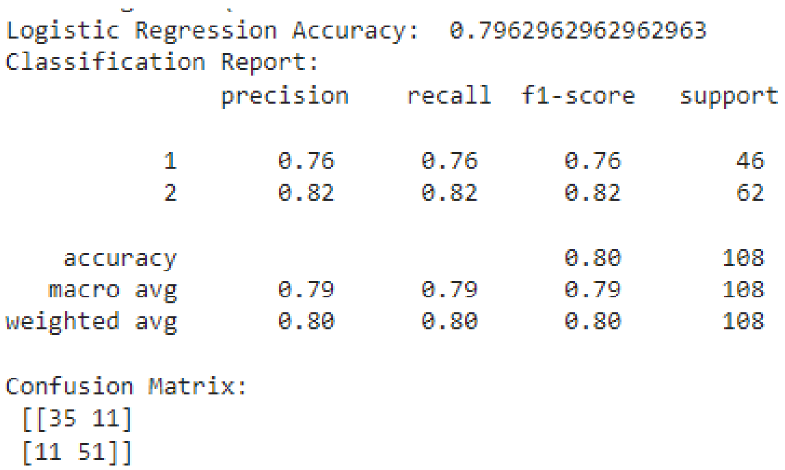
```
Random Forest Accuracy:  0.8703703703703703
Classification Report:
               precision    recall  f1-score   support

           1       0.82      0.89      0.85        46
           2       0.91      0.85      0.88        62

    accuracy                           0.87       108
   macro avg       0.87      0.87      0.87       108
weighted avg       0.87      0.87      0.87       108

Confusion Matrix:
 [[41  5]
 [ 9 53]]
```

</td>
</tr>
<tr>
<td>KNN</td>
<td>



```
KNN Accuracy:  0.8055555555555556
Classification Report:
               precision    recall  f1-score   support

           1       0.82      0.70      0.75        46
           2       0.80      0.89      0.84        62

    accuracy                           0.81       108
   macro avg       0.81      0.79      0.80       108
weighted avg       0.81      0.81      0.80       108

Confusion Matrix:
 [[32 14]
 [ 7 55]]
```

</td>
</tr>
</table>

Logistic Regression	

```
Logistic Regression Accuracy:  0.7962962962962963
Classification Report:
               precision    recall  f1-score   support

           1       0.76      0.76      0.76        46
           2       0.82      0.82      0.82        62

    accuracy                           0.80       108
   macro avg       0.79      0.79      0.79       108
weighted avg       0.80      0.80      0.80       108

Confusion Matrix:
 [[35 11]
 [11 51]]
```

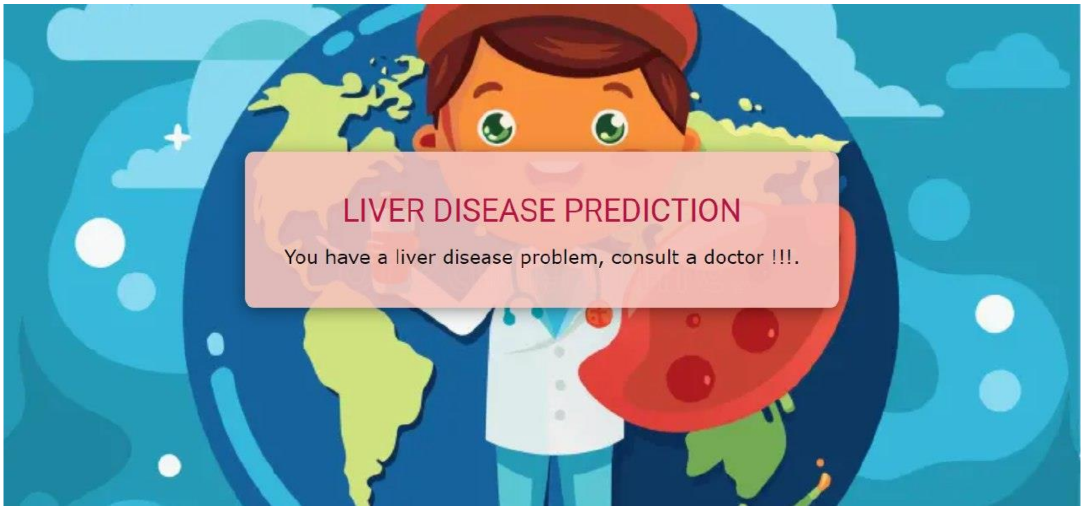
Final Model Selection Justification:

Final Model	Reasoning
Random Forest Classifier	We select the Random Forest classifier for liver disease prediction because it can handle complex, non-linear relationships in medical data and provides high accuracy. It offers inherent feature importance ranking, helping to identify key factors influencing liver disease. Random Forest is also robust to overfitting and effectively manages missing data and outliers, ensuring reliable predictions. Additionally, its ability to handle large datasets and a mix of feature types (continuous and categorical) makes it well-suited for liver patient data analysis.

6. Results

6.1 Outputs screenshots

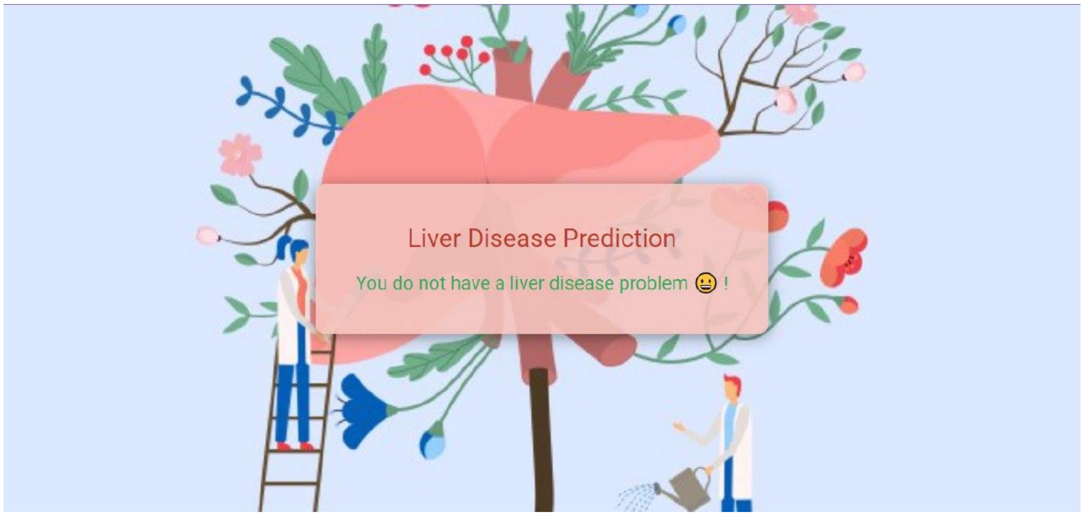
Output for having a liver disease:



LIVER DISEASE PREDICTION

You have a liver disease problem, consult a doctor !!!.

Output for not having a liver disease:



Liver Disease Prediction

You do not have a liver disease problem 😀 !

7. Advantages & Disadvantages

Advantages

1.Improved diagnostic accuracy: Machine learning models can analyze complex patterns in patient data, leading to more accurate predictions of liver disease compared to traditional methods.

2. Early detection: ML models can identify early signs and risk factors for liver disease, enabling timely intervention and treatment, which can improve patient outcomes.

3. Handling large and diverse data: Machine learning efficiently processes large datasets with a variety of clinical features (e.g., blood test results, demographics), making it suitable for analyzing complex liver patient data.

4.Automated analysis: ML can automate the data analysis process, saving time for healthcare professionals by providing quick and reliable predictions based on patient data.

5. Personalized treatment: Machine learning can help tailor treatment plans by identifying patient-specific risk factors, allowing for more personalized and targeted healthcare.

6.Feature importance analysis: ML techniques, such as Random Forest, can highlight the most significant predictors of liver disease, improving the understanding of which factors are most critical for diagnosis and prognosis.

7.Reduction of human error: By relying on data-driven predictions, machine learning reduces the risk of human bias and error in diagnosing **liver diseases**

Disadvantages

1. **Data quality dependency**: Machine learning models heavily rely on the quality and completeness of data. Missing or inaccurate clinical data can negatively affect the model's performance and lead to unreliable predictions.
2. **Complexity in interpretation**: Some advanced machine learning models, like neural networks, are often considered "black boxes," making it difficult for healthcare professionals to understand how specific predictions are made or which factors influenced the decision.

3. **Overfitting risk**: If not properly managed, machine learning models can overfit the training data, leading to poor generalization when applied to new patient data, reducing their reliability in real-world scenarios.
4. **Requires large datasets**: To achieve high accuracy, machine learning models often require large amounts of labeled data, which may not always be available in medical settings, especially for rare liver conditions.
5. **Computationally intensive**: Training complex models on large datasets can be computationally expensive and time-consuming, requiring specialized hardware and expertise that may not be readily available in healthcare facilities.
6. **Ethical and privacy concerns**: Using patient data in machine learning models raises concerns about data privacy and the ethical implications of automated decision-making in healthcare, especially if models are deployed without human oversight.

8. Conclusion

This project demonstrated the effectiveness of using machine learning algorithms to predict and analyze liver disease based on patient data. By leveraging various models, such as Random Forest and Support Vector Machines, we were able to achieve improved diagnostic accuracy and identify key factors influencing liver disease progression. The integration of feature selection methods helped streamline the models, enhancing both performance and interpretability. Although challenges such as the need for large datasets and careful handling of missing data were encountered, the project showcased how machine learning can transform liver disease detection. This not only aids early intervention but also assists healthcare professionals in making data-driven decisions for better patient management. Future work could involve refining models further with larger datasets and exploring more interpretable approaches to ensure broader adoption in clinical practice.patterns.

9. Future Scope

1. Integration of advanced models: Future work could explore the use of deep learning techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), for more sophisticated analysis of liver disease, especially for imaging data or time-series data like patient histories.

2. Incorporation of larger, diverse datasets: Expanding the dataset with more diverse patient populations and multi-center data can improve model generalization and enhance its applicability across different demographics, ensuring more robust and accurate predictions.

3.Development of explainable AI: There is potential to integrate explainable AI techniques that provide clearer insights into the decision-making process of machine learning models, making them more interpretable for healthcare professionals and facilitating their adoption in clinical settings.

4. Real-time prediction systems: Future advancements could involve implementing machine learning models in real-time clinical environments, allowing doctors to use predictive tools during patient visits for immediate liver disease assessment and treatment recommendations.

5. Incorporation of genomic and lifestyle data: Including additional data types, such as genetic information, lifestyle factors, and environmental exposure, can enhance the models' predictive power and lead to a more comprehensive understanding of liver disease risk factors.

6. Integration with healthcare systems: Machine learning models can be integrated into electronic health record (EHR) systems to provide automated liver disease risk assessments, alerting healthcare providers about high-risk patients and improving workflow efficiency.

7. Personalized treatment plans: The models could be further refined to offer personalized treatment suggestions based on individual risk profiles, enabling more tailored medical interventions and improving patient outcomes.

8.Collaboration with medical experts: In the future, closer collaboration with hepatologists and data scientists can enhance model development and ensure that the algorithms align with real-world clinical needs and practices

10. Appendix

10.1 Source Code

Prediction and analysis of liver patients data using machine learning.ipynb

```
#IMPORTING NECESSARY LIBRARIES

import pandas as pd

import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt

import pickle


#LOADING THE DATASET

df = pd.read_csv("/content/indian_liver_patient.csv")


#EXPLORATORY DATA ANALYSIS

df.head(5)

df.tail()

df = df.drop_duplicates()

df.shape

df.value_counts('Dataset')

df.describe()

df.info()


#CHECKING NULL VALUES

df.isnull().any()


#HANDLING NULL VALUES

mode_value = df['Albumin_and_Globulin_Ratio'].mode()[0]

df['Albumin_and_Globulin_Ratio'] = df['Albumin_and_Globulin_Ratio'].fillna(mode_value)

df.isnull().sum()


#DATA VISUALIZATION

#UNIVARIATE

sns.countplot(data= df, x = 'Gender',label = 'Count')
```

```
m,f = df['Gender'].value_counts()
print("No of Males: ",m)
print("No of Females: ",f)

sns.countplot(data= df, x= 'Dataset')
LD,NLD = df['Dataset'].value_counts()
print("liver disease patients: ",LD)
print("Non-liver disease patients: ",NLD)

plt.pie(df['Dataset'].value_counts(),[0,0.1],labels=['LIVER_DISEASE','NON-LIVER_DISEASE'],autopct ="%1.1f%%",shadow = True,colors = ['#73C5C5','#BDE2B9'])
plt.title("LIVER DISEASE")
plt.show()

#BIVARIATE
sns.scatterplot(x = df['Albumin_and_Globulin_Ratio'], y = df['Dataset'],palette = 'Set2',data = df)
#MULTI-VARIATE
sns.pairplot(df,hue = 'Gender',diag_kind = 'kde')

#CORRELATION
sns.heatmap(df.corr())

df.corr()['Dataset'].sort_values(ascending = False)

df['Gender']=df['Gender'].map({'Male':1,'Female':0})
df['Gender']=df['Gender'].astype('int64')
```

```
#SPLITTING THE DATASET
 x = df.iloc[0:400,0:-1]
 y = df.iloc[0:400,-1]
 x.head()
 y.head()
 y.value_counts()

 #BALANCING THE DATASET
 from imblearn.over_sampling import SMOTE
 from imblearn.combine import SMOTETomek
 smote = SMOTETomek(sampling_strategy='auto')
 x_bal,y_bal = smote.fit_resample(x,y)
 print(y.value_counts())
 print(y_bal.value_counts())
 #SCALING THE DATA
 from sklearn.preprocessing import StandardScaler
 scaler = StandardScaler()
 xtrain_scaled = scaler.fit_transform(xtrain)
 xtest_scaled = scaler.transform(xtest)
 names=x.columns
 x_bal=pd.DataFrame(x_bal,columns=names)
 x_bal.head()
 x_bal.tail()
```

```
y_bal.head()


#TRAINING AND TESTING

from sklearn.model_selection import train_test_split

xtrain,xtest,ytrain, ytest = train_test_split(x_bal,y_bal,test_size=0.2,random_state=49)


x_bal.tail()

y_bal.head()

xtrain.shape

xtest.shape


#MODEL BUILDING

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy_score

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear_model import LogisticRegression

from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix


#SUPPORT VECTOR MACHINE MODEL
svm = SVC()
RFmodel=RandomForestClassifier()
KNNmodel= KNeighborsClassifier()
scaler = StandardScaler()
xtrain = scaler.fit_transform(xtrain)
xtest = scaler.transform(xtest)
svm.fit(xtrain,ytrain)
SVMPred = svm.predict(xtest)
SVMaccuracy = accuracy_score(ytest, SVMPred)
SVMaccuracy
def svm_model(X_train, X_test, y_train, y_test):
    svm = SVC()
    svm.fit(X_train, y_train)
    SVMpred = svm.predict(X_test)
    SVMaccuracy = accuracy_score(y_test, SVMpred)
    print("SVM Accuracy Score: {}".format(SVMaccuracy))
    print("Classification Report:\n", classification_report(y_test, SVMpred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, SVMpred))
svm_model(xtrain, xtest, ytrain, ytest)
```

```
#KNN MODEL
KNNmodel.fit(xtrain,ytrain)
KNNpred = KNNmodel.predict(xtest)
KNNaccuracy = accuracy_score(KNNpred,ytest)
KNNaccuracy
def knn(X_train, X_test, y_train, y_test):
    knn_model = KNeighborsClassifier()
    knn_model.fit(X_train, y_train)
    KNNpred = knn_model.predict(X_test)
    KNNaccuracy = accuracy_score(y_test, KNNpred)
    print("KNN Accuracy Score: {}".format(KNNaccuracy))
    print("Classification Report:\n", classification_report(y_test, KNNpred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, KNNpred))
knn(xtrain, xtest, ytrain, ytest)

#RANDOM FOREST MODEL
RFmodel.fit(xtrain,ytrain)
RFpred=RFmodel.predict(xtest)
RFaccuracy = accuracy_score(RFpred,ytest)
RFaccuracy
def random_forest(xtrain, xtest, ytrain, ytest):
    rf = RandomForestClassifier()
    rf.fit(xtrain, ytrain)
    RFpred = rf.predict(xtest)
    RFaccuracy = accuracy_score(ytest, RFpred)
    print("Random Forest Accuracy Score: {}".format(RFaccuracy))
    print("Classification Report:\n", classification_report(ytest, RFpred))
    print("Confusion Matrix:\n", confusion_matrix(ytest, RFpred))
random_forest(xtrain, xtest, ytrain, ytest)

#DECISION TREE CLASSIFIER
from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier()
dt_model.fit(xtrain, ytrain)
dt_pred = dt_model.predict(xtest)
DTaccuracy = accuracy_score(ytest, dt_pred)
DTaccuracy
def decision_tree(X_train, X_test, y_train, y_test):
    dt = DecisionTreeClassifier()
    dt.fit(X_train, y_train)
    y_pred = dt.predict(X_test)
    print("Decision Tree Accuracy Score: {}".format(accuracy_score(y_test, y_pred)))
    print("Classification Report:\n", classification_report(y_test, y_pred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
decision_tree(xtrain, xtest, ytrain, ytest)


#LOGISTIC REGRESSION MODEL
from sklearn.linear_model import LogisticRegression
log_reg_model = LogisticRegression()
log_reg_model.fit(xtrain, ytrain)
y_pred = log_reg_model.predict(xtest)
log_reg_accuracy = accuracy_score(ytest, y_pred)
log_reg_accuracy
def logistic_regression(X_train, X_test, y_train, y_test):
    lr = LogisticRegression(max_iter=1000)
    lr.fit(X_train, y_train)
    LRpred = lr.predict(X_test)
    LRaccuracy = accuracy_score(y_test, LRpred)
    print("Logistic Regression Accuracy Score: {}".format(LRaccuracy))
    print("Classification Report:\n", classification_report(y_test, LRpred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, LRpred))
logistic_regression(xtrain, xtest, ytrain, ytest)


#COMPARING THE MODELS
svm = SVC()
dt_model = DecisionTreeClassifier()
RFmodel = RandomForestClassifier()
```

```
KNNmodel = KNeighborsClassifier()
log_reg_model = LogisticRegression()
svm.fit(xtrain,ytrain)
dt_model.fit(xtrain,ytrain)
RFmodel.fit(xtrain,ytrain)
KNNmodel.fit(xtrain,ytrain)
log_reg_model.fit(xtrain,ytrain)
pred0=svm.predict(xtrain)
pred1=dt_model.predict(xtrain)
pred2=RFmodel.predict(xtrain)
pred3=KNNmodel.predict(xtrain)
pred4=log_reg_model.predict(xtrain)

print('SVM:',accuracy_score(ytrain,pred0))
print('Decision Tree:',accuracy_score(ytrain,pred1))
print('Random Forest:',accuracy_score(ytrain,pred2))
print('KNN:',accuracy_score(ytrain,pred3))
print('Logistic:',accuracy_score(ytrain,pred4))
y_pred1=svm.predict(xtest)
y_pred2=dt_model.predict(xtest)
y_pred3=RFmodel.predict(xtest)
y_pred4=KNNmodel.predict(xtest)
y_pred5 =log_reg_model.predict(xtest)
```

```
print('Support Vector:',accuracy_score(ytest,y_pred1))
print('Decision Tree:',accuracy_score(ytest,y_pred2))
print('Random Forest:',accuracy_score(ytest,y_pred3))
print('KNN:',accuracy_score(ytest,y_pred4))
print('Logistic:',accuracy_score(ytest,y_pred5))


#HYPERPARAMETER TUNNING
def randomForest(xtrain,xtest,ytrain,ytest):
  model = RandomForestClassifier(verbose=2,n_estimators=120,max_features='log2',max_depth=10,criterion='entropy')
  model.fit(xtrain,ytrain)
  y_tr=model.predict(xtrain)
  print("Accuracy Score {}".format(accuracy_score(ytrain,y_tr)))
  y_pr=model.predict(xtest)
  print("Accuracy Score {}".format(accuracy_score(ytest,y_pr)))
randomForest(xtrain,xtest,ytrain,ytest)


from sklearn.metrics import f1_score
RFmodel.fit(xtrain,ytrain)
y_pred3 = RFmodel.predict(xtest)
print("Accuracy Score {}".format(accuracy_score(ytest,y_pred3)))
print("f1_Score {}".format(f1_score(y_pred3,ytest,average='weighted')))
```

```
from sklearn.model_selection import GridSearchCV
# Hyperparameter tuning for SVM
svm_params = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto']
}

svm_grid = GridSearchCV(SVC(), svm_params, cv=5)
svm_grid.fit(xtrain, ytrain)

# Best model
best_svm = svm_grid.best_estimator_
y_pred_svm = best_svm.predict(xtest)

# Evaluation
print("SVM Accuracy: ", accuracy_score(ytest, y_pred_svm))
print("Classification Report:\n", classification_report(ytest, y_pred_svm))
print("Confusion Matrix:\n", confusion_matrix(ytest, y_pred_svm))
print(best_svm)

# Hyperparameter tuning for Decision Tree
```

```
dt_params = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10]
}

dt_grid = GridSearchCV(DecisionTreeClassifier(), dt_params, cv=5)
dt_grid.fit(xtrain, ytrain)

# Best model
best_dt = dt_grid.best_estimator_
y_pred_dt = best_dt.predict(xtest)

# Evaluation
print("Decision Tree Accuracy: ", accuracy_score(ytest, y_pred_dt))
print("Classification Report:\n", classification_report(ytest, y_pred_dt))
print("Confusion Matrix:\n", confusion_matrix(ytest, y_pred_dt))
print(best_dt)

# Hyperparameter tuning for Random Forest
rf_params = {
    'n_estimators': [10, 50, 100],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10]
}

rf_grid = GridSearchCV(RandomForestClassifier(), rf_params, cv=5)
rf_grid.fit(xtrain, ytrain)

# Best model
best_rf = rf_grid.best_estimator_
y_pred_rf = best_rf.predict(xtest)

# Evaluation
print("Random Forest Accuracy: ", accuracy_score(ytest, y_pred_rf))
print("Classification Report:\n", classification_report(ytest, y_pred_rf))
print("Confusion Matrix:\n", confusion_matrix(ytest, y_pred_rf))
print(best_rf)

# Hyperparameter tuning for KNN
knn_params = {
    'n_neighbors': [3, 5, 7, 9],
    'weights': ['uniform', 'distance']
}

knn_grid = GridSearchCV(KNeighborsClassifier(), knn_params, cv=5)
```

```
knn_grid.fit(xtrain, ytrain)


# Best model

best_knn = knn_grid.best_estimator_

y_pred_knn = best_knn.predict(xtest)


# Evaluation

print("KNN Accuracy: ", accuracy_score(ytest, y_pred_knn))

print("Classification Report:\n", classification_report(ytest, y_pred_knn))

print("Confusion Matrix:\n", confusion_matrix(ytest, y_pred_knn))

print(best_knn)


# Hyperparameter tuning for Logistic Regression

lr_params = {

    'C': [0.001, 0.01, 0.1, 1, 10],

    'solver': ['liblinear', 'saga']

}


lr_grid = GridSearchCV(LogisticRegression(), lr_params, cv=5)

lr_grid.fit(xtrain, ytrain)


# Best model

best_lr = lr_grid.best_estimator_
```

```
y_pred_lr = best_lr.predict(xtest)


# Evaluation
print("Logistic Regression Accuracy: ", accuracy_score(ytest, y_pred_lr))
print("Classification Report:\n", classification_report(ytest, y_pred_lr))
print("Confusion Matrix:\n", confusion_matrix(ytest, y_pred_lr))
print(best_lr)


#EVALUATING PERFORMANCE OF THE MODEL & SAVING THE MODEL
from sklearn.metrics import f1_score
RFmodel.fit(xtrain,ytrain)
y_pred3 = RFmodel.predict(xtest)
print("Accuracy Score {}".format(accuracy_score(ytest,y_pred3)))
print("f1_Score {}".format(f1_score(y_pred3,ytest,average='weighted')))
model = RandomForestClassifier(verbose=2,n_estimators=200,max_features='log2',max_depth=None,criterion='entropy')
model.fit(xtrain,ytrain)
import pickle
pickle.dump(model,open('liver_analysis.pk1','wb'))
pickle.dump(scaler,open('scaling.pkl','wb'))


#TESTING WITH VALUES
```

```
input=[[65,1,0.7,0.1,187,16,18,6.8,3.3,0.90]]

input=scaler.transform(input)

prediction = model.predict(input)

prediction


input=[[17,0,0.9,0.3,202,22,19,7.4,4.1,1.2]]

input=scaler.transform(input)

prediction = model.predict(input)

prediction
```

APP.PY

```
# app.py

der_template('home.html')


@app.route('/predict', methods=['POST', 'GET'])

def index():

    return render_template("index.html")
```

```
@app.route('/submit', methods=['POST', 'GET'])
def submit():
    age = request.form['age']
    gender = request.form['gender']
    tb = request.form['tb']
db = request.form['db']
    ap = request.form['ap']
    aa1 = request.form['aa1']
    aa2 = request.form['aa2']
tp = request.form['tp']
    a = request.form['a']
agr = request.form['agr']

    data = [[float(age), float(gender), float(tb), float(db), float(ap), float(aa1), float(aa2), float(tp), float(a), float(agr)]]

    model = pickle.load(open(r'C:\Users\hp\Downloads\Smartinternz_project\liver_analysis (1).pk1', 'rb'))
    scale = pickle.load(open(r'C:\Users\hp\Downloads\Smartinternz_project\scaling (1).pkl', 'rb'))
scaled_data = scale.transform(data)
    prediction= model.predict(scaled_data)[0]
    if prediction == 1:

        return render_template('chance.html', prediction='You have a liver disease
problem, consult a doctor !!!.')

elif prediction == 2:

        return render_template('chance1.html', prediction='You do not have a liver
disease problem \U0001f600 !')



if _name=='main_':

    port = int(os.environ.get('PORT', 5000))

app.run(debug=False)
```

Home.html

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Liver Patient Analysis</title>

<style>

        body {

            font-family: 'Arial', sans-serif;

            background-image:
url('https://media.istockphoto.com/id/1272365959/photo/hepatitis-virus-with-
human-
liver.jpg?s=612x612&w=0&k=20&c=0Y8eNGsdsl3uqCmT55eqYD4keGSByw_z
VKtm8VJowqg='); /* Replace with your background image URL */
            background-size: 100%;
            background-position: center;
            margin: 0;
            padding: center;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            color: #fff;
        }
        .box {
            max-width: 550px;
            width: 100%;
            background: rgba(255,255,255,0.8); /* Semi-transparent white */
            border-radius: 12px;
            box-shadow: 0 4px 20px rgba(0, 0, 0, 0.3);
            text-align: center;
            padding: 30px;
            transition: transform 0.2s;
        }
        .box:hover {
            transform: translateY(-5px);
        }
        h1 {
            font-size: 2.5em;
            color: #007bff;
            margin-bottom: 15px;
```

```
}
p {
    font-size: 1.1em;
    line-height: 1.6;
    color: #333;
    margin-bottom: 20px;
}
a {
    display: inline-block;
    margin-top: 20px;
    padding: 12px 24px;
    background-color: #007bff;
    color: white;
    text-decoration: none;
    border-radius: 5px;
    font-weight: bold;
    transition: background-color 0.3s, transform 0.2s;
}
a:hover {
    background-color: #0056b3;
    transform: translateY(-2px);
}
@media (max-width: 600px) {
    h1 {
        font-size: 2em;
    }
    p {
        font-size: 1em;
```

```
            }
            a {
                padding: 10px 20px;
            }
        }
</style>
</head>
<body>
<div class="box">
<h1>Liver Disease Prediction</h1>
<p>
            Simply Enter your health metrics, and our machine learning model will analyze the data to provide you with a prediction regarding liver disease.
</p>
<a href="/predict">Predict</a>
</div>
</body>
</html>
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Liver Disease Prediction</title>
```

```
<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">
<style>
    body {
        font-family: 'Roboto', sans-serif;
        background-color:rgba(15, 15, 15, 0.7);
        color: #b18f8f; /* Sky blue background color */
        background-image: url('https://wallpapercave.com/wp/wp5935595.jpg'); /* Replace with your image URL */
        background-size: cover ; /* Cover the whole viewport */
        background-position: cover; /* Center the background image */
        background-blend-mode:overlay; /* Blend the color and image */
        margin: 0;
        padding: 0;
        display: flex;
        justify-content: center;
        align-items: center;
        height: 0vh;
        color: #050505;
    }
    .container {
        background: #92c2d1(236, 230, 230, 0.9);
        border-radius: 12px;
        box-shadow: 0 4px 20px rgba(0, 0, 0, 0.2);
        width: 90%; /* Set to full width */
        max-width: 100vw; /* Allow full width of viewport */
        height: 100%; /* Alow full height */
        box-sizing: border-box; /* Ensure padding is included in width */
    }
    h3 {
        text-align: center;
        color: #f1f1f1;
        margin-bottom: 20px;
        font-weight: 700;
        text-shadow: 1px 1px 2px rgba(122, 7, 7, 0.2);
    }
    label {
        display: block;
        margin: 10px 0 5px;
        font-weight: 600;
        color: #ffffff;
    }
    input[type="number"],
    select {
        width: 75%;
        padding: 12px;
        margin-bottom: 15px;
        border: 2px solid #4277af;
        border-radius: 5px;
        transition: border-color 0.3s;
    }
    input[type="number"]:focus,
    select:focus {
        border-color: #4f8dcf;
        outline: none;
    }
```

```
        input[type="submit"] {
            width: 10%;
            padding: 12px;
            background-color: #086dd8;
            color: white;
            border: none;
            border-radius: 5px;
            font-weight: bold;
            cursor: pointer;
            transition: background-color 0.3s;
            font-size: 16px;
        }
        input[type="submit"]:hover {
            background-color: #4b6177;
        }
        @media (max-width: 600px) {
            .container {
                padding: 30px;
            }
        }
</style>
</head>
<body>
<div class="container">
<h3>Enter Your Details for Liver Disease Prediction</h3>
<form action='/submit' method='POST'>
<label for="age">Age</label>
<input type="number" id="age" min="0" max="120" name="age" placeholder="Enter your age..." required>
```

```
<label for="gender">Gender</label>
<select id="gender" name="gender" required>
<option value="0">Male</option>
<option value="1">Female</option>
</select>

<label for="tb">Total Bilirubin</label>
<input type="number" id="tb" min="0" name="tb" step="0.1" placeholder="Total Bilirubin value" required>

<label for="db">Direct Bilirubin</label>
<input type="number" id="db" min="0" name="db" step="0.1" placeholder="Direct Bilirubin value" required>

<label for="ap">Alkaline Phosphotase</label>
<input type="number" id="ap" min="0" name="ap" placeholder="Alkaline Phosphotase" required>

<label for="aa1">Alamine Aminotransferase</label>
<input type="number" id="aa1" min="0" name="aa1" placeholder="Alamine Aminotransferase" required>

<label for="aa2">Aspartate Aminotransferase</label>
<input type="number" id="aa2" min="0" name="aa2" placeholder="Aspartate Aminotransferase" required>

<label for="tp">Total Proteins</label>
<input type="number" id="tp" min="0" step="0.1" name="tp" placeholder="Total Proteins" required>
```

```
<label for="a">Albumin</label>
<input type="number" id="a" min="0" step="0.1" name="a"
placeholder="Albumin" required>

<label for="agr">Albumin and Globulin Ratio</label>
<input type="number" id="agr" step="0.01" name="agr" placeholder="Albumin
and Globulin Ratio" required><br>

<input type='submit' value='Submit'>
</form>
</div>
</body>
</html>
```

Chance.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>LIVER DISEASE PREDICTION</title>
<link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=
swap" rel="stylesheet">
<style>
    body {
        font-family: 'Roboto', sans-serif;
        background: url('https://thumbs.dreamstime.com/b/cartoon-doctor-holding-large-liver-front-world-to-raise-awareness-hepatitis-day-customizable-illustration-322191059.jpg') no-repeat center center fixed; /* Add your background image URL here */
        background-size: 100%;
        margin: 0;
        padding: center;
        display: flex;
        justify-content: center;
        align-items: center;
        height: 90vh;
        color: #ecf0f1; /* Light text color */
    }
    .content {
        background: rgba(245, 195, 188, 0.90); /* Semi-transparent background for the content */
        border-radius: 12px;
        padding: 50px;
        text-align: center;
        box-shadow: 0 4px 20px rgba(0, 0, 0, 0.4);
    }
    .title {
        font-size: 2.5em;
        font-style:normal;
        margin-bottom: 20px;
        color: #b91144; /* Maroon color for the title */
    }
    .result {
        font-size: 1.5em;
        font-family: Verdana, Geneva, Tahoma, sans-serif;
        margin-top: 20px;
```

```
            color: #000000; /* Green color for the result */
        }
</style>
</head>
<body>
<div class="content">
<div class="title-box">
<div class="title">LIVER DISEASE PREDICTION</div>
<div class="result">{{ prediction }}</div>
</div>
</div>
</body>
</html>
```

Chance1.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Liver Disease Prediction</title>
<link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=
swap" rel="stylesheet">
<style>
        body {
```

```
    font-family: 'Roboto', sans-serif;
    background: url('https://cdn.vectorstock.com/i/500p/78/18/an-abstract-human-liver-cute-vector-36327818.jpg') no-repeat center center fixed; /* Add your background image URL here */
    background-size: cover;
    margin: 0;
    padding: center;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    color: #ecf0f1; /* Light text color */
  }
  .content {
    background: rgba(245, 195, 188, 0.90); /* Semi-transparent background for the content */
    border-radius: 12px;
    padding: 50px;
    text-align: center;
    box-shadow: 0 4px 20px rgba(0, 0, 0, 0.4);
  }
  .title {
    font-size: 2em;
    margin-bottom: 20px;
    color: #c0392b; /* Maroon color for the title */
  }
  .result {
    font-size: 1.5em;
    margin-top: 20px;
    color: #27ae60; /* Green color for the result */
  }
```

```
</style>
</head>
<body>
<div class="content">
<div class="title-box">
<div class="title">Liver Disease Prediction</div>
<div class="result">{{ prediction }}</div>
</div>
</div>
</body>
</html>
```

10.2 GitHub & Project Demo Link

https://github.com/B-Prasanthi-4561/Prediction-and-Analysis-of-Liver-Patient-Data-Using-Machine-Learning

10.3 Video Demonstration Link

https://drive.google.com/file/d/1s06aUbccyBqmZNZG8ibD3YB2WWWNFKTj/view?usp=sharing